# PIC18F87K22 FAMILY

## 64/80-Pin, High-Performance, 1-Mbit Enhanced Flash MCUs with 12-Bit A/D and XLP Technology

### Low-Power Features:

- Power-Managed modes:
  - Run: CPU on, peripherals on
  - Idle: CPU off, peripherals on
  - Sleep: CPU off, peripherals off
- Two-Speed Oscillator Start-up
- Fail-Safe Clock Monitor
- Power-Saving Peripheral Module Disable (PMD)
- Ultra Low-Power Wake-up
- Fast Wake-up, 1 μs Typical
- Low-Power WDT, 300 nA Typical
- Ultra Low 50 nA Input Leakage
- Run mode Currents Down to 5.5 μA, Typical
- Idle mode Currents Down to 1.7 μA Typical
- Sleep mode Currents Down to Very Low 20 nA, Typical
- RTCC Current Downs to Very Low 700 nA, Typical

### Special Microcontroller Features:

- Operating Voltage Range: 1.8V to 5.5V
- On-Chip 3.3V Regulator
- Operating Speed up to 64 MHz
- Up to 128 Kbytes On-Chip Flash Program Memory
- Data EEPROM of 1,024 Bytes
- 4K x 8 General Purpose Registers (SRAM)
- 10,000 Erase/Write Cycle Flash Program Memory, Minimum
- 1,000,000 Erase/write Cycle Data EEPROM Memory, Typical
- Flash Retention: 40 Years, Minimum
- Three Internal Oscillators: LF-INTRC (31 kHz), MF-INTOSC (500 kHz) and HF-INTOSC (16 MHz)
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 4,194s (about 70 minutes)
- In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug via Two Pins
- Programmable:
  - BOR
  - LVD

| Device | Program Memory | | Data Memory | | I/O | 12-Bit A/D (ch) | CCP/ ECCP (PWM) | MSSP | | EUSART | Comparators | Timers 8/16-Bit | External Bus | CTMU | RTCC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Flash (bytes) | # Single-Word Instructions | SRAM (bytes) | EEPROM (bytes) | | | | | SPI | Master $I^2C$ | | | | | |
| PIC18F65K22 | 32K | 16,383 | 2K | 1K | 53 | 16 | 5/3 | 2 | Y | Y | 2 | 3 | 4/4 | N | Y | Y |
| PIC18F66K22 | 64K | 32,768 | 4K | 1K | 53 | 16 | 7/3 | 2 | Y | Y | 2 | 3 | 6/5 | N | Y | Y |
| PIC18F67K22 | 128K | 65,536 | 4K | 1K | 53 | 16 | 7/3 | 2 | Y | Y | 2 | 3 | 6/5 | N | Y | Y |
| PIC18F85K22 | 32K | 16,383 | 2K | 1K | 69 | 24 | 5/3 | 2 | Y | Y | 2 | 3 | 4/4 | Y | Y | Y |
| PIC18F86K22 | 64K | 32,768 | 4K | 1K | 69 | 24 | 7/3 | 2 | Y | Y | 2 | 3 | 6/5 | Y | Y | Y |
| PIC18F87K22 | 128K | 65,536 | 4K | 1K | 69 | 24 | 7/3 | 2 | Y | Y | 2 | 3 | 6/5 | Y | Y | Y |

© 2009-2018 Microchip Technology Inc.

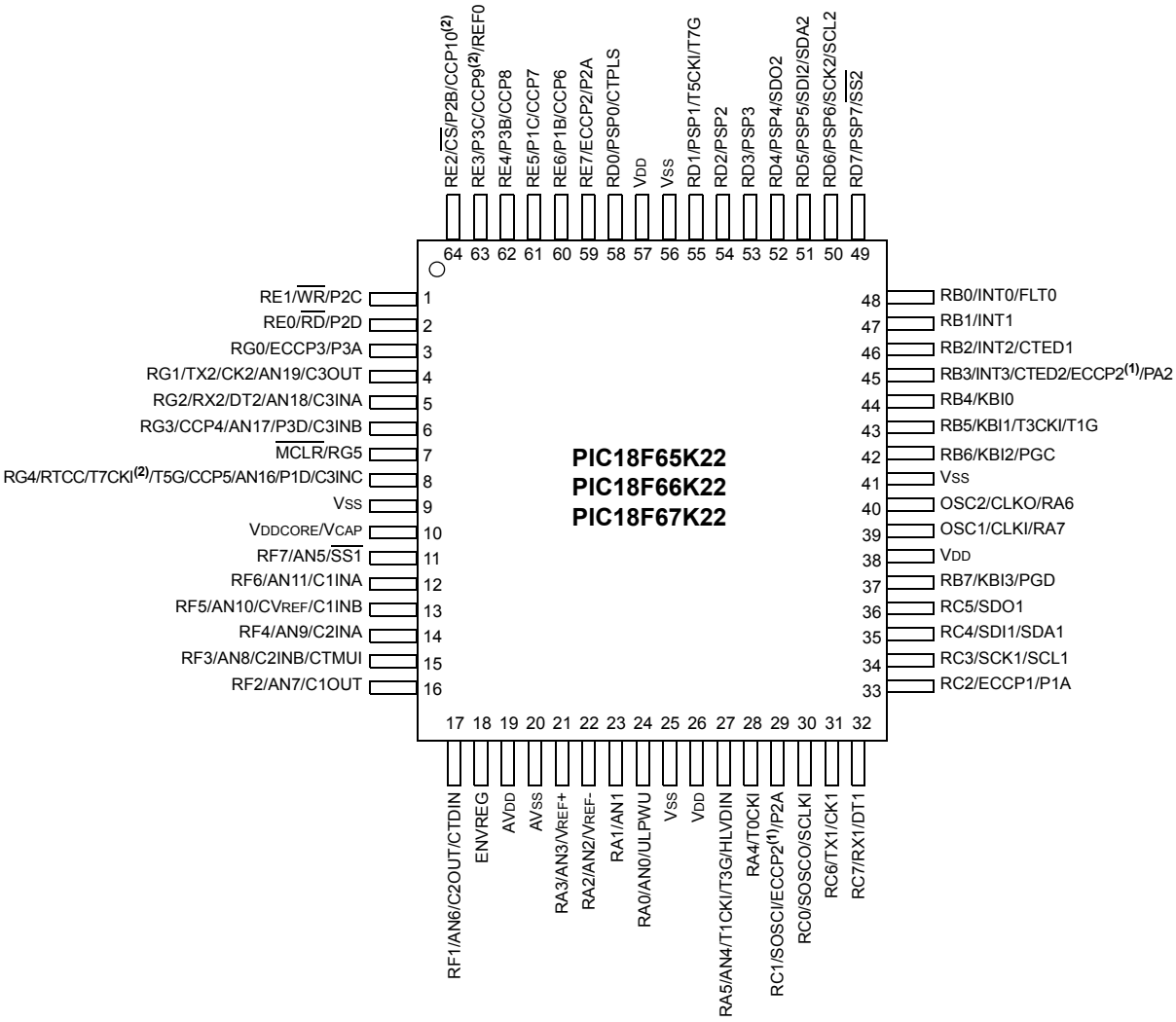
### Peripheral Highlights:

- Up to Ten CCP/ECCP modules:
  - Up to seven Capture/Compare/PWM (CCP) modules
  - Three Enhanced Capture/Compare/PWM (ECCP) modules
- Up to Eleven 8/16-Bit Timer/Counter modules:
  - Timer0 – 8/16-bit timer/counter with 8-bit programmable prescaler
  - Timer1,3 – 16-bit timer/counter
  - Timer2,4,6,8 – 8-bit timer/counter
  - Timer5,7 – 16-bit timer/counter for 64k and 128k parts
  - Timer10,12 – 8-bit timer/counter for 64k and 128k parts
- Three Analog Comparators
- Configurable Reference Clock Output
- Hardware Real-Time Clock and Calendar (RTCC) module with Clock, Calendar and Alarm Functions
- Charge Time Measurement Unit (CTMU):
  - Capacitance measurement for mTouch® sensing solution
  - Time measurement with 1 ns typical resolution
  - Integrated temperature sensor
- High-Current Sink/Source 25 mA/25 mA (PORTB and PORTC)
- Up to Four External Interrupts
- Two Master Synchronous Serial Port (MSSP) modules:
  - 3/4-wire SPI (supports all four SPI modes)
  - $I^2C$ Master and Slave modes
- Two Enhanced Addressable USART modules:
  - LIN/J2602 support
  - Auto-Baud Detect (ABD)
- 12-Bit A/D Converter with up to 24 Channels:
  - Auto-acquisition and Sleep operation
  - Differential input mode of operation
- Integrated Voltage Reference


© 2009-2018 Microchip Technology Inc.

## Pin Diagrams – PIC18F6XK22

**64-Pin TQFP, QFN**

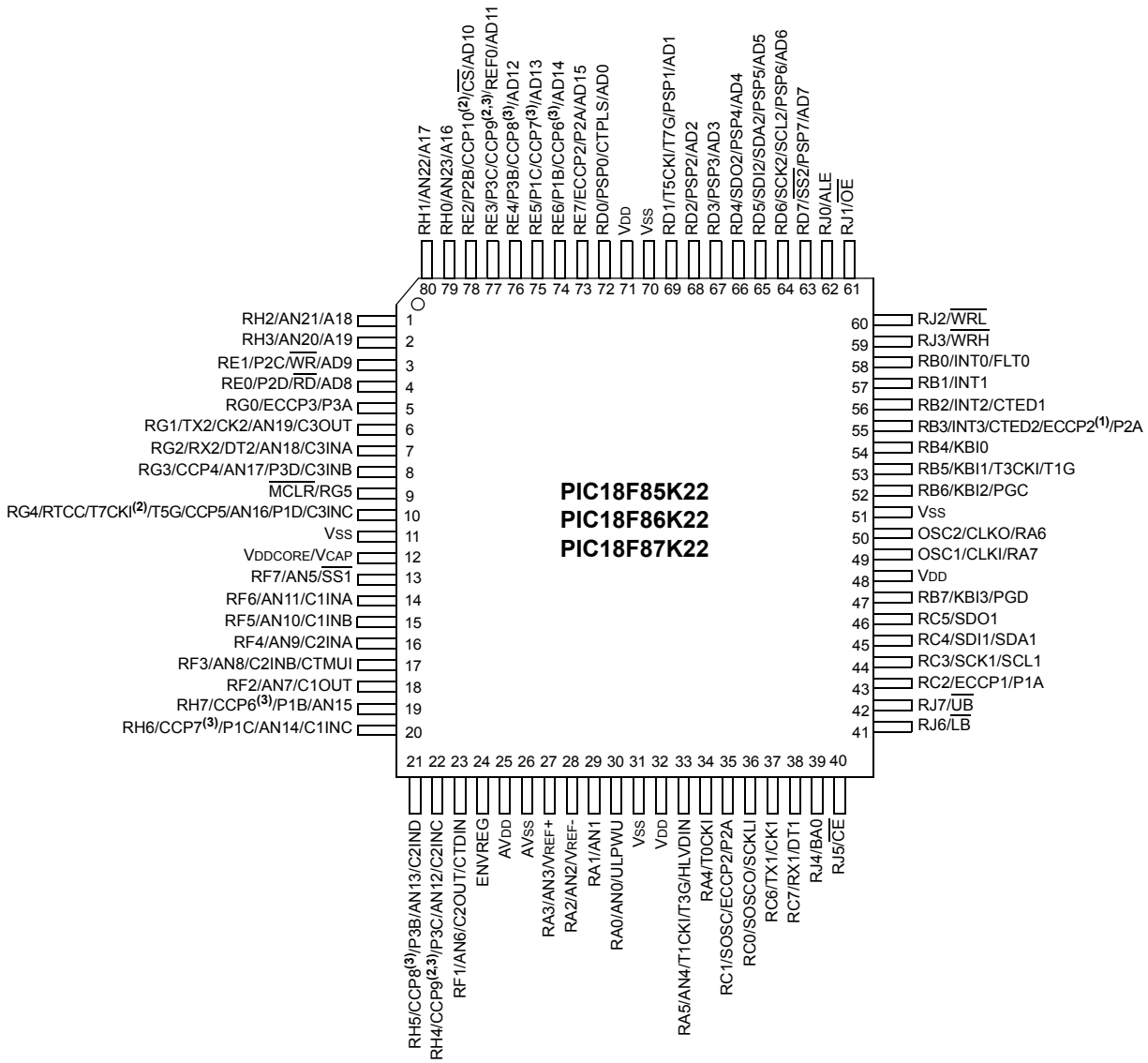



**Note 1:** The ECCP2 pin placement depends on the CCP2MX Configuration bit setting and whether the device is in Microcontroller or Extended Microcontroller mode.

**2:** Not available on the PIC18F65K22 and PIC18F85K22 devices.

## Pin Diagrams – PIC18F8XK22

**80-Pin TQFP**




**Note 1:** The ECCP2 pin placement depends on the CCP2MX Configuration bit setting and whether the device is in Microcontroller or Extended Microcontroller mode.

**2:** Not available on the PIC18F65K22 and PIC18F85K22 devices.

**3:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

## Table of Contents

1.0 Device Overview ..... 7
2.0 Guidelines for Getting Started with PIC18FXXKXX Microcontrollers ..... 35
3.0 Oscillator Configurations ..... 40
4.0 Power-Managed Modes ..... 54
5.0 Reset ..... 69
6.0 Memory Organization ..... 83
7.0 Flash Program Memory ..... 107
8.0 External Memory Bus ..... 117
9.0 Data EEPROM Memory ..... 128
10.0 8 x 8 Hardware Multiplier ..... 134
11.0 Interrupts ..... 136
12.0 I/O Ports ..... 160
13.0 Timer0 Module ..... 187
14.0 Timer1 Module ..... 190
15.0 Timer2 Module ..... 202
16.0 Timer3/5/7 Modules ..... 204
17.0 Timer4/6/8/10/12 Modules ..... 216
18.0 Real-Time Clock and Calendar (RTCC) ..... 219
19.0 Capture/Compare/PWM (CCP) Modules ..... 236
20.0 Enhanced Capture/Compare/PWM (ECCP) Module ..... 249
21.0 Master Synchronous Serial Port (MSSP) Module ..... 272
22.0 Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) ..... 318
23.0 12-Bit Analog-to-Digital Converter (A/D) Module ..... 342
24.0 Comparator Module ..... 358
25.0 Comparator Voltage Reference Module ..... 366
26.0 High/Low-Voltage Detect (HLVD) ..... 369
27.0 Charge Time Measurement Unit (CTMU) ..... 375
28.0 Special Features of the CPU ..... 392
29.0 Instruction Set Summary ..... 419
30.0 Development Support ..... 469
31.0 Electrical Characteristics ..... 473
32.0 Packaging Information ..... 515
Appendix A: Revision History ..... 523
Appendix B: Migration From PIC18F87J11 and PIC18F8722 to PIC18F87K22 ..... 524
The Microchip WebSite ..... 525
Customer Change Notification Service ..... 525
Customer Support ..... 525

## 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.


© 2009-2018 Microchip Technology Inc.

## 1.0 DEVICE OVERVIEW

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

- PIC18F65K22
- PIC18F66K22
- PIC18F67K22
- PIC18F85K22
- PIC18F86K22
- PIC18F87K22

This family combines the traditional advantages of all PIC18 microcontrollers – namely, high computational performance and a rich feature set – with an extremely competitive price point. These features make the PIC18F87K22 family a logical choice for many high-performance applications where price is a primary consideration.

### 1.1 Core Features

#### 1.1.1 XLP TECHNOLOGY

All of the devices in the PIC18F87K22 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.
- **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.
- **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.
- **XLP:** An extra low-power Sleep, BOR, RTCC and Watchdog Timer

#### 1.1.2 OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F87K22 family offer different oscillator options, allowing users a range of choices in developing application hardware. These include:

- External Resistor/Capacitor (RC); RA6 available
- External Resistor/Capacitor with Clock Out (RCIO)
- Three External Clock modes:
  - External Clock (EC); RA6 available
  - External Clock with Clock Out (ECIO)
  - External Crystal (XT, HS, LP)
- A Phase Lock Loop (PLL) frequency multiplier, available to the External Oscillator modes, which allows clock speeds of up to 64 MHz. PLL can also be used with the internal oscillator.
- An internal oscillator block that provides a 16 MHz clock (±2% accuracy) and an INTRC source (approximately 31 kHz, stable over temperature and $V_{DD}$)
  - Operates as HF-INTOSC or MF-INTOSC when block selected for 16 MHz or 500 kHz
  - Frees the two oscillator pins for use as additional general purpose I/O

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 MEMORY OPTIONS

The PIC18F87K22 family provides ample room for application code, from 32 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 40 years.

The Flash program memory is readable and writable. During normal operation, the PIC18F87K22 family also provides plenty of room for dynamic application data with up to 3,862 bytes of data RAM.

#### 1.1.4 EXTERNAL MEMORY BUS

Should 128 Kbytes of memory be inadequate for an application, the 80-pin members of the PIC18F87K22 family have an External Memory Bus (EMB) enabling the controller's internal Program Counter to address a memory space of up to 2 Mbytes. This is a level of data access that few 8-bit devices can claim and enables:

- Using combinations of on-chip and external memory of up to 2 Mbytes
- 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 PIC18F87K22 family implements the optional extension to the PIC18 instruction set, adding eight 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

All devices share the same rich set of peripherals except that the devices with 32 Kbytes of program memory (PIC18F65K22 and PIC18F85K22) have two less CCPs and three less timers. This provides a smooth migration path within the device family as applications evolve and grow.

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.

All of the devices in the family share the same rich set of peripherals, except for those with 32 Kbytes of program memory (PIC18F65K22 and PIC18F85K22). Those devices have two less CCPs and three less timers.

The PIC18F87K22 family is also largely pin compatible with other PIC18 families, such as the PIC18F8720 and PIC18F8722 and the PIC18F85J11. This allows a new dimension to the evolution of applications, allowing developers to select different price points within Microchip's PIC18 portfolio, while maintaining a similar feature set.

## 1.2 Other Special Features

- **Communications:** The PIC18F87K22 family incorporates a range of serial communication peripherals, including two Enhanced USARTs (EUSART) that support LIN/J2602, and two Master SSP modules, capable of both SPI and $I^2C$ (Master and Slave) modes of operation.
- **CCP Modules:** PIC18F87K22 family devices incorporate up to seven Capture/Compare/PWM (CCP) modules. Up to six different time bases can be used to perform several different operations at once.
- **ECCP Modules:** The PIC18F87K22 family has three Enhanced CCP (ECCP) modules to maximize flexibility in control applications:
  - Up to eight different time bases for performing several different operations at once
  - Up to four PWM outputs for each module, for a total of 12 PWMs
  - Other beneficial features, such as polarity selection, programmable dead time, auto-shutdown and restart, and Half-Bridge and Full-Bridge Output modes
- **12-Bit A/D Converter:** The PIC18F87K22 family has differential ADC. It 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.
- **Charge Time Measurement Unit (CTMU):** The CTMU is a flexible analog module that provides accurate differential time measurement between pulse sources, as well as asynchronous pulse generation.
- Together with other on-chip analog modules, the CTMU can precisely measure time, measure capacitance or relative changes in capacitance, or generate output pulses that are independent of the system clock.
- **LP Watchdog Timer (WDT):** This enhanced version incorporates a 22-bit prescaler, allowing an extended time-out range that is stable across operating voltage and temperature. See **Section 31.0 "Electrical Characteristics"** for time-out periods.
- **Real-Time Clock and Calendar Module (RTCC):** The RTCC module is intended for applications requiring that accurate time be maintained for extended periods of time with minimum to no intervention from the CPU.
- The module is a 100-year clock and calendar with automatic leap year detection. The range of the clock is from 00:00:00 (midnight) on January 1, 2000 to 23:59:59 on December 31, 2099.


© 2009-2018 Microchip Technology Inc.

## 1.3 Details on Individual Family Members

Devices in the PIC18F87K22 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 these ways:

- Flash Program Memory:
  - PIC18FX5K22 (PIC18F65K22 and PIC18F85K22) – 32 Kbytes
  - PIC18FX6K22 (PIC18F66K22 and PIC18F86K22) – 64 Kbytes
  - PIC18FX7K22 (PIC18F67K22 and PIC18F87K22) – 128 Kbytes
- Data RAM:
  - All devices except PIC18FX5K22 – 4 Kbytes
  - PIC18FX5K22 – 2 Kbytes
- I/O Ports:
  - PIC18F6XK22 (64-pin devices) – seven bidirectional ports
  - PIC18F8XK22 (80-pin devices) – nine bidirectional ports
- CCP modules:
  - PIC18FX5K22 (PIC18F65K22 and PIC18F85K22) – five CCP modules
  - PIC18FX6K22 and PIC18FX7K22 (PIC18F66K22, PIC18F86K22, PIC18F67K22, and PIC18F87K22) – seven CCP modules
- Timer modules:
  - PIC18FX5K22 (PIC18F65K22 and PIC18F85K22) – Four 8-bit timer/counters and four 16-bit timer/counters
  - PIC18FX6K22 and PIC18FX7K22 (PIC18F66K22, PIC18F86K22, PIC18F67K22, and PIC18F87K22) – Six 8-bit timer/counters and five 16-bit timer/counters
- A/D Channels:
  - PIC18F6XK22 (64-pin devices) – 24 channels
  - PIC18F8XK22 (80-pin devices) – 16 channels

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.

© 2009-2018 Microchip Technology Inc.

**TABLE 1-1: DEVICE FEATURES FOR THE PIC18F6XK22 (64-PIN DEVICES)**

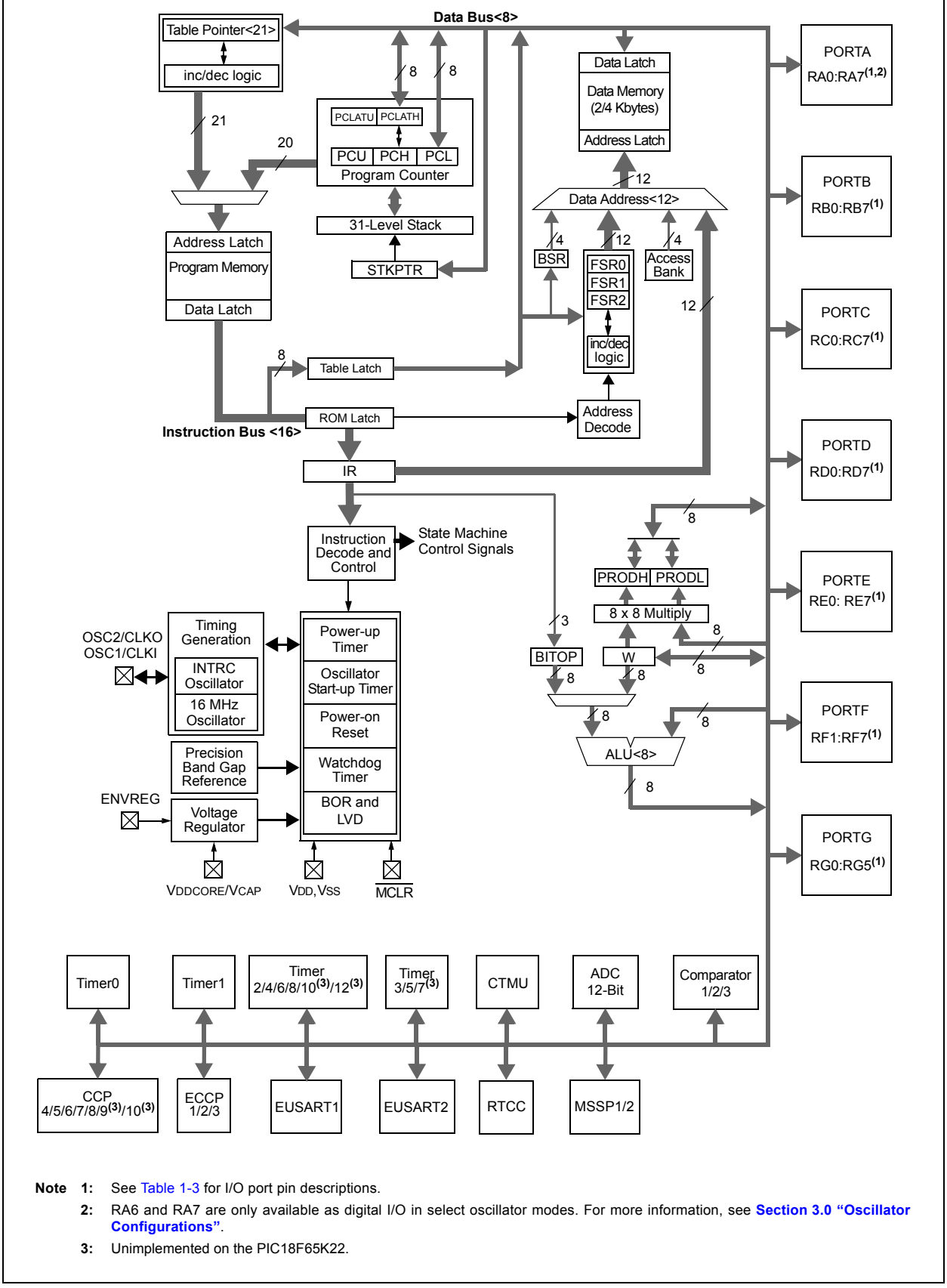
| Features | PIC18F65K22 | PIC18F66K22 | PIC18F67K22 |
|---|---|---|---|
| Operating Frequency | DC – 64 MHz | | |
| Program Memory (Bytes) | 32K | 64K | 128K |
| Program Memory (Instructions) | 16,384 | 32,768 | 65,536 |
| Data Memory (Bytes) | 2K | 4K | 4K |
| Interrupt Sources | 42 | 48 | |
| I/O Ports | Ports A, B, C, D, E, F, G | | |
| Parallel Communications | Parallel Slave Port (PSP) | | |
| Timers | 8 | 11 | |
| Comparators | 3 | | |
| CTMU | Yes | | |
| RTCC | Yes | | |
| Capture/Compare/PWM (CCP) Modules | 5 | 7 | 7 |
| Enhanced CCP (ECCP) Modules | 3 | | |
| Serial Communications | Two MSSPs and two Enhanced USARTs (EUSART) | | |
| 12-Bit Analog-to-Digital Module | 16 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 QFN, 64-Pin TQFP | | |

**TABLE 1-2: DEVICE FEATURES FOR THE PIC18F8XK22 (80-PIN DEVICES)**

| Features | PIC18F85K22 | PIC18F86K22 | PIC18F87K22 |
|---|---|---|---|
| Operating Frequency | DC – 64 MHz | | |
| Program Memory (Bytes) | 32K | 64K | 128K |
| | (Up to 2 Mbytes with Extended Memory) | | |
| Program Memory (Instructions) | 16,384 | 32,768 | 65,536 |
| Data Memory (Bytes) | 2K | 4K | 4K |
| Interrupt Sources | 42 | 48 | |
| I/O Ports | Ports A, B, C, D, E, F, G, H, J | | |
| Parallel Communications | Parallel Slave Port (PSP) | | |
| Timers | 8 | 11 | |
| Comparators | 3 | | |
| CTMU | Yes | | |
| RTCC | Yes | | |
| Capture/Compare/PWM (CCP) Modules | 5 | 7 | 7 |
| Enhanced CCP (ECCP) Modules | 3 | | |
| Serial Communications | Two MSSPs and 2 Enhanced USARTs (EUSART) | | |
| 12-Bit Analog-to-Digital Module | 24 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 | | |


© 2009-2018 Microchip Technology Inc.

**FIGURE 1-1: PIC18F6XK22 (64-PIN) BLOCK DIAGRAM**

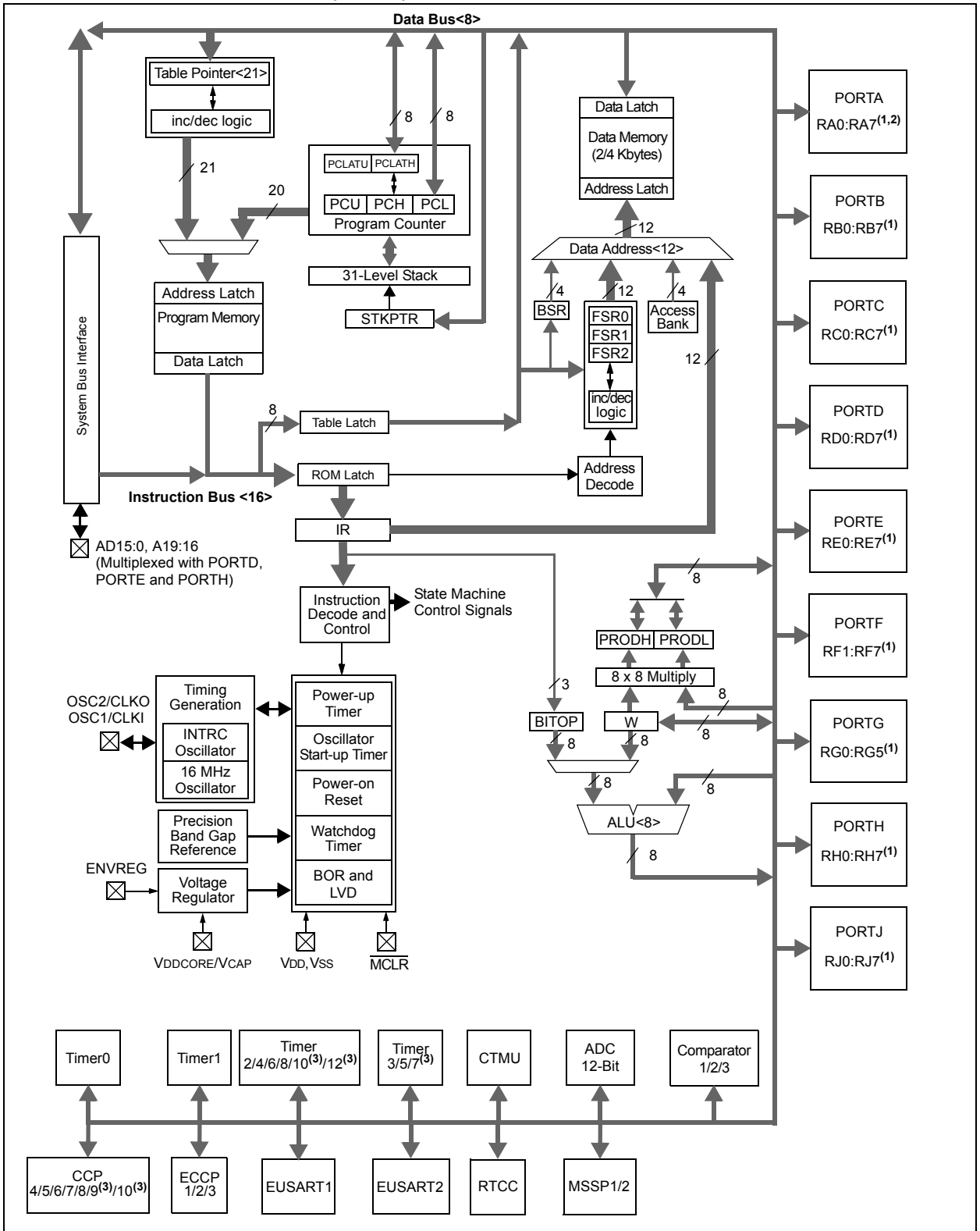



**Note 1:** See Table 1-3 for I/O port pin descriptions.

**2:** RA6 and RA7 are only available as digital I/O in select oscillator modes. For more information, see **Section 3.0 "Oscillator Configurations"**.

**3:** Unimplemented on the PIC18F65K22.

**FIGURE 1-2: PIC18F8XK22 (80-PIN) BLOCK DIAGRAM**




**Note 1:** See Table 1-3 for I/O port pin descriptions.

**2:** RA6 and RA7 are only available as digital I/O in select oscillator modes. See **Section 3.0 "Oscillator Configurations"** for more information.

**3:** Unimplemented on the PIC18F85K22.


© 2009-2018 Microchip Technology Inc.

**TABLE 1-3: PIC18F6XK22 PINOUT I/O DESCRIPTIONS**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | QFN/TQFP | | | |
| MCLR/RG5 | 7 | | | Master Clear (input) or programming voltage (input). |
| MCLR | | I | ST | This pin is an active-low Reset to the device. |
| RG5 | | I | ST | General purpose, input only pin. |
| OSC1/CLKI/RA7 | 39 | | | Oscillator crystal or external clock input. |
| OSC1 | | I | CMOS | Oscillator crystal input. |
| CLKI | | I | CMOS | 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. |
| OSC2/CLKO/RA6 | 40 | | | Oscillator crystal or clock output. |
| OSC2 | | O | — | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| CLKO | | O | — | In certain 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. |

**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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

© 2009-2018 Microchip Technology Inc.

**TABLE 1-3: PIC18F6XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | QFN/TQFP | | | |
| | | | | PORTA is a bidirectional I/O port. |
| RA0/AN0/ULPWU | 24 | | | |
| RA0 | | I/O | TTL | Digital I/O. |
| AN0 | | I | Analog | Analog Input 0. |
| ULPWU | | I | Analog | Ultra Low-Power Wake-up input. |
| 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/T1CKI/T3G/ HLVDIN | 27 | | | |
| RA5 | | I/O | TTL | Digital I/O. |
| AN4 | | I | Analog | Analog Input 4. |
| T1CKI | | I | ST | Timer1 clock input. |
| T3G | | I | ST | Timer3 external clock gate input. |
| HLVDIN | | I | Analog | High/Low-Voltage Detect input. |
| 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
$I^2C$ = $I^2C$/SMBus
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 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-3: PIC18F6XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | QFN/TQFP | | | |
| | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/INT0/FLTO | 48 | | | |
| RB0 | | I/O | TTL | Digital I/O. |
| INT0 | | I | ST | External Interrupt 0. |
| FLT0 | | I | ST | Enhanced PWM Fault input for ECCP1/2/3. |
| RB1/INT1 | 47 | | | |
| RB1 | | I/O | TTL | Digital I/O. |
| INT1 | | I | ST | External Interrupt 1. |
| RB2/INT2/CTED1 | 46 | | | |
| RB2 | | I/O | TTL | Digital I/O. |
| INT2 | | I | ST | External Interrupt 2. |
| CTED1 | | I | ST | CTMU Edge 1 input. |
| RB3/INT3/CTED2/ ECCP2/P2A | 45 | | | |
| RB3 | | I/O | TTL | Digital I/O. |
| INT3 | | I | ST | External Interrupt 3. |
| CTED2 | | I | ST | CTMU Edge 2 input. |
| ECCP2 | | I/O | ST | Capture 2 input/Compare 2 output/PWM2. |
| P2A | | O | — | Enhanced PWM2 Output A. |
| RB4/KBI0 | 44 | | | |
| RB4 | | I/O | TTL | Digital I/O. |
| KBI0 | | I | TTL | Interrupt-on-change pin. |
| RB5/KBI1/T3CKI/T1G | 43 | | | |
| RB5 | | I/O | TTL | Digital I/O. |
| KBI1 | | I | TTL | Interrupt-on-change pin. |
| T3CKI | | I | ST | Timer3 clock input. |
| T1G | | I | ST | Timer1 external clock gate input. |
| 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
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
$I^2C$ = $I^2C$/SMBus
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to $V_{DD}$)

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-3: PIC18F6XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | QFN/TQFP | | | |
| | | | | PORTC is a bidirectional I/O port. |
| RC0/SOSCO/SCLKI | 30 | | | |
| RC0 | | I/O | ST | Digital I/O. |
| SOSCO | | O | — | SOSC oscillator output. |
| SCLKI | | I | ST | Digital SOSC input. |
| RC1/SOSCI/ECCP2/P2A | 29 | | | |
| RC1 | | I/O | ST | Digital I/O. |
| SOSCI | | I | CMOS | SOSC oscillator input. |
| ECCP2(1) | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A | | O | — | Enhanced PWM2 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 | — | Enhanced PWM1 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(4) | | 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(4) | | 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 | — | EUSART asynchronous transmit. |
| CK1 | | I/O | ST | EUSART synchronous clock (see related RX1/DT1). |
| RC7/RX1/DT1 | 32 | | | |
| RC7 | | I/O | ST | Digital I/O. |
| RX1 | | I | ST | EUSART asynchronous receive. |
| DT1 | | I/O | ST | EUSART synchronous data (see related TX1/CK1). |

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

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-3: PIC18F6XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | QFN/TQFP | | | |
| | | | | PORTD is a bidirectional I/O port. |
| RD0/PSP0/CTPLS | 58 | | | |
| RD0 | | I/O | ST | Digital I/O. |
| PSP0 | | I/O | TTL | Parallel Slave Port data. |
| CTPLS | | O | — | CTMU pulse generator output. |
| RD1/PSP1/T5CKI/T7G | 55 | | | |
| RD1 | | I/O | ST | Digital I/O. |
| PSP1 | | I/O | TTL | Parallel Slave Port. |
| T5CKI | | I | ST | Timer5 clock input. |
| T7G | | I | ST | Timer7 external clock gate input. |
| RD2/PSP2 | 54 | | | |
| RD2 | | I/O | ST | Digital I/O. |
| PSP2 | | O | TTL | Parallel Slave Port. |
| RD3/PSP3 | 53 | | | |
| RD3 | | I/O | ST | Digital I/O. |
| PSP3 | | I/O | TTL | Parallel Slave Port. |
| RD4/PSP4/SDO2 | 52 | | | |
| RD4 | | I/O | ST | Digital I/O. |
| PSP4 | | I/O | TTL | Parallel Slave Port. |
| SDO2 | | O | — | SPI data out. |
| RD5/PSP5/SDI2/SDA2 | 51 | | | |
| RD5 | | I/O | ST | Digital I/O. |
| PSP5 | | I/O | TTL | Parallel Slave Port. |
| SDI2 | | I | ST | SPI data in. |
| SDA2 | | I/O | $I^2C$ | $I^2C$ data I/O. |
| RD6/PSP6/SCK2/SCL2 | 50 | | | |
| RD6 | | I/O | ST | Digital I/O. |
| PSP6 | | I/O | TTL | Parallel Slave Port. |
| SCK2 | | I/O | ST | Synchronous serial clock. |
| SCL2(4) | | I/O | $I^2C$ | Synchronous serial clock I/O for $I^2C$ mode. |
| RD7/PSP7/$\overline{SS2}$ | 49 | | | |
| RD7 | | I/O | ST | Digital I/O. |
| PSP7 | | I/O | TTL | Parallel Slave Port. |
| $\overline{SS2}$ | | 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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-3: PIC18F6XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number QFN/TQFP | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | | | | PORTE is a bidirectional I/O port. |
| RE0/$\overline{RD}$/P2D | 2 | | | |
| RE0 | | I/O | ST | Digital I/O. |
| $\overline{RD}$ | | I | TTL | Parallel Slave Port read strobe. |
| P2D | | O | — | EECP2 PWM Output D. |
| RE1/$\overline{WR}$/P2C | 1 | | | |
| RE1 | | I/O | ST | Digital I/O. |
| $\overline{WR}$ | | I | TTL | Parallel Slave Port write strobe. |
| P2C | | O | — | ECCP2 PWM Output C. |
| RE2/$\overline{CS}$/P2B/CCP10 | 64 | | | |
| RE2 | | I/O | ST | Digital I/O. |
| $\overline{CS}$ | | I | TTL | Parallel Slave Port chip select. |
| P2B | | O | — | ECCP2 PWM Output B. |
| CCP10(3) | | I/O | S/T | Capture 10 input/Compare 10 output/PWM10 output. |
| RE3/P3C/CCP9/REFO | 63 | | | |
| RE3 | | I/O | ST | Digital I/O. |
| P3C | | O | — | ECCP3 PWM Output C. |
| CCP9(3,4) | | I/O | S/T | Capture 9 input/Compare 9 output/PWM9 output. |
| REFO | | O | — | Reference clock out. |
| RE4/P3B/CCP8 | 62 | | | |
| RE4 | | I/O | ST | Digital I/O. |
| P3B | | O | — | ECCP3 PWM Output B. |
| CCP8(4) | | I/O | S/T | Capture 8 input/Compare 8 output/PWM8 output. |
| RE5/P1C/CCP7 | 61 | | | |
| RE5 | | I/O | ST | Digital I/O. |
| P1C | | O | — | ECCP1 PWM Output C. |
| CCP7(4) | | I/O | S/T | Capture 7 input/Compare 7 output/PWM7 output. |
| RE6/P1B/CCP6 | 60 | | | |
| RE6 | | I/O | ST | Digital I/O. |
| P1B | | O | — | ECCP1 PWM Output B. |
| CCP6(4 | | I/O | S/T | Capture 6 input/Compare 6 output/PWM6 output. |
| RE7/ECCP2/P2A | 59 | | | |
| RE7 | | I/O | ST | Digital I/O. |
| ECCP2(2) | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A | | 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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-3: PIC18F6XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | QFN/TQFP | | | |
| | | | | PORTF is a bidirectional I/O port. |
| RF1/AN6/C2OUT/CTDIN | 17 | | | |
| RF1 | | I/O | ST | Digital I/O. |
| AN6 | | I | Analog | Analog Input 6. |
| C2OUT | | O | — | Comparator 2 output. |
| CTDIN | | I | ST | CTMU pulse delay input. |
| RF2/AN7/C1OUT | 16 | | | |
| RF2 | | I/O | ST | Digital I/O. |
| AN7 | | I | Analog | Analog Input 7. |
| C1OUT | | O | — | Comparator 1 output. |
| RF3/AN8/C2INB/CTMUI | 15 | | | |
| RF3 | | I/O | ST | Digital I/O. |
| AN8 | | I | Analog | Analog Input 8. |
| C2INB | | I | Analog | Comparator 2 Input B. |
| CTMUI | | O | — | CTMU pulse generator charger for the C2INB comparator input. |
| RF4/AN9/C2INA | 14 | | | |
| RF4 | | I/O | ST | Digital I/O. |
| AN9 | | I | Analog | Analog Input 9. |
| C2INA | | I | Analog | Comparator 2 Input A. |
| RF5/AN10/CVREF/C1INB | 13 | | | |
| RF5 | | I/O | ST | Digital I/O. |
| AN10 | | I | Analog | Analog Input 10. |
| CVREF | | O | Analog | Comparator reference voltage output. |
| C1INB | | I | Analog | Comparator 1 Input B. |
| 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/AN5/$\overline{SS1}$ | 11 | | | |
| RF7 | | I/O | ST | Digital I/O. |
| AN5 | | O | Analog | Analog Input 5. |
| $\overline{SS1}$ | | I | TTL | SPI1 slave select input. |

**Legend:** TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
$I^2C$ = $I^2C$/SMBus
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 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-3: PIC18F6XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | QFN/TQFP | | | |
| | | | | PORTG is a bidirectional I/O port. |
| RG0/ECCP3/P3A | 3 | | | |
| RG0 | | I/O | ST | Digital I/O. |
| ECCP3 | | I/O | ST | Capture 3 input/Compare 3 output/PWM3 output. |
| P3A | | O | — | ECCP3 PWM Output A. |
| RG1/TX2/CK2/AN19/ C3OUT | 4 | | | |
| RG1 | | I/O | ST | Digital I/O. |
| TX2 | | O | — | EUSART asynchronous transmit. |
| CK2 | | I/O | ST | EUSART synchronous clock (see related RX2/DT2). |
| AN19 | | I | Analog | Analog Input 19. |
| C3OUT | | O | — | Comparator 3 output. |
| RG2/RX2/DT2/AN18/ C3INA | 5 | | | |
| RG2 | | I/O | ST | Digital I/O. |
| RX2 | | I | ST | EUSART asynchronous receive. |
| DT2 | | I/O | ST | EUSART synchronous data (see related TX2/CK2). |
| AN18 | | I | Analog | Analog Input 18. |
| C3INA | | I | Analog | Comparator 3 Input A. |
| RG3/CCP4/AN17/P3D/ C3INB | 6 | | | |
| RG3 | | I/O | ST | Digital I/O. |
| CCP4 | | I/O | S/T | Capture 4 input/Compare 4 output/PWM4 output. |
| AN17 | | I | Analog | Analog Input 18. |
| P3D | | O | — | ECCP3 PWM Output D. |
| C3INB | | I | Analog | Comparator 3 Input B. |
| RG4/RTCC/T7CKI/T5G/ CCP5/AN16/P1D/C3INC | 8 | | | |
| RG4 | | I/O | ST | Digital I/O. |
| RTCC | | O | — | RTCC output |
| T7CKI(3) | | I | ST | Timer7 clock input. |
| T5G | | I | ST | Timer5 external clock gate input. |
| CCP5 | | I/O | ST | Capture 5 input/Compare 5 output/PWM5 output. |
| AN16 | | I | Analog | Analog Input 16. |
| P1D | | O | — | ECCP1 PWM Output D. |
| C3INC | | I | Analog | Comparator 3 Input C. |
| RG5 | 7 | | | See the $\overline{\text{MCLR}}$/RG5 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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-3: PIC18F6XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | QFN/TQFP | | | |
| VSS | 9, 25, 41, 56 | P | — | Ground reference for logic and I/O pins. |
| VDD | 26, 38, 57 | P | — | Positive supply for 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<br>VDDCORE<br>VCAP | 10 | P | — | Core logic power or external filter capacitor connection.<br>External filter capacitor connection (regulator enabled/disabled). |

**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); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| $\overline{MCLR}$/RG5 | 9 | | | Master Clear (input) or programming voltage (input). |
| RG5 | | I | ST | This pin is an active-low Reset to the device. |
| $\overline{MCLR}$ | | I | ST | General purpose, input only pin. |
| OSC1/CLKI/RA7 | 49 | | | Oscillator crystal or external clock input. |
| OSC1 | | I | CMOS | Oscillator crystal input. |
| CLKI | | I | CMOS | 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. |
| OSC2/CLKO/RA6 | 50 | | | Oscillator crystal or clock output. |
| OSC2 | | O | — | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. |
| CLKO | | O | — | In certain 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. |

**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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| | | | | PORTA is a bidirectional I/O port. |
| RA0/AN0/ULPWU | 30 | | | |
| RA0 | | I/O | TTL | Digital I/O. |
| AN0 | | I | Analog | Analog Input 0. |
| ULPWU | | I | Analog | Ultra Low-Power Wake-up input. |
| 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/T0CKI | 34 | | | |
| RA4 | | I/O | ST | Digital I/O. |
| T0CKI | | I | ST | Timer0 external clock input. |
| RA5/AN4/T1CKI/ T3G/HLVDIN | 33 | | | |
| RA5 | | I/O | TTL | Digital I/O. |
| AN4 | | I | Analog | Analog Input 4. |
| T1CKI | | I | ST | Timer1 clock input. |
| T3G | | I | ST | Timer3 external clock gate input. |
| HLVDIN | | I | Analog | High/Low-Voltage Detect input. |
| RA6 | | | | See the OSC2/CLKO/RA6 pin. |
| RA7 | | | | See the OSC1/CLKI/RA7 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); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/INT0/FLT0 | 58 | | | |
| RB0 | | I/O | TTL | Digital I/O. |
| INT0 | | I | ST | External Interrupt 0. |
| FLT0 | | I | ST | Enhanced PWM Fault input for ECCP1/2/3. |
| RB1/INT1 | 57 | | | |
| RB1 | | I/O | TTL | Digital I/O. |
| INT1 | | I | ST | External Interrupt 1. |
| RB2/INT2/CTED1 | 56 | | | |
| RB2 | | I/O | TTL | Digital I/O. |
| INT2 | | I | ST | External Interrupt 2. |
| CTED1 | | I | ST | CTMU Edge 1 input. |
| RB3/INT3/CTED2/ ECCP2/P2A | 55 | | | |
| RB3 | | I/O | TTL | Digital I/O. |
| INT3 | | I | ST | External Interrupt 3. |
| CTED2 | | I | ST | CTMU Edge 2 input. |
| ECCP2 | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A | | O | ST | Enhanced PWM2 Output A. |
| RB4/KBI0 | 54 | | | |
| RB4 | | I/O | TTL | Digital I/O. |
| KBI0 | | I | TTL | Interrupt-on-change pin. |
| RB5/KBI1/T3CKI/T1G | 53 | | | |
| RB5 | | I/O | TTL | Digital I/O. |
| KBI1 | | I | TTL | Interrupt-on-change pin. |
| T3CKI | | I | ST | Timer3 clock input. |
| T1G | | I | ST | Timer1 external clock gate input. |
| 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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| | | | | PORTC is a bidirectional I/O port. |
| RC0/SOSCO/SCKLI | 36 | | | |
| RC0 | | I/O | ST | Digital I/O. |
| SOSCO | | O | — | SOSC oscillator output. |
| SCKLI | | I | ST | Digital SOSC input. |
| RC1/SOSCI/ECCP2/P2A | 35 | | | |
| RC1 | | I/O | ST | Digital I/O. |
| SOSCI | | I | CMOS | SOSC oscillator input. |
| ECCP2(1) | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A | | O | — | Enhanced PWM2 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 | — | Enhanced PWM1 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 | — | EUSART asynchronous transmit. |
| CK1 | | I/O | ST | EUSART synchronous clock (see related RX1/DT1). |
| RC7/RX1/DT1 | 38 | | | |
| RC7 | | I/O | ST | Digital I/O. |
| RX1 | | I | ST | EUSART asynchronous receive. |
| DT1 | | I/O | ST | EUSART synchronous data (see related TX1/CK1). |

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

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| | | | | PORTD is a bidirectional I/O port. |
| RD0/PSP0/CTPLS/AD0 | 72 | | | |
| RD0 | | I/O | ST | Digital I/O |
| PSP0(4) | | I/O | TTL | Parallel Slave Port data |
| CTPLS | | O | ST | CTMU pulse generator output |
| AD0 | | I/O | TTL | External Memory Address/Data 0 |
| RD1/T5CKI/T7G/PSP1/AD1 | 69 | | | |
| RD1 | | I/O | ST | Digital I/O |
| T5CKI | | I | ST | Timer5 clock input |
| T7G | | I | ST | Timer7 external clock gate input |
| PSP1(4) | | I/O | TTL | Parallel Slave Port data |
| AD1 | | I/O | TTL | External Memory Address/Data 1 |
| RD2/PSP2/AD2 | 68 | | | |
| RD2 | | I/O | ST | Digital I/O. |
| PSP2(4) | | I/O | TTL | Parallel Slave Port data. |
| AD2 | | I/O | TTL | External Memory Address/Data 2. |
| RD3/PSP3/AD3 | 67 | | | |
| RD3 | | I/O | ST | Digital I/O. |
| PSP3(4) | | I/O | TTL | Parallel Slave Port data. |
| AD3 | | I/O | TTL | External Memory Address/Data 3. |
| RD4/SDO2/PSP4/AD4 | 66 | | | |
| RD4 | | I/O | ST | Digital I/O. |
| SDO2 | | O | — | SPI data out. |
| PSP4(4) | | I/O | TTL | Parallel Slave Port data. |
| AD4 | | I/O | TTL | External Memory Address/Data 4. |
| RD5/SDI2/SDA2/PSP5/AD5 | 65 | | | |
| RD5 | | I/O | ST | Digital I/O. |
| SDI2 | | I | ST | SPI data in. |
| SDA2 | | I/O | $I^2C$ | $I^2C$ data I/O. |
| PSP5(4) | | I/O | TTL | Parallel Slave Port data. |
| AD5 | | I/O | TTL | External Memory Address/Data 5. |

**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); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| RD6/SCK2/SCL2/PSP6/ AD6 | 64 | | | |
| RD6 | | I/O | ST | Digital I/O. |
| SCK2 | | I/O | ST | Synchronous serial clock input/output for SPI mode. |
| SCL2 | | I/O | $I^2C$ | Synchronous serial clock input/output for $I^2C$ mode. |
| PSP6(4) | | I/O | TTL | Parallel Slave Port data. |
| AD6 | | I/O | TTL | External Memory Address/Data 6. |
| RD7/$\overline{SS2}$/PSP7/AD7 | 63 | | | |
| RD7 | | I/O | ST | Digital I/O. |
| $\overline{SS2}$ | | I | TTL | SPI slave select input. |
| PSP7(4) | | I/O | TTL | Parallel Slave Port data. |
| AD7 | | I/O | TTL | External Memory Address/Data 7. |

**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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| | | | | PORTE is a bidirectional I/O port. |
| RE0/P2D/$\overline{RD}$/AD8 | 4 | | | |
| RE0 | | I/O | ST | Digital I/O. |
| P2D | | O | — | ECCP2 PWM Output D. |
| $\overline{RD}$(4) | | I | TTL | Parallel Slave Port read strobe. |
| AD8 | | I/O | TTL | External Memory Address/Data 8. |
| RE1/P2C/$\overline{WR}$/AD9 | 3 | | | |
| RE1 | | I/O | ST | Digital I/O. |
| P2C | | O | — | ECCP2 PWM Output C. |
| $\overline{WR}$(4) | | I | TTL | Parallel Slave Port write strobe. |
| AD9 | | I/O | TTL | External Memory Address/Data 9. |
| RE2/P2B/CCP10/$\overline{CS}$/AD10 | 78 | | | |
| RE2 | | I/O | ST | Digital I/O. |
| P2B | | O | ST | ECCP2 PWM Output B. |
| CCP10(3) | | I/O | ST | Capture 10 input/Compare 10 output/PWM10 output. |
| $\overline{CS}$(4) | | I | TTL | Parallel Slave Port chip select. |
| AD10 | | I/O | TTL | External Memory Address/Data 10. |
| RE3/P3C/CCP9/REFO AD11 | 77 | | | |
| RE3 | | I/O | ST | Digital I/O. |
| P3C | | O | — | ECCP3 PWM Output C. |
| CCP9(3,5) | | I/O | S/T | Capture 9 input/Compare 9 output/PWM9 output. |
| REFO | | O | — | Reference clock out. |
| AD11 | | I/O | TTL | External Memory Address/Data 11. |
| RE4/P3B/CCP8/AD12 | 76 | | | |
| RE4 | | I/O | ST | Digital I/O. |
| P3B | | O | — | ECCP4 PWM Output B. |
| CCP8(5) | | I/O | ST | Capture 8 input/Compare 8 output/PWM8 output. |
| AD12 | | I/O | TTL | External Memory Address/Data 12. |
| RE5/P1C/CCP7/AD13 | 75 | | | |
| RE5 | | I/O | ST | Digital I/O. |
| P1C | | O | — | ECCP1 PWM Output C. |
| CCP7(5) | | I/O | ST | Capture 7 input/Compare 7 output/PWM7 output. |
| AD13 | | I/O | TTL | External Memory Address/Data 13. |

**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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| RE6/P1B/CCP6/AD14 | 74 | | | |
| RE6 | | I/O | ST | Digital I/O. |
| P1B | | O | — | ECCP1 PWM Output B. |
| CCP6(5) | | I/O | ST | Capture 6 input/Compare 6 output/PWM6 output. |
| AD14 | | I/O | ST | External Memory Address/Data 14. |
| RE7/ECCP2/P2A/AD15 | 73 | | | |
| RE7 | | I/O | ST | Digital I/O. |
| ECCP2(2) | | I/O | ST | Capture 2 input/Compare 2 output/PWM2 output. |
| P2A | | O | — | ECCP2 PWM Output A. |
| AD15 | | I/O | ST | External Memory Address/Data 15. |

**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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| | | | | PORTF is a bidirectional I/O port. |
| RF1/AN6/C2OUT/CTDIN | 23 | | | |
| RF1 | | I/O | ST | Digital I/O. |
| AN6 | | I | Analog | Analog Input 6. |
| C2OUT | | O | — | Comparator 2 output. |
| CTDIN | | I | ST | CTMU pulse delay input. |
| RF2/AN7/C1OUT | 18 | | | |
| RF2 | | I/O | ST | Digital I/O. |
| AN7 | | I | Analog | Analog Input 7. |
| C1OUT | | O | — | Comparator 1 output. |
| RF3/AN8/C2INB/CTMUI | 17 | | | |
| RF3 | | I/O | ST | Digital I/O. |
| AN8 | | I | Analog | Analog Input 8. |
| C2INB | | I | Analog | Comparator 2 Input B. |
| CTMUI | | O | — | CTMU pulse generator charger for the C2INB comparator input. |
| RF4/AN9/C2INA | 16 | | | |
| RF4 | | I/O | ST | Digital I/O. |
| AN9 | | I | Analog | Analog Input 9. |
| C2INA | | I | Analog | Comparator 2 Input A. |
| RF5/AN10/C1INB | 15 | | | |
| RF5 | | I/O | ST | Digital I/O. |
| AN10 | | I | Analog | Analog Input 10. |
| C1INB | | I | Analog | Comparator 1 Input B. |
| RF6/AN11/C1INA | 14 | | | |
| RF6 | | I/O | ST | Digital I/O. |
| AN11 | | I | Analog | Analog Input 11. |
| C1INA | | I | Analog | Comparator 1 Input A. |
| RF7/AN5/$\overline{SS1}$ | 13 | | | |
| RF7 | | I/O | ST | Digital I/O. |
| AN5 | | O | Analog | Analog Input 5. |
| $\overline{SS1}$ | | I | ST | 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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| | | | | PORTG is a bidirectional I/O port. |
| RG0/ECCP3/P3A | 5 | | | |
| RG0 | | I/O | ST | Digital I/O. |
| ECCP3 | | I/O | ST | Capture 3 input/Compare 3 output/PWM3 output. |
| P3A | | O | — | ECCP3 PWM Output A. |
| RG1/TX2/CK2/AN19/ C3OUT | 6 | | | |
| RG1 | | I/O | ST | Digital I/O. |
| TX2 | | O | — | EUSART asynchronous transmit. |
| CK2 | | I/O | ST | EUSART synchronous clock (see related RX2/DT2). |
| AN19 | | I | Analog | Analog Input 19. |
| C3OUT | | O | — | Comparator 3 output. |
| RG2/RX2/DT2/AN18/ C3INA | 7 | | | |
| RG2 | | I/O | ST | Digital I/O. |
| RX2 | | I | ST | EUSART asynchronous receive. |
| DT2 | | I/O | ST | EUSART synchronous data (see related TX2/CK2). |
| AN18 | | I | Analog | Analog Input 18. |
| C3INA | | I | Analog | Comparator 3 Input A. |
| RG3/CCP4/AN17/P3D/ C3INB | 8 | | | |
| RG3 | | I/O | ST | Digital I/O. |
| CCP4 | | I/O | ST | Capture 4 input/Compare 4 output/PWM4 output. |
| AN17 | | I | Analog | Analog Input 17. |
| P3D | | O | — | ECCP3 PWM Output D. |
| C3INB | | I | Analog | Comparator 3 Input B. |
| RG4/RTCC/T7CKI/T5G/ CCP5/AN16/P1D/C3INC | 10 | | | |
| RG4 | | I/O | ST | Digital I/O. |
| RTCC | | O | — | RTCC output. |
| T7CKI(3) | | I | ST | Timer7 clock input. |
| T5G | | I | ST | Timer5 external clock gate input. |
| CCP5 | | I/O | ST | Capture 5 input/Compare 5 output/PWM5 output. |
| AN16 | | I | Analog | Analog Input 16. |
| P1D | | O | — | ECCP1 PWM Output D. |
| C3INC | | I | Analog | Comparator 3 Input C. |
| RG5 | 9 | | | See the $\overline{\text{MCLR}}$/RG5 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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| | | | | PORTH is a bidirectional I/O port. |
| RH0/AN23/A16 | 79 | | | |
| RH0 | | I/O | ST | Digital I/O. |
| AN23 | | I | Analog | Analog Input 23. |
| A16 | | I/O | TTL | External Memory Address/Data 16. |
| RH1/AN22/A17 | 80 | | | |
| RH1 | | I/O | ST | Digital I/O. |
| AN22 | | I | Analog | Analog Input 22. |
| A17 | | I/O | TTL | External Address/Data 17. |
| RH2/AN21/A18 | 1 | | | |
| RH2 | | I/O | ST | Digital I/O. |
| AN21 | | I | Analog | Analog Input 21. |
| A18 | | I/O | TTL | External Address/Data 18. |
| RH3/AN20/A19 | 2 | | | |
| RH3 | | I/O | ST | Digital I/O. |
| AN20 | | I | Analog | Analog Input 20. |
| A19 | | I/O | TTL | External Address/Data 19. |
| RH4/CCP9/P3C/AN12/ C2INC | 22 | | | |
| RH4 | | I/O | ST | Digital I/O. |
| CCP9(3,5) | | I/O | ST | Capture 9 input/Compare 9 output/PWM9 output. |
| P3C | | O | — | ECCP3 PWM Output C. |
| AN12 | | I | Analog | Analog Input 12. |
| C2INC | | I | Analog | Comparator 2 Input C. |
| RH5/CCP8/P3B/AN13/ C2IND | 21 | | | |
| RH5 | | I/O | ST | Digital I/O. |
| CCP8(5) | | I/O | ST | Capture 8 input/Compare 8 output/PWM8 output. |
| P3B | | O | — | ECCP3 PWM Output B. |
| AN13 | | I | Analog | Analog Input 13. |
| C2IND | | I | Analog | Comparator 1 Input D. |
| RH6/CCP7/P1C/AN14/ C1INC | 20 | | | |
| RH6 | | I/O | ST | Digital I/O. |
| CCP7(5) | | I/O | ST | Capture 7 input/Compare 7 output/PWM7 output. |
| P1C | | O | — | ECCP1 PWM Output C. |
| AN14 | | I | Analog | Analog Input 14. |
| C1INC | | I | Analog | Comparator 1 Input C. |

**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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| RH7/CCP6/P1B/AN15 | 19 | | | |
| RH7 | | I/O | ST | Digital I/O. |
| CCP6(5) | | I/O | ST | Capture 6 input/Compare 6 output/PWM6 output. |
| P1B | | O | — | ECCP1 PWM Output B. |
| AN15 | | I | Analog | Analog Input 15. |

**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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

**TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS (CONTINUED)**

| Pin Name | Pin Number | Pin Type | Buffer Type | Description |
|---|---|---|---|---|
| | TQFP | | | |
| | | | | 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 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. |
| $V_{SS}$ | 11, 31, 51, 70 | P | — | Ground reference for logic and I/O pins. |
| $V_{DD}$ | 32, 48, 71 | P | — | Positive supply for logic and I/O pins. |
| $AV_{SS}$ | 26 | P | — | Ground reference for analog modules. |
| $AV_{DD}$ | 25 | P | — | Positive supply for analog modules. |
| ENVREG | 24 | I | ST | Enable for on-chip voltage regulator. |
| $V_{DDCORE}/V_{CAP}$ | 12 | | | Core logic power or external filter capacitor connection. |
| $V_{DDCORE}$ | | | | |
| $V_{CAP}$ | | P | — | External filter capacitor connection (regulator enabled/disabled). |

**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 $V_{DD}$); $I^2C$ = $I^2C$/SMBus

**Note 1:** Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

**2:** Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

**3:** Not available on PIC18F65K22 and PIC18F85K22 devices.

**4:** PSP is available only in Microcontroller mode.

**5:** The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).


© 2009-2018 Microchip Technology Inc.

# 2.0 GUIDELINES FOR GETTING STARTED WITH PIC18FXXKXX MICROCONTROLLERS

## 2.1 Basic Connection Requirements

Getting started with the PIC18F87K22 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"**)
- $\overline{\text{MCLR}}$ 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"**)

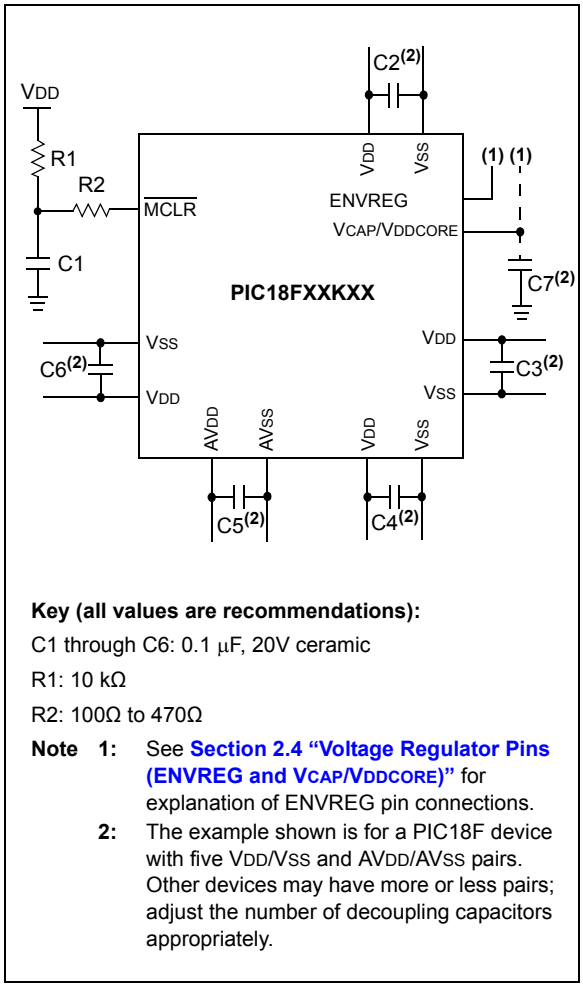
Additionally, the following pins may be required:

- 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

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 VDD, VSS, AVDD and AVSS, 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.

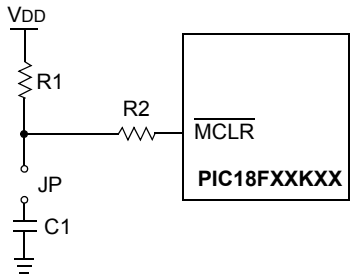
## 2.3 Master Clear ($\overline{\text{MCLR}}$) Pin

The $\overline{\text{MCLR}}$ 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 VDD 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 $\overline{\text{MCLR}}$ pin. Consequently, specific voltage levels (VIH and VIL) 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 $\overline{\text{MCLR}}$ 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 $\overline{\text{MCLR}}$ pin should be placed within 0.25 inch (6 mm) of the pin.

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




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

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


© 2009-2018 Microchip Technology Inc.

## 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 28.3 "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.

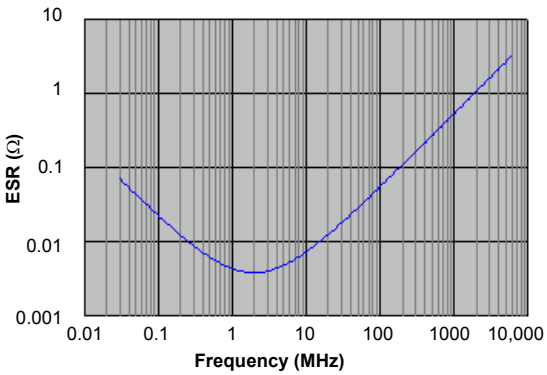
It is recommended that the trace length not exceed 0.25 inch (6 mm). Refer to **Section 31.0 "Electrical Characteristics"** for additional information.

When the regulator is disabled, the VCAP/VDDCORE pin must only be tied to a 0.1 F capacitor. Refer to **Section 28.3 "On-Chip Voltage Regulator"** for information on VDD and VDDCORE.

Some PIC18FXXKXX families, or some devices within a family, do not provide the option of enabling or disabling the on-chip voltage regulator:

- Some devices (with the name, PIC18**LF**XXKXX) permanently disable the voltage regulator.
  These devices do not have the ENVREG pin and require a 0.1 µF capacitor on the VCAP/VDDCORE pin. The VDD level of these devices must comply with the "voltage regulator disabled" specification for Parameter D001, in **Section 31.0 "Electrical Characteristics"**.
- Some devices permanently enable the voltage regulator.
  These devices also do not have the ENVREG pin. The 10 µF capacitor is still required on the VCAP/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 internal 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.

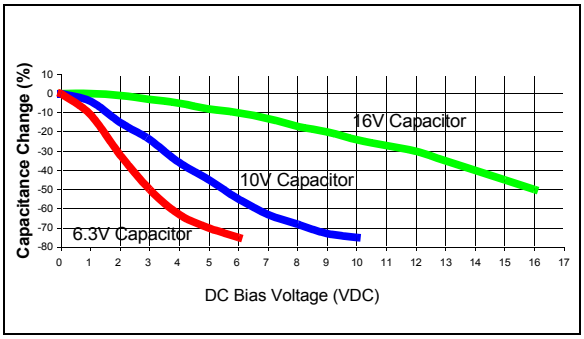
Typical low-cost, 10 μF ceramic capacitors are available in X5R, X7R and Y5V dielectric ratings (other types are also available, but are less common). The initial tolerance specifications for these types of capacitors are often specified as ±10% to ±20% (X5R and X7R), or -20%/+80% (Y5V). However, the 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 internal voltage regulator stability and transient response requirements. Therefore, Y5V capacitors are not recommended for use with the internal 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 internal 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 core 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 ($V_{IH}$) and input low ($V_{IL}$) 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 30.0 "Development Support"**.


© 2009-2018 Microchip Technology Inc.

## 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 2-4. 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).

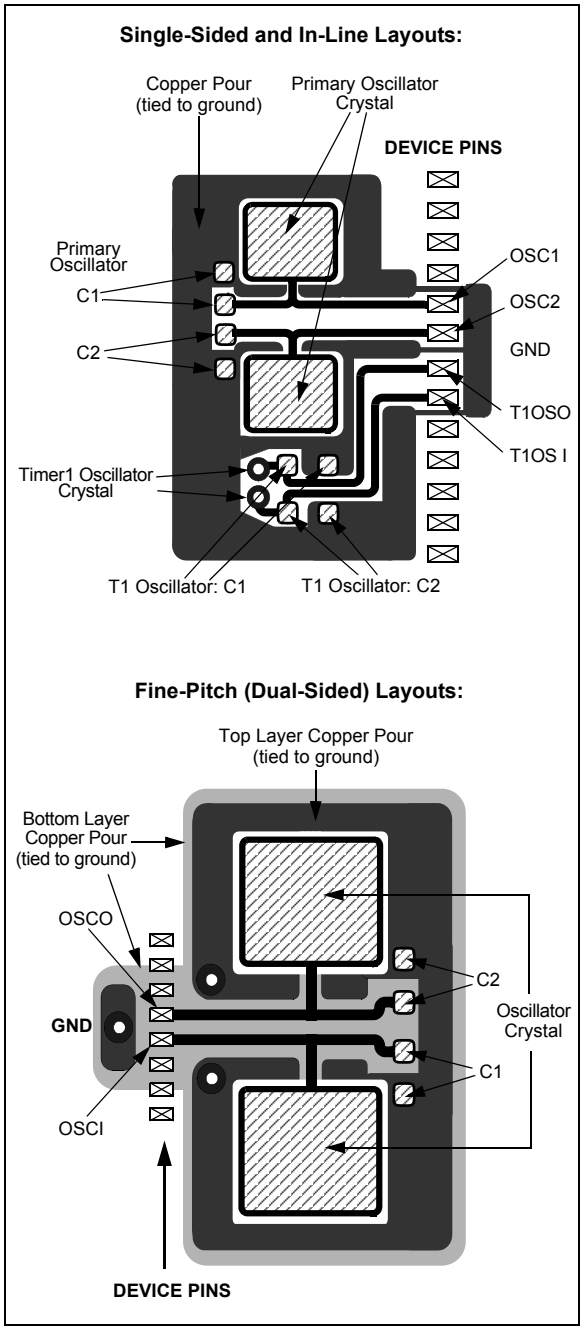
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate web site (www.microchip.com):

- *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**



# 3.0 OSCILLATOR CONFIGURATIONS

## 3.1 Oscillator Types

The PIC18F87K22 family of devices can be operated in the following oscillator modes:

- EC External clock, RA6 available
- ECIO External clock, clock out RA6 (FOSC/4 on RA6)
- HS High-Speed Crystal/Resonator
- XT Crystal/Resonator
- LP Low-Power Crystal
- RC External Resistor/Capacitor, RA6 available
- RCIO External Resistor/Capacitor, clock out RA6 (FOSC/4 on RA6)
- INTIO2 Internal Oscillator with I/O on RA6 and RA7
- INTIO1 Internal Oscillator with FOSC/4 output on RA6 and I/O on RA7

There is also an option for running the 4xPLL on any of the clock sources in the input frequency range of 4 to 16 MHz.

The PLL is enabled by setting the PLLCFG bit (CONFIG1H<4>) or the PLLEN bit (OSCTUNE<6>).

For the EC and HS mode, the PLLEN (software) or PLLCFG (CONFIG) bit can be used to enable the PLL.

For the INTIOx modes (HF-INTOSC):

- Only the PLLEN can enable the PLL (PLLCFG is ignored).
- When the oscillator is configured for the internal oscillator (FOSC<3:0> = 100x), the PLL can be enabled only when the HF-INTOSC frequency is 8 or 16 MHz.

When the RA6 and RA7 pins are not used for an oscillator function or CLKOUT function, they are available as general purpose I/Os.

To optimize power consumption when using EC/HS/XT/LP/RC as the primary oscillator, the frequency input range can be configured to yield an optimized power bias:

- Low-Power Bias – External frequency less than 160 kHz
- Medium Power Bias – External frequency between 160 kHz and 16 MHz
- High-Power Bias – External frequency greater than 16 MHz

All of these modes are selected by the user by programming the FOSC<3:0> Configuration bits (CONFIG1H<3:0>). In addition, PIC18F87K22 family devices can switch between different clock sources, either under software control or, under certain conditions, automatically. This allows for additional power savings by managing device clock speed in real time without resetting the application. The clock sources for the PIC18F87K22 family of devices are shown in Figure 3-1.

For the HS and EC mode, there are additional power modes of operation – depending on the frequency of operation.

HS1 is the Medium Power mode with a frequency range of 4 MHz to 16 MHz. HS2 is the High-Power mode, where the oscillator frequency can go from 16 MHz to 25 MHz. HS1 and HS2 are achieved by setting the CONFIG1H<3:0> correctly. (For details, see Register 28-2 on page 395.)

EC mode has these modes of operation:

- EC1 – For low power with a frequency range up to 160 kHz
- EC2 – Medium power with a frequency range of 160 kHz to 16 MHz
- EC3 – High power with a frequency range of 16 MHz to 64 MHz

EC1, EC2 and EC3 are achieved by setting the CONFIG1H<3:0> correctly. (For details, see Register 28-2 on page 395.)

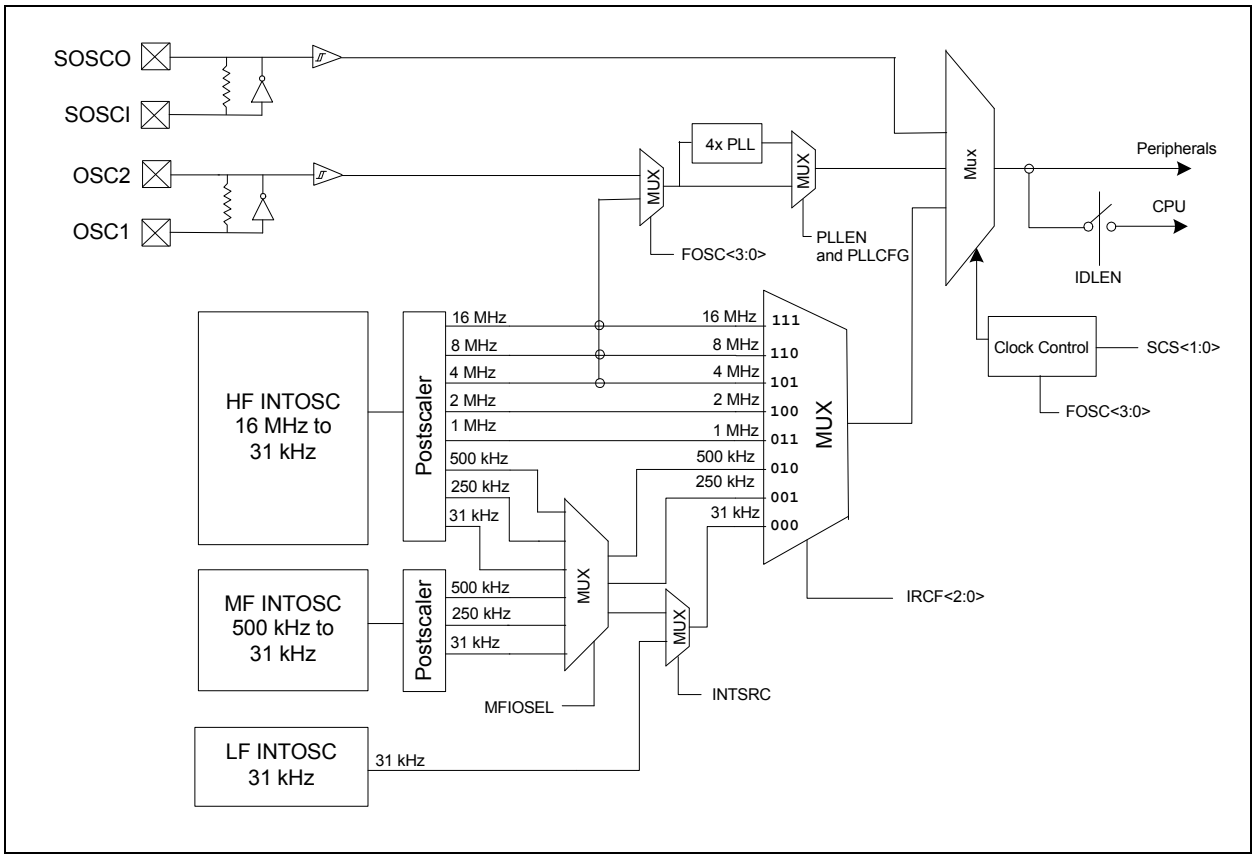
Table 3-1 shows the HS and EC modes' frequency range and FOSC<3:0> settings.


© 2009-2018 Microchip Technology Inc.

**TABLE 3-1: HS, EC, XT, LP AND RC MODES: RANGES AND SETTINGS**

| Mode | Frequency Range | FOSC<3:0> Setting |
|---|---|---|
| EC1 (low power) | DC-160 kHz | 1101 |
| (EC1 & EC1IO) | | 1100 |
| EC2 (medium power) | 160 kHz-16 MHz | 1011 |
| (EC2 & EC2IO) | | 1010 |
| EC3 (high power) | 16 MHz-64 MHz | 0101 |
| (EC3 & EC3IO) | | 0100 |
| HS1 (medium power) | 4 MHz-16 MHz | 0011 |
| HS2 (high power) | 16 MHz-25 MHz | 0010 |
| XT | 100 kHz-4 MHz | 0001 |
| LP | 31.25 kHz | 0000 |
| RC (External) | 0-4 MHz | 001x |
| INTIO | 32 KHz-16 MHz | 100x (and OSCCON, OSCCON2) |

**FIGURE 3-1: PIC18F87K22 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 3-3) controls the tuning and operation of the internal oscillator block. It also implements the PLLEN bit which controls the operation of the Phase Locked Loop (PLL) (see **Section 3.5.3 "PLL Frequency Multiplier"**).

**REGISTER 3-1: OSCCON: OSCILLATOR CONTROL REGISTER(1)**

| R/W-0 | R/W-1 | R/W-1 | R/W-0 | R(1) | R-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IDLEN | IRCF2(2) | IRCF1(2) | IRCF0(2) | OSTS | HFIOFS | SCS1(4) | SCS0(4) |
| 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>:** Internal Oscillator Frequency Select bits(2)
- 111 = HF-INTOSC output frequency is used (16 MHz)
- 110 = HF-INTOSC/2 output frequency is used (8 MHz, default)
- 101 = HF-INTOSC/4 output frequency is used (4 MHz)
- 100 = HF-INTOSC/8 output frequency is used (2 MHz)
- 011 = HF-INTOSC/16 output frequency is used (1 MHz)

If INTSRC = 0 and MFIOSEL = 0:(3,5)
- 010 = HF-INTOSC/32 output frequency is used (500 kHz)
- 001 = HF-INTOSC/64 output frequency is used (250 kHz)
- 000 = LF-INTOSC output frequency is used (31.25 kHz)

If INTSRC = 1 and MFIOSEL = 0:(3,5)
- 010 = HF-INTOSC/32 output frequency is used (500 kHz)
- 001 = HF-INTOSC/64 output frequency is used (250 kHz)
- 000 = HF-INTOSC/512 output frequency is used (31.25 kHz)

If INTSRC = 0 and MFIOSEL = 1:(3,5)
- 010 = MF-INTOSC output frequency is used (500 kHz)
- 001 = MF-INTOSC/2 output frequency is used (250 kHz)
- 000 = LF-INTOSC output frequency is used (31.25 kHz)(6)

If INTSRC = 1 and MFIOSEL = 1:(3,5)
- 010 = MF-INTOSC output frequency is used (500 kHz)
- 001 = MF-INTOSC/2 output frequency is used (250 kHz)
- 000 = MF-INTOSC/16 output frequency is used (31.25 kHz)

bit 3 **OSTS:** Oscillator Start-up Timer Time-out Status bit(1)
- 1 = Oscillator Start-up Timer (OST) time-out has expired; primary oscillator is running, as defined by FOSC<3:0>
- 0 = Oscillator Start-up Timer (OST) time-out is running; primary oscillator is not ready – device is running from internal oscillator (HF-INTOSC, MF-INTOSC or LF-INTOSC)

**Note 1:** The Reset state depends on the state of the IESO Configuration bit (CONFIG1H<7>).
**2:** Modifying these bits will cause an immediate clock frequency switch if the internal oscillator is providing the device clocks.
**3:** Source selected by the INTSRC bit (OSCTUNE<7>).
**4:** Modifying these bits will cause an immediate clock source switch.
**5:** INTSRC = OSCTUNE<7> and MFIOSEL = OSCCON2<0>.
**6:** Lowest power option for an internal source.


© 2009-2018 Microchip Technology Inc.

**REGISTER 3-1: OSCCON: OSCILLATOR CONTROL REGISTER(1) (CONTINUED)**

bit 2 **HFIOFS:** INTOSC Frequency Stable bit
1 = HF-INTOSC oscillator frequency is stable
0 = HF-INTOSC oscillator frequency is not stable

bit 1-0 **SCS<1:0>:** System Clock Select bits(4)
1x = Internal oscillator block (LF-INTOSC, MF-INTOSC or HF-INTOSC)
01 = SOSC oscillator
00 = Default primary oscillator (OSC1/OSC2 or HF-INTOSC with or without PLL; defined by the FOSC<3:0> Configuration bits, CONFIG1H<3:0>.)

**Note 1:** The Reset state depends on the state of the IESO Configuration bit (CONFIG1H<7>).
**2:** Modifying these bits will cause an immediate clock frequency switch if the internal oscillator is providing the device clocks.
**3:** Source selected by the INTSRC bit (OSCTUNE<7>).
**4:** Modifying these bits will cause an immediate clock source switch.
**5:** INTSRC = OSCTUNE<7> and MFIOSEL = OSCCON2<0>.
**6:** Lowest power option for an internal source.

**REGISTER 3-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2**

| U-0 | R-0 | U-0 | U-0 | R/W-0 | U-0 | R-x | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | SOSCRUN | — | — | SOSCGO | — | MFIOFS | MFIOSEL |
| 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 **SOSCRUN:** SOSC Run Status bit
1 = System clock comes from a secondary SOSC
0 = System clock comes from an oscillator other than SOSC

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

bit 3 **SOSCGO:** Oscillator Start Control bit
1 = Oscillator is running, even if no other sources are requesting it
0 = Oscillator is shut off if no other sources are requesting it (When the SOSC is selected to run from a digital clock input, rather than an external crystal, this bit has no effect.)

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **MFIOFS:** MF-INTOSC Frequency Stable bit
1 = MF-INTOSC is stable
0 = MF-INTOSC is not stable

bit 0 **MFIOSEL:** MF-INTOSC Select bit
1 = MF-INTOSC is used in place of HF-INTOSC frequencies of 500 kHz, 250 kHz and 31.25 kHz
0 = MF-INTOSC is not used

**REGISTER 3-3: 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 16 MHz INTOSC source (divide-by-512 enabled, HF-INTOSC)
0 = 31 kHz device clock derived from INTRC 31 kHz oscillator (LF-INTOSC)

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


© 2009-2018 Microchip Technology Inc.

## 3.3 Clock Sources and Oscillator Switching

Essentially, PIC18F87K22 family devices have these 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<3:0> Configuration bits (CONFIG1H<3:0>), the internal oscillator block may be considered a primary oscillator. The internal oscillator block can be one of the following:

- 31 kHz LF-INTRC source
- 31 kHz to 500 kHz MF-INTOSC source
- 31 kHz to 16 MHz HF-INTOSC source

The particular mode is defined by the FOSC Configuration bits. The details of these modes are covered in **Section 3.5 "External Oscillator Modes"**.

The **secondary oscillators** are external clock sources that are not connected to the OSC1 or OSC2 pin. These sources may continue to operate, even after the controller is placed in a power-managed mode. PIC18F87K22 family devices offer the SOSC (Timer1/3/5/7) 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 SOSC can be enabled from any peripheral that requests it. There are eight ways the SOSC can be enabled: if the SOSC is selected as the source by any of the odd timers, which is done by each respective SOSCEN bit (TxCON<3>), if the SOSC is selected as the RTCC source by the RTCOSC Configuration bit (CONFIG3L<1>), if the SOSC is selected as the CPU clock source by the SCS bits (OSCCON<1:0>) or if the SOSCGO bit is set (OSCCON2<3>). The SOSCGO bit is used to warm up the SOSC so that it is ready before any peripheral requests it.

The secondary oscillator has three Run modes. The SOSCSEL<1:0> bits (CONFIG1L<4:3>) decide the SOSC mode of operation:

- `11` = High-power SOSC circuit
- `10` = Digital (SCLKI) mode
- `01` = Low-power SOSC circuit

If a secondary oscillator is not desired and digital I/O on port pins, RC0 and RC1, is needed, the SOSCSEL bits must be set to Digital mode.

In addition to being a primary clock source in some circumstances, the **internal oscillator** is available as a power-managed mode clock source. The LF-INTOSC 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 3.6 "Internal Oscillator Block"**.

The PIC18F87K22 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 OSC1/OSC2 OSCILLATOR

The OSC1/OSC2 oscillator block is used to provide the oscillator modes and frequency ranges:

| Mode | Design Operating Frequency |
|---|---|
| LP | 31.25-100 kHz |
| XT | 100 kHz to 4 MHz |
| HS | 4 MHz to 25 MHz |
| EC | 0 to 64 MHz (external clock) |
| EXTRC | 0 to 4 MHz (external RC) |

The crystal-based oscillators (XT, HS and LP) have a built-in start-up time. The operation of the EC and EXTRC clocks is immediate.

### 3.3.2 CLOCK SOURCE SELECTION

The System Clock Select bits, SCS<1:0> (OSCCON2<1:0>), select the clock source. The available clock sources are the primary clock defined by the FOSC<3:0> Configuration bits, the secondary clock (SOSC oscillator) and the internal oscillator. The clock source changes after one or more of the bits is written to, following a brief clock transition interval.

The OSTS (OSCCON<3>) and SOSCRUN (OSCCON<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 SOSCRUN bit indicates when the SOSC oscillator (from Timer1/3/5/7) 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 (OSCCON<7>) determines if the device goes into Sleep mode or one of the Idle modes when the `SLEEP` instruction is executed.

© 2009-2018 Microchip Technology Inc.

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 Timer1/3/5/7 oscillator must be enabled to select the secondary clock source. The Timerx oscillator is enabled by setting the SOSCEN bit in the Timerx Control register (TxCON<3>). If the Timerx 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 Timerx oscillator be operating and stable before executing the SLEEP instruction or a very long delay may occur while the Timerx oscillator starts. |

#### 3.3.2.1 System Clock Selection and Device Resets

Since the SCS bits are cleared on all forms of Reset, this means the primary oscillator, defined by the FOSC<3:0> 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 source (HS, EC, XT, LP, External RC and PLL-Enabled modes).

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 8 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 FOSC Configuration bits are read and the oscillator selection of the operational mode is made.

Note that either the primary clock source or the internal oscillator will have two bit setting options for the possible values of the SCS<1:0> bits, at any given time.

### 3.3.3 OSCILLATOR TRANSITIONS

PIC18F87K22 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 RC Oscillator

For timing-insensitive applications, the RC and RCIO Oscillator modes offer additional cost savings. The actual oscillator frequency is a function of several factors:

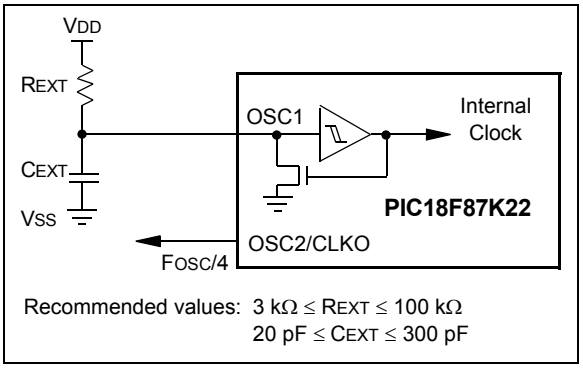
- Supply Voltage
- Values of the External Resistor (REXT) and Capacitor (CEXT)
- Operating Temperature

Given the same device, operating voltage and temperature, and component values, there will also be unit to unit frequency variations. These are due to factors, such as:

- Normal manufacturing variation
- Difference in lead frame capacitance between package types (especially for low CEXT values)
- Variations within the tolerance of limits of REXT and CEXT

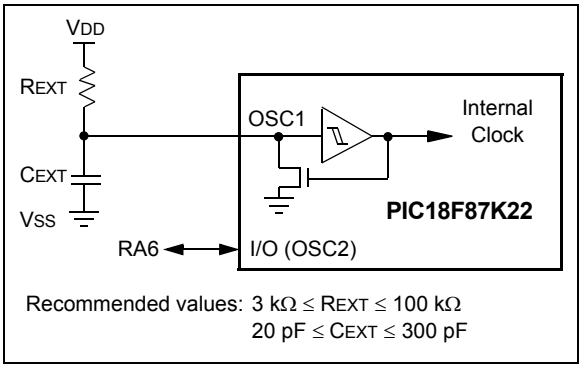
In the RC Oscillator mode, the oscillator frequency, divided by 4, is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 3-2 shows how the R/C combination is connected.

**FIGURE 3-2: RC OSCILLATOR MODE**




The RCIO Oscillator mode (Figure 3-3) functions like the RC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

**FIGURE 3-3: RCIO OSCILLATOR MODE**





© 2009-2018 Microchip Technology Inc.

## 3.5 External Oscillator Modes

### 3.5.1 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 3-4 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 3-2: 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 3-3 for additional information.

**TABLE 3-3: 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 |

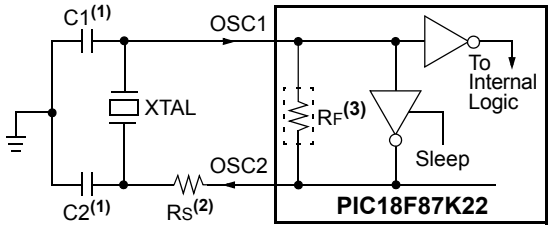
**Capacitor values are for design guidance only.**

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 3-2 for oscillator-specific information. Also see the notes following this table for additional information.

| | |
|---|---|
| **Note 1:** | Higher capacitance increases the stability of the 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 a low drive level specification. |
| **4:** | Always verify oscillator performance over the VDD and temperature range that is expected for the application. |

**FIGURE 3-4: CRYSTAL/CERAMIC RESONATOR OPERATION (HS OR HSPLL CONFIGURATION)**



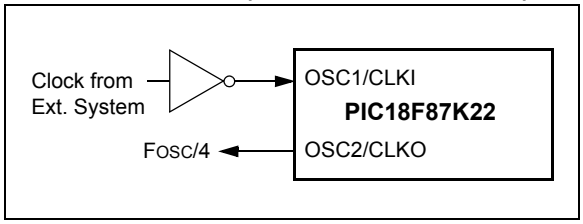

| | |
|---|---|
| **Note 1:** | See Table 3-2 and Table 3-3 for initial values of C1 and C2. |
| **2:** | A series resistor (RS) may be required for AT strip cut crystals. |
| **3:** | RF varies with the oscillator mode chosen. |

### 3.5.2 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.

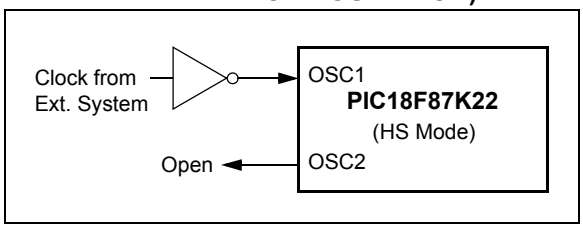
In the EC Oscillator mode, the oscillator frequency, divided by 4, is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 3-5 shows the pin connections for the EC Oscillator mode.

**FIGURE 3-5: EXTERNAL CLOCK INPUT OPERATION (EC CONFIGURATION)**




An external clock source may also be connected to the OSC1 pin in HS mode, as shown in Figure 3-6. 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 3-6: EXTERNAL CLOCK INPUT OPERATION (HS OSC CONFIGURATION)**




### 3.5.3 PLL FREQUENCY MULTIPLIER

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

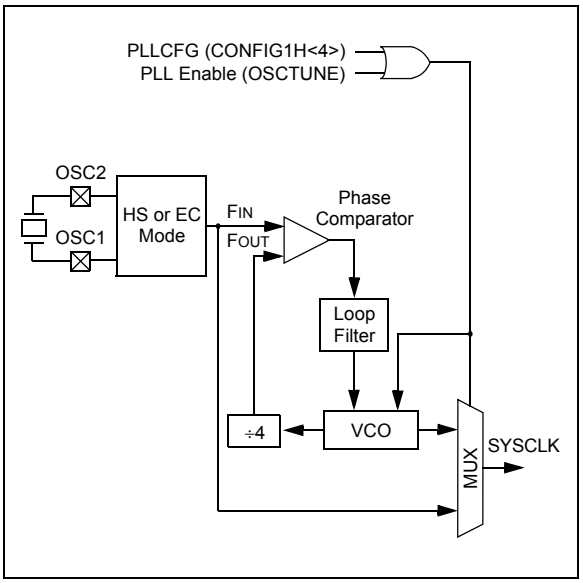
#### 3.5.3.1 HSPLL and ECPLL Modes

The HSPLL and ECPLL modes provide the ability to selectively run the device at four times the external oscillating source to produce frequencies up to 64 MHz.

The PLL is enabled by setting the PLLEN bit (OSCTUNE<6>) or the PLLCFG bit (CONFIG1H<4>). The PLLEN bit provides a software control for the PLL, even if PLLCFG is set to '0'. The PLL is enabled only when the HS or EC oscillator frequency is within the 4 MHz to16 MHz input range.

This enables additional flexibility for controlling the application's clock speed in software. The PLLEN should be enabled in HS or EC Oscillator mode only if the input frequency is in the range of 4 MHz-16 MHz.

**FIGURE 3-7: PLL BLOCK DIAGRAM**




#### 3.5.3.2 PLL and HF-INTOSC

The PLL is 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 64 MHz.

The operation of INTOSC with the PLL is described in **Section 3.6.2 "INTPLL Modes"**. Care should be taken that the PLL is enabled only if the HF-INTOSC postscaler is configured for 8 MHz or 16 MHz.


© 2009-2018 Microchip Technology Inc.

## 3.6 Internal Oscillator Block

The PIC18F87K22 family of devices includes an internal oscillator block which generates two different clock signals. Either clock can be used as the microcontroller's clock source, which may eliminate the need for an external oscillator circuit on the OSC1 and/or OSC2 pins.

The internal oscillator consists of three blocks, depending on the frequency of operation. They are HF-INTOSC, MF-INTOSC and LF-INTRC.

In HF-INTOSC mode, the internal oscillator can provide a frequency ranging from 31 kHz to 16 MHz, with the postscaler deciding the selected frequency (IRCF<2:0>).

The INTSRC bit (OSCTUNE<7>) and MFIOSEL bit (OSCCON2<0>) also decide which INTOSC provides the lower frequency (500 kHz to 31 KHz). For the HF-INTOSC to provide these frequencies, INTSRC = 1 and MFIOSEL = 0.

In HF-INTOSC, the postscaler (IRCF<2:0>) provides the frequency range of 31 kHz to 16 MHz. If HF-INTOSC is used with the PLL, the input frequency to the PLL should be 8 MHz or 16 MHz (IRCF<2:0> = 111 or 110).

For MF-INTOSC mode to provide a frequency range of 500 kHz to 31 kHz, INTSRC = 1 and MFIOSEL = 1. The postscaler (IRCF<2:0>), in this mode, provides the frequency range of 31 kHz to 500 kHz.

The LF-INTRC can provide only 31 kHz if INTSRC = 0.

The LF-INTRC provides 31 kHz and is enabled if it is selected as the device clock source. The mode is 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 28.0 "Special Features of the CPU"**.

The clock source frequency (HF-INTOSC, MF-INTOSC or LF-INTRC direct) is selected by configuring the IRCF bits of the OSCCON register, as well the INTSRC and MFIOSEL bits. The default frequency on device Resets is 8 MHz.

### 3.6.1 INTIO MODES

Using the internal oscillator as the clock source eliminates the need for up to two external oscillator pins, which can then be used for digital I/O. Two distinct oscillator configurations, which are determined by the FOSC Configuration bits, are available:

- In INTIO1 mode, the OSC2 pin (RA6) outputs FOSC/4, while OSC1 functions as RA7 (see Figure 3-8) for digital input and output.
- In INTIO2 mode, OSC1 functions as RA7 and OSC2 functions as RA6 (see Figure 3-9). Both are available as digital input and output ports.

**FIGURE 3-8: INTIO1 OSCILLATOR MODE**

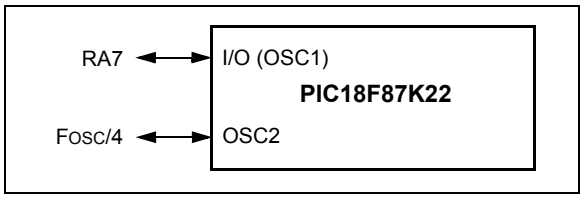
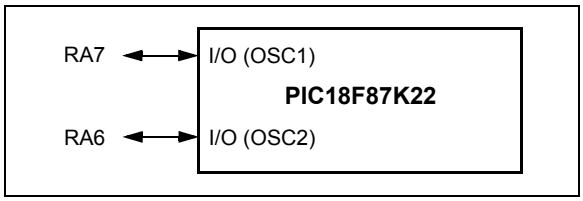



**FIGURE 3-9: INTIO2 OSCILLATOR MODE**




### 3.6.2 INTPLL MODES

The 4x Phase Lock Loop (PLL) can be used with the HF-INTOSC to produce faster device clock speeds than are normally possible with the internal oscillator sources. When enabled, the PLL produces a clock speed of 32 MHz or 64 MHz.

PLL operation is controlled through software. The control bit, PLLEN (OSCTUNE<6>), is used to enable or disable its operation. Additionally, the PLL will only function when the selected HF-INTOSC frequency is either 8 MHz or 16 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 3-8).
- 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 3-9).

### 3.6.3 INTERNAL OSCILLATOR OUTPUT FREQUENCY AND TUNING

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 16 MHz. It can be adjusted in the user's application by writing to TUN<5:0> (OSCTUNE<5:0>) in the OSCTUNE register (Register 3-3).

When the OSCTUNE register is modified, the INTOSC (HF-INTOSC and MF-INTOSC) frequency will begin shifting to the new frequency. The oscillator will require some time to stabilize. Code execution continues during this shift and there is no indication that the shift has occurred.

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

### 3.6.4 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 LF-INTOSC 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.

#### 3.6.4.1 Compensating with the EUSART

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

#### 3.6.4.2 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 SOSC oscillator.

Both timers are cleared, but the timer clocked by the reference source 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.

#### 3.6.4.3 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.

### 3.6.5 LFINTOSC OPERATION IN SLEEP

When the Watchdog Timer (WDT) or Real-Time Clock and Calendar (RTCC) modules are enabled and configured to use the LFINTOSC, the LFINTOSC will continue to run when the device is in Sleep, unlike other internal clock sources.

While in Sleep, the LFINTOSC has two power modes, a High-Power and a Low-Power mode, controlled by the INTOSCSEL bit in the CONFIG1L Configuration Word. The High-Power mode is the same as the LFINTOSC while the part is awake and conforms to the specifications outlined for that oscillator. The Low-Power mode consumes less current, but has a much lower accuracy and is not recommended for timing-sensitive applications.

## 3.7 Reference Clock Output

In addition to the FOSC/4 clock output, in certain oscillator modes, the device clock in the PIC18F87K22 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 submultiples to drive external devices in the application.

This reference clock output is controlled by the REFOCON register (Register 3-4). 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.


© 2009-2018 Microchip Technology Inc.

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. If not, 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.

**REGISTER 3-4: 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:** For ROSEL (REVOCON<4>), the primary oscillator is available only when configured as the default via the FOSC settings. This is regardless of whether the device is in Sleep mode.


© 2009-2018 Microchip Technology Inc.

## 3.8 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 SOSC oscillator is operating and providing the device clock. The SOSC oscillator may also run in all power-managed modes if required to clock SOSC.

In RC_RUN and RC_IDLE modes, the internal oscillator provides the device clock source. The 31 kHz LF-INTOSC 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 28.2 "Watchdog Timer (WDT)"** through **Section 28.5 "Fail-Safe Clock Monitor"** for more information on WDT, Fail-Safe Clock Monitor and Two-Speed Start-up).

If 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 INTOSC is required to support WDT operation. The SOSC 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, INTx pins and others). Peripherals that may add significant current consumption are listed in **Section 31.2 "DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/ Extended)"**.

## 3.9 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 a power-up time of about 1 ms (Parameter 33, Table 31-14).

The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable (HS, XT or LP modes). The OST does this by counting 1,024 oscillator cycles before allowing the oscillator to clock the device.

There is a delay of interval, $T_{CSD}$ (Parameter 38, Table 31-14), following POR, while the controller becomes ready to execute instructions.

**TABLE 3-4: 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 is disabled at quiescent voltage level | Feedback inverter is disabled at quiescent voltage level |
| INTOSC, INTPLL1/2 | I/O pin, RA6, direction is controlled by TRISA<6> | I/O pin, RA6, direction is controlled by TRISA<7> |

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

## 4.0 POWER-MANAGED MODES

The PIC18F87K22 family of devices offers a total of seven operating modes for more efficient power management. These modes provide a variety of options for selective power conservation in applications where resources may be limited (such as battery-powered devices).

There are three categories of power-managed mode:

- Run modes
- Idle modes
- Sleep mode

There is an Ultra Low-Power Wake-up (ULPWU) for waking from the Sleep mode.

These categories define which portions of the device are clocked, and sometimes, 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 ULPWU mode, on the RA0 pin, enables a slow falling voltage to generate a wake-up, even from Sleep, without excess current consumption. (See **Section 4.7 "Ultra Low-Power Wake-up"**.)

The power-managed modes include several power-saving features offered on previous PIC® devices. One is the clock switching feature, offered in other PIC18 devices. This feature allows the controller to use the SOSC oscillator instead of the primary one. Another power-saving feature is Sleep mode, offered by all PIC devices, where all device clocks are stopped.

### 4.1 Selecting Power-Managed Modes

Selecting a power-managed mode requires two decisions:

- Will the CPU be clocked or not
- What will be the clock source

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 select one of three clock sources for power-managed modes. Those sources are:

- The primary clock as defined by the FOSC<3:0> Configuration bits
- The secondary clock (the SOSC oscillator)
- The internal oscillator block (for LF-INTOSC modes)

#### 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 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 considerations are discussed in **Section 4.1.3 "Clock Transitions and Status Indicators"** and subsequent sections.

Entering 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 and impending mode, a change to a power-managed mode does not always require setting all of the previously discussed bits. Many transitions can 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 as desired, it may only be necessary to perform a `SLEEP` instruction to switch to the desired mode.

**TABLE 4-1: POWER-MANAGED MODES**

| Mode | OSCCON Bits | | Module Clocking | | Available Clock and Oscillator Source |
|---|---|---|---|---|---|
| | IDLEN<7>(1) | SCS<1:0> | CPU | Peripherals | |
| Sleep | 0 | N/A | Off | Off | None – All clocks are disabled |
| PRI_RUN | N/A | 00 | Clocked | Clocked | Primary – XT, LP, HS, EC, RC and PLL modes. This is the normal, Full-Power Execution mode. |
| SEC_RUN | N/A | 01 | Clocked | Clocked | Secondary – SOSC Oscillator |
| RC_RUN | N/A | 1x | Clocked | Clocked | Internal oscillator block(2) |
| PRI_IDLE | 1 | 00 | Off | Clocked | Primary – LP, XT, HS, RC, EC |
| SEC_IDLE | 1 | 01 | Off | Clocked | Secondary – SOSC oscillator |
| RC_IDLE | 1 | 1x | Off | Clocked | Internal oscillator block(2) |

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

**2:** Includes INTOSC (HF-INTOSC and MG-INTOSC) and INTOSC postscaler, as well as the LF-INTOSC source.


© 2009-2018 Microchip Technology Inc.

### 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. The HF-INTOSC and MF-INTOSC are termed as INTOSC in this chapter.

Three bits indicate the current clock source and its status, as shown in Table 4-2. The three bits are:

- OSTS (OSCCON<3>)
- HFIOFS (OSCCON<2>)
- SOSCRUN (OSCCON2<6>)

**TABLE 4-2: SYSTEM CLOCK INDICATOR**

| Main Clock Source | OSTS | HFIOFS or MFIOFS | SOSCRUN |
|---|---|---|---|
| Primary Oscillator | 1 | 0 | 0 |
| INTOSC (HF-INTOSC or MF-INTOSC) | 0 | 1 | 0 |
| Secondary Oscillator | 0 | 0 | 1 |
| MF-INTOSC or HF-INTOSC as Primary Clock Source | 1 | 1 | 0 |
| LF-INTOSC is Running or INTOSC is Not Yet Stable | 0 | 0 | 0 |

When the OSTS bit is set, the primary clock is providing the device clock. When the HFIOFS or MFIOFS bit is set, the INTOSC output is providing a stable 16 MHz clock source to a divider that actually drives the device clock. When the SOSCRUN bit is set, the SOSC oscillator is providing the clock. If none of these bits are set, either the LF-INTOSC clock source is clocking the device or the INTOSC source is not yet stable.

If the internal oscillator block is configured as the primary clock source by the FOSC<3:0> Configuration bits (CONFIG1H<3:0>), then the OSTS and HFIOFS or MFIOFS bits can be set when in PRI_RUN or PRI_IDLE modes. This indicates that the primary clock (INTOSC output) is generating a stable 16 MHz output. Entering another INTOSC power-managed mode at the same frequency would clear the OSTS bit.

**Note 1:** Caution should be used when modifying a single IRCF bit. At a lower VDD, it is possible to select a higher clock speed than is supportable by that VDD. Improper device operation may result if the VDD/FOSC specifications are violated.

**2:** 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. (For details, see **Section 28.4 "Two-Speed Start-up"**.) In this mode, the OSTS bit is set. The HFIOFS or MFIOFS bit may be set if the internal oscillator block is the primary clock source. (See **Section 3.2 "Control Registers"**.)

### 4.2.2 SEC_RUN MODE

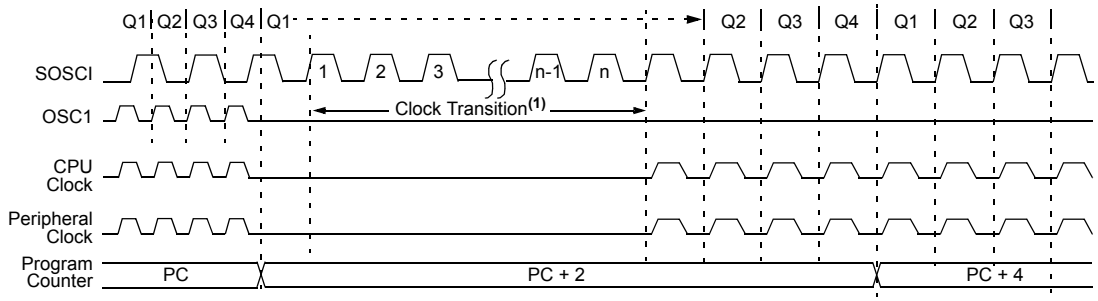
The SEC_RUN mode is the compatible mode to the "clock-switching" feature offered in other PIC18 devices. In this mode, the CPU and peripherals are clocked from the SOSC oscillator. This enables lower power consumption while retaining a high-accuracy clock source.

SEC_RUN mode is entered by setting the SCS<1:0> bits to '01'. The device clock source is switched to the SOSC oscillator (see Figure 4-1), the primary oscillator is shut down, the SOSCRUN bit (OSCCON2<6>) is set and the OSTS bit is cleared.

**Note:** The SOSC oscillator can be enabled by setting the SOSCGO bit (OSCCON2<3>). If this bit is set, the clock switch to the SEC_RUN mode can switch immediately once SCS<1:0> are set to '01'.

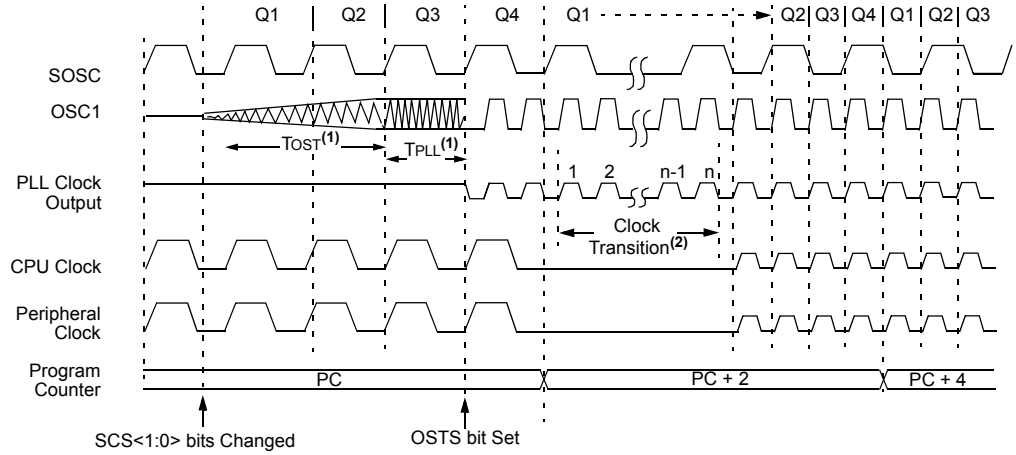
On transitions from SEC_RUN mode to PRI_RUN mode, the peripherals and CPU continue to be clocked from the SOSC oscillator while the primary clock is started. When the primary clock becomes ready, a clock switch back to the primary clock occurs (see Figure 4-2). When the clock switch is complete, the SOSCRUN bit is cleared, the OSTS bit is set and the primary clock is providing the clock. The IDLEN and SCS bits are not affected by the wake-up and the SOSC oscillator continues to run.

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




**Note 1:** Clock transition typically occurs within 2-4 TOSC.

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




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

**2:** Clock transition typically occurs within 2-4 TOSC.


© 2009-2018 Microchip Technology Inc.

### 4.2.3 RC_RUN MODE

In RC_RUN mode, the CPU and peripherals are clocked from the internal oscillator block using the INTOSC multiplexer. In this mode, the primary clock is shut down. When using the LF-INTOSC source, 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.

If the primary clock source is the internal oscillator block – either LF-INTOSC or INTOSC (MF-INTOSC or HF-INTOSC) – there are no distinguishable differences between the PRI_RUN and RC_RUN modes during execution. Entering or exiting RC_RUN mode, however, causes a clock switch delay. Therefore, if the primary clock source is the internal oscillator block, using RC_RUN mode is not recommended.

This mode is entered by setting the SCS1 bit to '1'. To maintain software compatibility with future devices, it is recommended that the SCS0 bit also be cleared, even though the bit is ignored. When the clock source is switched to the INTOSC multiplexer (see Figure 4-3), the primary oscillator is shut down and the OSTS bit is cleared. The IRCF bits may be modified at any time to immediately change the clock speed.

| | |
|---|---|
| **Note:** | Caution should be used when modifying a single IRCF bit. At a lower VDD, it is possible to select a higher clock speed than is supportable by that VDD. Improper device operation may result if the VDD/FOSC specifications are violated. |

If the IRCF bits and the INTSRC bit are all clear, the INTOSC output (HF-INTOSC/MF-INTOSC) is not enabled and the HFIOFS and MFIOFS bits will remain clear. There will be no indication of the current clock source. The LF-INTOSC source is providing the device clocks.

If the IRCF bits are changed from all clear (thus, enabling the INTOSC output), or if INTSRC or MFIOSEL is set, the HFIOFS or MFIOFS bit is set after the INTOSC output becomes stable. For details, see Table 4-3.

**TABLE 4-3: INTERNAL OSCILLATOR FREQUENCY STABILITY BITS**

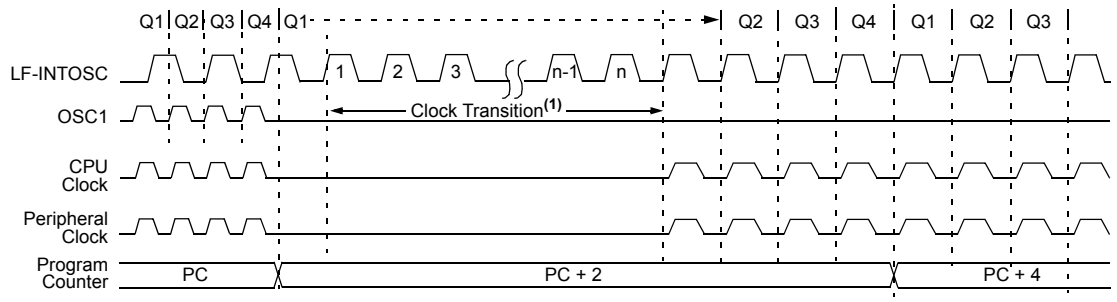
| IRCF<2:0> | INTSRC | MFIOSEL | Status of MFIOFS or HFIOFS when INTOSC is Stable |
|---|---|---|---|
| 000 | 0 | x | MFIOFS = 0, HFIOFS = 0 and clock source is LF-INTOSC |
| 000 | 1 | 0 | MFIOFS = 0, HFIOFS = 1 and clock source is HF-INTOSC |
| 000 | 1 | 1 | MFIOFS = 1, HFIOFS = 0 and clock source is MF-INTOSC |
| Non-Zero | x | 0 | MFIOFS = 0, HFIOFS = 1 and clock source is HF-INTOSC |
| Non-Zero | x | 1 | MFIOFS = 1, HFIOFS = 0 and clock source is MF-INTOSC |

Clocks to the device continue while the INTOSC source stabilizes after an interval of TIOBST (Parameter 39, Table 31-14).

If the IRCF bits were previously at a non-zero value, or if INTSRC was set before setting SCS1 and the INTOSC source was already stable, the HFIOFS or MFIOFS bit will remain set.

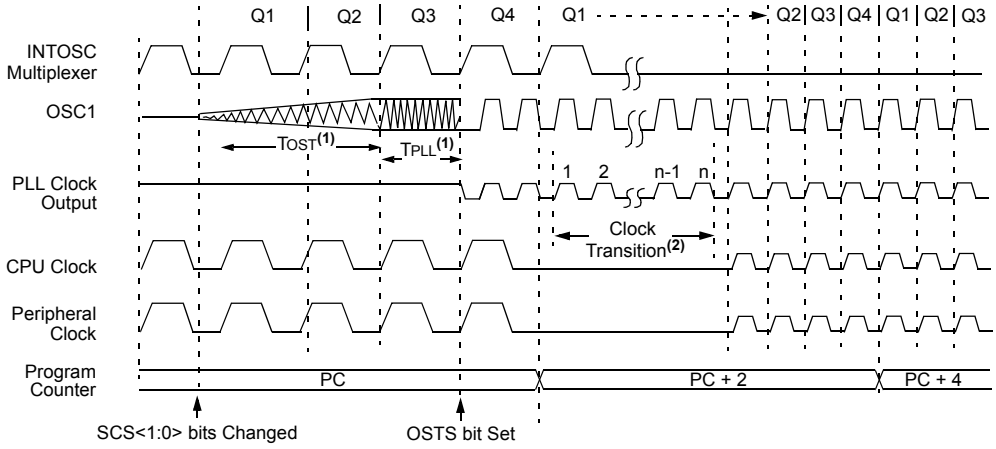
On transitions from RC_RUN mode to PRI_RUN mode, the device continues to be clocked from the INTOSC multiplexer while the primary clock is started. When the primary clock becomes ready, a clock switch to the primary clock occurs (see Figure 4-4). When the clock switch is complete, the HFIOFS or MFIOFS bit is cleared, the OSTS bit is set and the primary clock is providing the device clock. The IDLEN and SCS bits are not affected by the switch. The LF-INTOSC 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**




**Note 1:** Clock transition typically occurs within 2-4 TOSC.

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




**Note 1:** TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.
**2:** Clock transition typically occurs within 2-4 TOSC.


© 2009-2018 Microchip Technology Inc.

## 4.3 Sleep Mode

The power-managed Sleep mode in the PIC18F87K22 family of devices 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 LF-INTOSC source will continue to operate. If the SOSC 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). Alternately, the device will be clocked from the internal oscillator block if either the Two-Speed Start-up or the Fail-Safe Clock Monitor is enabled (see **Section 28.0 "Special Features of the CPU"**). In either case, the OSTS bit is set when the primary clock is providing the device clocks. The IDLEN and SCS bits are not affected by the wake-up.

## 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 a '1' when a SLEEP instruction is executed, the peripherals will be clocked from the clock source selected using the SCS<1:0> bits. The CPU, however, will not be clocked. The clock source status bits are not affected. This approach is a quick method to switch from a given Run mode to its corresponding Idle mode.

If the WDT is selected, the LF-INTOSC source will continue to operate. If the SOSC oscillator is enabled, it will also continue to run.

Since the CPU is not executing instructions, the only exits from any of the Idle modes are by interrupt, WDT time-out or a Reset. When a wake event occurs, CPU execution is delayed by an interval of TCSD (Parameter 38, Table 31-14) while it becomes ready to execute code. When the CPU begins executing code, it resumes with the same clock source for the current Idle mode. For example, when waking from RC_IDLE mode, the internal oscillator block will clock the CPU and peripherals (in other words, RC_RUN mode). The IDLEN and SCS bits are not affected by the wake-up.

While in any Idle mode or 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**

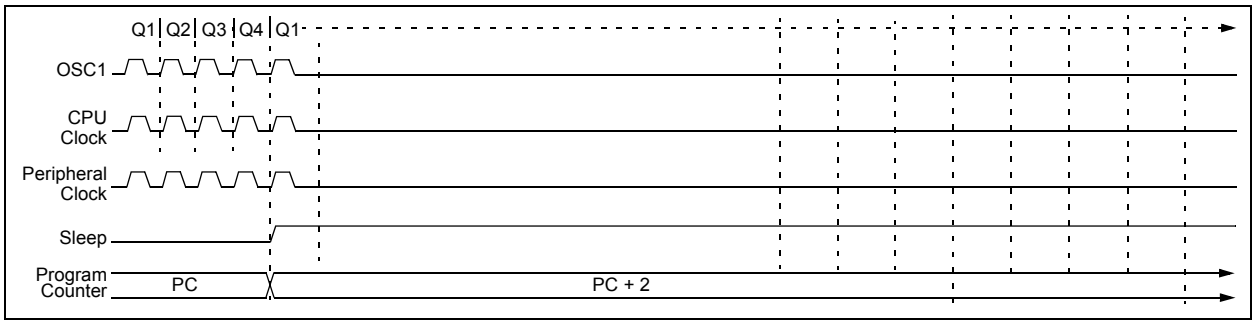
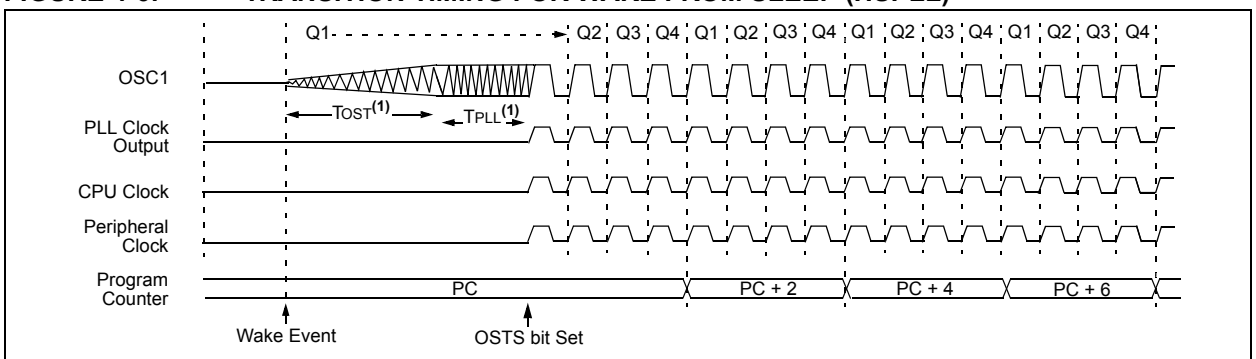



**FIGURE 4-6: TRANSITION TIMING FOR WAKE FROM SLEEP (HSPLL)**




**Note 1:** TOST = 1024 TOSC; TPLL = 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 clear the SCS bits and execute `SLEEP`. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified by the FOSC<3:0> 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 (Parameter 39, Table 31-14), is required between the wake event and the start of code execution. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTS bit remains set. The IDLEN and SCS bits are not affected by the wake-up (see Figure 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 SOSC 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 the IDLEN bit first, then set the SCS<1:0> bits to ‘`01`’ and execute `SLEEP`. When the clock source is switched to the SOSC oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the SOSCRUN bit is set.

When a wake event occurs, the peripherals continue to be clocked from the SOSC oscillator. After an interval of TCSD following the wake event, the CPU begins executing code being clocked by the SOSC oscillator. The IDLEN and SCS bits are not affected by the wake-up and the SOSC oscillator continues to run (see Figure 4-8).

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

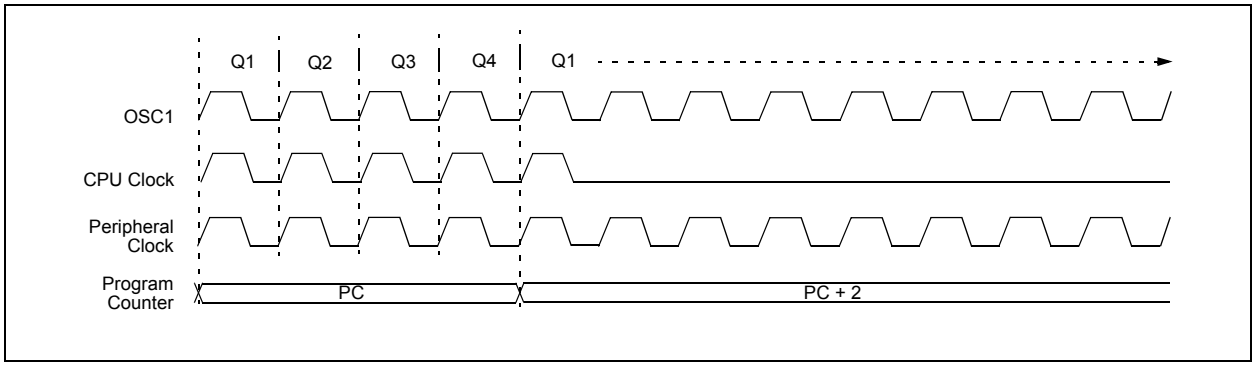
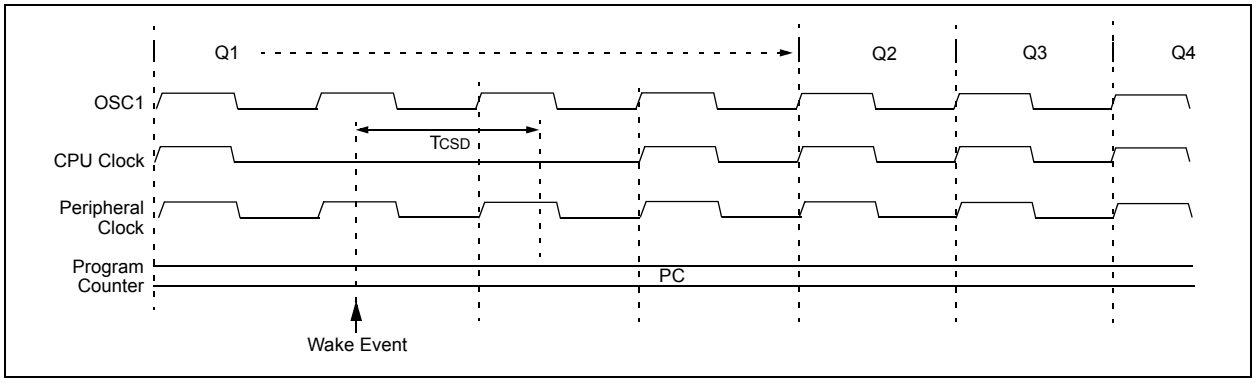



**FIGURE 4-8: TRANSITION TIMING FOR WAKE FROM IDLE TO RUN MODE**





© 2009-2018 Microchip Technology Inc.

#### 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 using the INTOSC multiplexer. This mode provides 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 set the SCS1 bit and execute SLEEP. To maintain software compatibility with future devices, it is recommended that SCS0 also be cleared, though its value is ignored. The INTOSC multiplexer may be used to select a higher clock frequency by modifying the IRCF bits before executing the SLEEP instruction. When the clock source is switched to the INTOSC multiplexer, the primary oscillator is shut down and the OSTS bit is cleared.

If the IRCF bits are set to any non-zero value, or the INTSRC/MFIOSEL bit is set, the INTOSC output is enabled. The HFIOFS/MFIOFS bits become set, after the INTOSC output becomes stable, after an interval of TIOBST (Parameter 38, Table 31-14). (For information on the HFIOFS/MFIOFS bits, see Table 4-3.)

Clocks to the peripherals continue while the INTOSC source stabilizes. The HFIOFS/MFIOFS bits will remain set if the IRCF bits were previously at a non-zero value or if INTSRC was set before the SLEEP instruction was executed and the INTOSC source was already stable. If the IRCF bits and INTSRC are all clear, the INTOSC output will not be enabled, the HFIOFS/MFIOFS bits will remain clear and there will be no indication of the current clock source.

When a wake event occurs, the peripherals continue to be clocked from the INTOSC multiplexer. After a delay of TCSD (Parameter 38, Table 31-14) following the wake event, the CPU begins executing code clocked by the INTOSC multiplexer. The IDLEN and SCS bits are not affected by the wake-up. The INTRC source will continue to run if either the WDT or the Fail-Safe Clock Monitor is enabled.

### 4.5 Selective Peripheral Module Control

Idle mode allows users to substantially reduce power consumption by stopping the CPU clock. Even so, peripheral modules still remain clocked, and thus, consume power. There may be cases where the application needs what this mode does not provide: the allocation of power resources to the CPU processing with minimal power consumption from the peripherals.

PIC18F87K22 family devices address this requirement by allowing peripheral modules to be selectively disabled, reducing or eliminating their power consumption. This can be done with two control bits:

- Peripheral Enable bit, generically named XXXEN – Located in the respective module's main control register
- Peripheral Module Disable (PMD) bit, generically named, XXXMD – Located in one of the PMDx Control registers (PMD0, PMD1, PMD2 or PMD3)

Disabling a module by clearing its XXXEN bit disables the module's functionality, but leaves its registers available to be read and written to. This reduces power consumption, but not by as much as the second approach.

Most peripheral modules have an enable bit.

In contrast, setting the PMD bit for a module disables all clock sources to that module, reducing its power consumption to an absolute minimum. In this state, the control and status registers associated with the peripheral are also disabled, so writes to those registers have no effect and read values are invalid. Many peripheral modules have a corresponding PMD bit.

There are four PMD registers in the PIC18F87K22 family devices: PMD0, PMD1, PMD2 and PMD3. These registers have bits associated with each module for disabling or enabling a particular peripheral.

© 2009-2018 Microchip Technology Inc.

**REGISTER 4-1: PMD3: PERIPHERAL MODULE DISABLE 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 |
|---|---|---|---|---|---|---|---|
| CCP10MD(1) | CCP9MD(1) | CCP8MD | CCP7MD | CCP6MD | CCP5MD | CCP4MD | TMR12MD(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 **CCP10MD:** PMD CCP10 Enable/Disable bit(1)
1 = Peripheral Module Disable (PMD) is enabled for CCP10, disabling all of its clock sources
0 = PMD is disabled for CCP10

bit 6 **CCP9MD:** PMD CCP9 Enable/Disable bit(1)
1 = Peripheral Module Disable (PMD) is enabled for CCP9, disabling all of its clock sources
0 = PMD is disabled for CCP9

bit 5 **CCP8MD:** PMD CCP8 Enable/Disable bit
1 = Peripheral Module Disable (PMD) is enabled for CCP8, disabling all of its clock sources
0 = PMD is disabled for CCP8

bit 4 **CCP7MD:** PMD CCP7 Enable/Disable bit
1 = Peripheral Module Disable (PMD) is enabled for CCP7, disabling all of its clock sources
0 = PMD is disabled for CCP7

bit 3 **CCP6MD:** PMD CCP6 Enable/Disable bit
1 = Peripheral Module Disable (PMD) is enabled for CCP6, disabling all of its clock sources
0 = PMD is disabled for CCP6

bit 2 **CCP5MD:** PMD CCP5 Enable/Disable bit
1 = Peripheral Module Disable (PMD) is enabled for CCP5, disabling all of its clock sources
0 = PMD is disabled for CCP5

bit 1 **CCP4MD:** PMD CCP4 Enable/Disable bit
1 = Peripheral Module Disable (PMD) is enabled for CCP4, disabling all of its clock sources
0 = PMD is disabled for CCP4

bit 0 **TMR12MD:** TMR12MD Disable bit(1)
1 = PMD is enabled and all TMR12MD clock sources are disabled
0 = PMD is disabled and TMR12MD is enabled

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).


© 2009-2018 Microchip Technology Inc.

**REGISTER 4-2: PMD2: PERIPHERAL MODULE DISABLE REGISTER 2**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| TMR10MD(1) | TMR8MD | TMR7MD(1) | TMR6MD | TMR5MD | CMP3MD | CMP2MD | CMP1MD |
| 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 **TMR10MD:** TMR10MD Disable bit(1)
- 1 = Peripheral Module Disable (PMD) is enabled and all TMR10MD clock sources are disabled
- 0 = PMD is disabled and TMR10MD is enabled

bit 6 **TMR8MD:** TMR8MD Disable bit
- 1 = PMD is enabled and all TMR8MD clock sources are disabled
- 0 = PMD is disabled and TMR8MD is enabled

bit 5 **TMR7MD:** TMR7MD Disable bit(1)
- 1 = PMD is enabled and all TMR7MD clock sources are disabled
- 0 = PMD is disabled and TMR7MD is enabled

bit 4 **TMR6MD:** TMR6MD Disable bit
- 1 = PMD is enabled and all TMR6MD clock sources are disabled
- 0 = PMD is disabled and TMR6MD is enabled

bit 3 **TMR5MD:** TMR5MD Disable bit
- 1 = PMD is enabled and all TMR5MD clock sources are disabled
- 0 = PMD is disabled and TMR5MD is enabled

bit 2 **CMP3MD:** PMD Comparator 3 Enable/Disable bit
- 1 = PMD is enabled for Comparator 3, disabling all of its clock sources
- 0 = PMD is disabled for Comparator 3

bit 1 **CMP2MD:** PMD Comparator 3 Enable/Disable bit
- 1 = PMD is enabled for Comparator 2, disabling all of its clock sources
- 0 = PMD is disabled for Comparator 2

bit 0 **CMP1MD:** PMD Comparator 3 Enable/Disable bit
- 1 = PMD is enabled for Comparator 1, disabling all of its clock sources
- 0 = PMD is disabled for Comparator 1

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

**REGISTER 4-3: PMD1: PERIPHERAL MODULE DISABLE 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 |
|---|---|---|---|---|---|---|---|
| PSPMD | CTMUMD | RTCCMD(1) | TMR4MD | TMR3MD | TMR2MD | TMR1MD | EMBMD |
| 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 **PSPMD:** Peripheral Module Disable (PMD) PSP Enable/Disable bit
1 = PMD is enabled for PSP, disabling all of its clock sources
0 = PMD is disabled for PSP

bit 6 **CTMUMD:** PMD CTMU Enable/Disable bit
1 = PMD is enabled for CTMU, disabling all of its clock sources
0 = PMD is disabled for CTMU

bit 5 **RTCCMD:** PMD RTCC Enable/Disable bit(1)
1 = PMD is enabled for RTCC, disabling all of its clock sources
0 = PMD is disabled for RTCC

bit 4 **TMR4MD:** TMR4MD Disable bit
1 = PMD is enabled and all TMR4MD clock sources are disabled
0 = PMD is disabled and TMR4MD is enabled

bit 3 **TMR3MD:** TMR3MD Disable bit
1 = PMD is enabled and all TMR3MD clock sources are disabled
0 = PMD is disabled and TMR3MD is enabled

bit 2 **TMR2MD:** TMR2MD Disable bit
1 = PMD is enabled and all TMR2MD clock sources are disabled
0 = PMD is disabled and TMR2MD is enabled

bit 1 **TMR1MD:** TMR1MD Disable bit
1 = PMD is enabled and all TMR1MD clock sources are disabled
0 = PMD is disabled and TMR1MD is enabled

bit 0 **EMBMD:** PMD EMB Enable/Disable bit
1 = PMD is enabled for EMB, disabling all of its clock sources
0 = PMD is disabled for EMB

**Note 1:** RTCCMD can only be set to '1' after an EECON2 unlock sequence. Refer to **Section 18.0 "Real-Time Clock and Calendar (RTCC)"** for the unlock sequence (Example 18-1).


© 2009-2018 Microchip Technology Inc.

**REGISTER 4-4: PMD0: PERIPHERAL MODULE DISABLE REGISTER 0**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |
| 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 **CCP3MD:** PMD ECCP3 Enable/Disable bit
1 = Peripheral Module Disable (PMD) is enabled for ECCP3, disabling all of its clock sources
0 = PMD is disabled for ECCP3

bit 6 **CCP2MD:** PMD ECCP2 Enable/Disable bit
1 = PMD is enabled for ECCP2, disabling all of its clock sources
0 = PMD is disabled for ECCP2

bit 5 **CCP1MD:** PMD ECCP1 Enable/Disable bit
1 = PMD is enabled for ECCP1, disabling all of its clock sources
0 = PMD is disabled for ECCP1

bit 4 **UART2MD:** PMD UART2 Enable/Disable bit
1 = PMD is enabled for UART2, disabling all of its clock sources
0 = PMD is disabled for UART2

bit 3 **UART1MD:** PMD UART1 Enable/Disable bit
1 = PMD is enabled for UART1, disabling all of its clock sources
0 = PMD is disabled for UART1

bit 2 **SSP2MD:** PMD MSSP2 Enable/Disable bit
1 = PMD is enabled for MSSP2, disabling all of its clock sources
0 = PMD is disabled for MSSP2

bit 1 **SSP1MD:** PMD MSSP1 Enable/Disable bit
1 = PMD is enabled for MSSP1, disabling all of its clock sources
0 = PMD is disabled for MSSP1

bit 0 **ADCMD:** PMD Analog/Digital Converter PMD Enable/Disable bit
1 = PMD is enabled for the Analog/Digital Converter, disabling all of its clock sources
0 = PMD is disabled for the Analog/Digital Converter

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

### 4.6.1 EXIT BY INTERRUPT

Any of the available interrupt sources can cause the device to exit from an Idle mode or 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 INTCONx or PIEx 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 11.0 "Interrupts"**).

### 4.6.2 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 28.2 "Watchdog Timer (WDT)"**).

Executing a `SLEEP` or `CLRWDT` instruction clears the WDT timer and postscaler, loses the currently selected clock source (if the Fail-Safe Clock Monitor is enabled) and modifies the IRCF bits in the OSCCON register (if the internal oscillator block is the device clock source).

### 4.6.3 EXIT BY RESET

Normally, the device is held in Reset by the Oscillator Start-up Timer (OST) until the primary clock becomes ready. At that time, the OSTS bit is set and the device begins executing code. If the internal oscillator block is the new clock source, the HFIOFS/MFIOFS bits are set instead.

The exit delay time from Reset to the start of code execution depends on both the clock sources before and after the wake-up, and the type of oscillator if the new clock source is the primary clock. Exit delays are summarized in Table 4-4.

Code execution can begin before the primary clock becomes ready. If either the Two-Speed Start-up (see **Section 28.4 "Two-Speed Start-up"**) or Fail-Safe Clock Monitor (see **Section 28.5 "Fail-Safe Clock Monitor"**) is enabled, the device may begin execution as soon as the Reset source has cleared. Execution is clocked by the INTOSC multiplexer driven by the internal oscillator block. Execution is clocked by the internal oscillator block until either the primary clock becomes ready or a power-managed mode is entered before the primary clock becomes ready; the primary clock is then shut down.

### 4.6.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

Certain exits from power-managed modes do not invoke the OST at all. The two cases are:

- When in PRI_IDLE mode, where the primary clock source is not stopped
- When the primary clock source is not any of the LP, XT, HS or HSPLL modes

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 (RC, EC and INTIO Oscillator modes). 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.


© 2009-2018 Microchip Technology Inc.

## 4.7 Ultra Low-Power Wake-up

The Ultra Low-Power Wake-up (ULPWU) on pin, RA0, allows a slow falling voltage to generate an interrupt without excess current consumption.

To use this feature:

1. Charge the capacitor on RA0 by configuring the RA0 pin to an output and setting it to '1'.
2. Stop charging the capacitor by configuring RA0 as an input.
3. Discharge the capacitor by setting the ULPEN and ULPSINK bits in the WDTCON register.
4. Configure Sleep mode.
5. Enter Sleep mode.

When the voltage on RA0 drops below VIL, the device wakes up and executes the next instruction.

This feature provides a low-power technique for periodically waking up the device from Sleep mode.

The time-out is dependent on the discharge time of the RC circuit on RA0.

When the ULPWU module wakes the device from Sleep mode, the ULPLVL bit (WDTCON<5>) is set. Software can check this bit upon wake-up to determine the wake-up source.

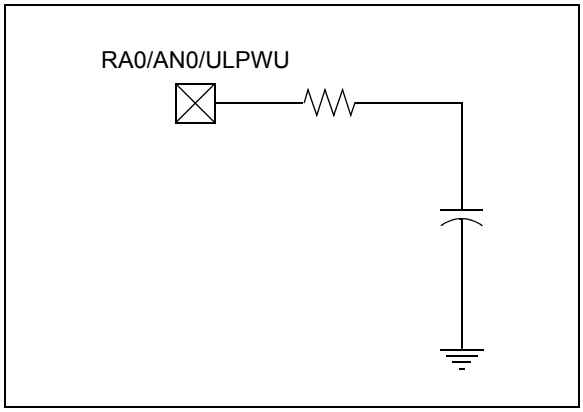
See Example 4-1 for initializing the ULPWU module.

**EXAMPLE 4-1: ULTRA LOW-POWER WAKE-UP INITIALIZATION**

```
            //***************************
            //Charge the capacitor on RA0
            //***************************
TRISAbits.TRISA0 = 0;
PORTAbits.RA0 = 1;
for(i = 0; i < 10000; i++) Nop();
            //*****************************
            //Stop Charging the capacitor
            //on RA0
            //*****************************
TRISAbits.TRISA0 = 1;
            //*****************************
            //Enable the Ultra Low Power
            //Wakeup module and allow
            //capacitor discharge
            //*****************************
WDTCONbits.ULPEN = 1;
WDTCONbits.ULPSINK = 1;
            //For Sleep
OSCCONbits.IDLEN = 0;
            //Enter Sleep Mode
            //
Sleep();
            //for sleep, execution will
            //resume here
```

A series resistor, between RA0 and the external capacitor, provides overcurrent protection for the RA0/AN0/ULPWU pin and enables software calibration of the time-out (see Figure 4-9).

**FIGURE 4-9: ULTRA LOW-POWER WAKE-UP INITIALIZATION**




A timer can be used to measure the charge time and discharge time of the capacitor. The charge time can then be adjusted to provide the desired delay in Sleep. This technique compensates for the affects of temperature, voltage and component accuracy. The peripheral can also be configured as a simple Programmable Low-Voltage Detect (LVD) or temperature sensor.

**Note:** For more information, see *AN879, "Using the Microchip Ultra Low-Power Wake-up Module"* (DS00879).

**TABLE 4-4: EXIT DELAY ON WAKE-UP BY RESET FROM SLEEP MODE OR ANY IDLE MODE (BY CLOCK SOURCES)**

| Power-Managed Mode | Clock Source(5) | Exit Delay | Clock Ready Status Bits |
|---|---|---|---|
| PRI_IDLE mode | LP, XT, HS | $T_{CSD}$(1) | OSTS |
| | HSPLL | | |
| | EC, RC | | |
| | HF-INTOSC(2) | | HFIOFS |
| | MF-INTOSC(2) | | MFIOFS |
| | LF-INTOSC | | None |
| SEC_IDLE mode | SOSC | $T_{CSD}$(1) | SOSCRUN |
| RC_IDLE mode | HF-INTOSC(2) | $T_{CSD}$(1) | HFIOFS |
| | MF-INTOSC(2) | | MFIOFS |
| | LF-INTOSC | | None |
| Sleep mode | LP, XT, HS | $T_{OST}$(3) | OSTS |
| | HSPLL | $T_{OST} + t_{rc}$(3) | |
| | EC, RC | $T_{CSD}$(1) | |
| | HF-INTOSC(2) | $T_{IOBST}$(4) | HFIOFS |
| | MF-INTOSC(2) | | MFIOFS |
| | LF-INTOSC | | None |

**Note 1:** $T_{CSD}$ (Parameter 38, Table 31-14) is a required delay when waking from Sleep and all Idle modes, and runs concurrently with any other required delays (see **Section 4.4 "Idle Modes"**).

**2:** Includes postscaler derived frequencies. On Reset, INTOSC defaults to HF-INTOSC at 8 MHz.

**3:** $T_{OST}$ is the Oscillator Start-up Timer (Parameter 32, Table 31-14). $T_{RC}$ is the PLL Lock-out Timer (Parameter F12, Table 31-8); it is also designated as $T_{PLL}$.

**4:** Execution continues during $T_{IOBST}$ (Parameter 39, Table 31-14), the INTOSC stabilization period.

**5:** The clock source is dependent upon the settings of the SCS (OSCCON<1:0>), IRCF (OSCCON<6:4>) and FOSC (CONFIG1H<3:0>) bits.


© 2009-2018 Microchip Technology Inc.

## 5.0 RESET

The PIC18F87K22 family of devices differentiates between various kinds of Reset:

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

This section discusses Resets generated by $\overline{\text{MCLR}}$, POR and BOR, and covers the operation of the various start-up timers. Stack Reset events are covered in **Section 6.1.3.4 "Stack Full and Underflow Resets"**. WDT Resets are covered in **Section 28.2 "Watchdog Timer (WDT)"**.

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

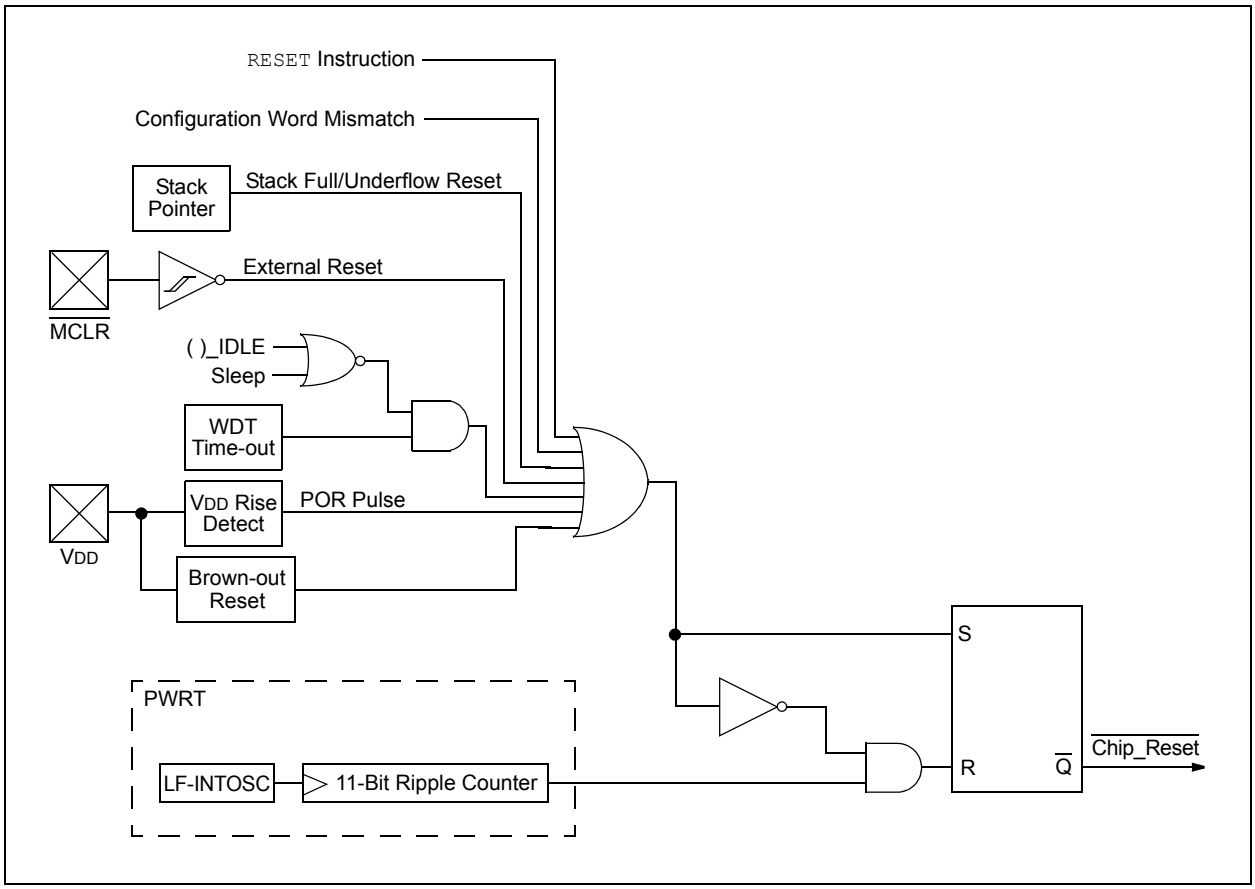
## 5.1 RCON Register

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 11.0 "Interrupts"**.

**FIGURE 5-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT**



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

| R/W-0 | R/W-1 | R/W-1 | R/W-1 | R-1 | R-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | SBOREN | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| 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 **SBOREN:** BOR Software Enable bit
If BOREN<1:0> = 01:
1 = BOR is enabled
0 = BOR is disabled
If BOREN<1:0> = 00, 10 or 11:
Bit is disabled and read as '0'.

bit 5 **$\overline{CM}$:** 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 **$\overline{RI}$:** 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 **$\overline{TO}$:** Watchdog Time-out Flag bit
1 = Set by power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{PD}$:** 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 **$\overline{POR}$:** 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 **$\overline{BOR}$:** 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 $\overline{POR}$ bit be set after a Power-on Reset has been detected, so that subsequent Power-on Resets may be detected.

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


© 2009-2018 Microchip Technology Inc.

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

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

The $\overline{MCLR}$ 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 VDD rises above a certain threshold. This allows the device to start in the initialized state when VDD is adequate for operation.

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

When the device starts normal operation (exiting the Reset condition), device operating parameters (such as voltage, frequency and temperature) 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 $\overline{POR}$ bit (RCON<1>). The state of the bit is set to '0' whenever a Power-on Reset occurs and does not change for any other Reset event. $\overline{POR}$ 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 PIC18F87K22 family has four BOR Power modes:

- High-Power BOR
- Medium Power BOR
- Low-Power BOR
- Zero-Power BOR

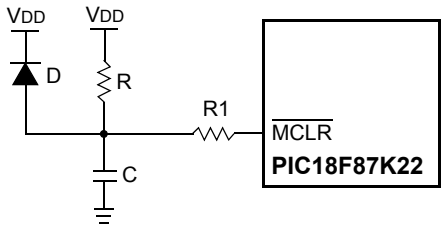
Each power Mode is selected by the BORPWR<1:0> bits setting (CONFIG2L<6:5>). For low, medium and high-power BOR, the module monitors the VDD depending on the BORV<1:0> setting (CONFIG1L<3:2>). A BOR event re-arms the Power-on Reset. It also causes a Reset, depending on which of the trip levels has been set: 1.8V, 2V, 2.7V or 3V.

BOR is enabled by the BOREN<1:0> bits (CONFIG2L<2:1>) and the SBOREN bit (RCON<6>). Typical power consumption is listed as Parameter D022A in **Section 31.0 "Electrical Characteristics"**.

In Zero-Power BOR (ZPBORMV), the module monitors the VDD voltage and re-arms the POR at about 2V. ZPBORMV does not cause a Reset, but re-arms the POR.

The BOR accuracy varies with its power level. The lower the power setting, the less accurate the BOR trip levels are. Therefore, the high-power BOR has the highest accuracy and the low-power BOR has the lowest accuracy. The trip levels (BVDD, Parameter D005), current consumption (**Section 31.2 "DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended)"**) and time required below BVDD (TBOR, Parameter 35) can all be found in **Section 31.0 "Electrical Characteristics"**.

**FIGURE 5-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)**




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

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

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

### 5.4.1 DETECTING BOR

The $\overline{BOR}$ 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 $\overline{BOR}$ alone. A more reliable method is to simultaneously check the state of both $\overline{POR}$ and $\overline{BOR}$. This assumes that the $\overline{POR}$ bit is reset to '1' in software immediately after any Power-on Reset event. If $\overline{BOR}$ is '0' while $\overline{POR}$ is '1', it can be reliably assumed that a Brown-out Reset event has occurred.

LP-BOR cannot be detected with the $\overline{BOR}$ bit in the RCON register. LP-BOR can rearm the $\overline{POR}$ and can cause a Power-on Reset.

## 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 that can cause widespread, single bit changes throughout the device and result in catastrophic failure.

In PIC18F87K22 family 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 $\overline{CM}$ bit (RCON<5>). The state of the bit is set to '0' whenever a CM event occurs and 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 Reset. 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)

PIC18F87K22 family devices incorporate an on-chip Power-up Timer (PWRT) to help regulate the Power-on Reset process. The PWRT is enabled by setting the PWRTEN bit (CONFIG2L<0>). The main function is to ensure that the device voltage is stable before code is executed.

The Power-up Timer (PWRT) of the PIC18F87K22 family devices is a 11-bit counter that uses the LF-INTOSC source as the clock input.

While the PWRT is counting, the device is held in Reset.

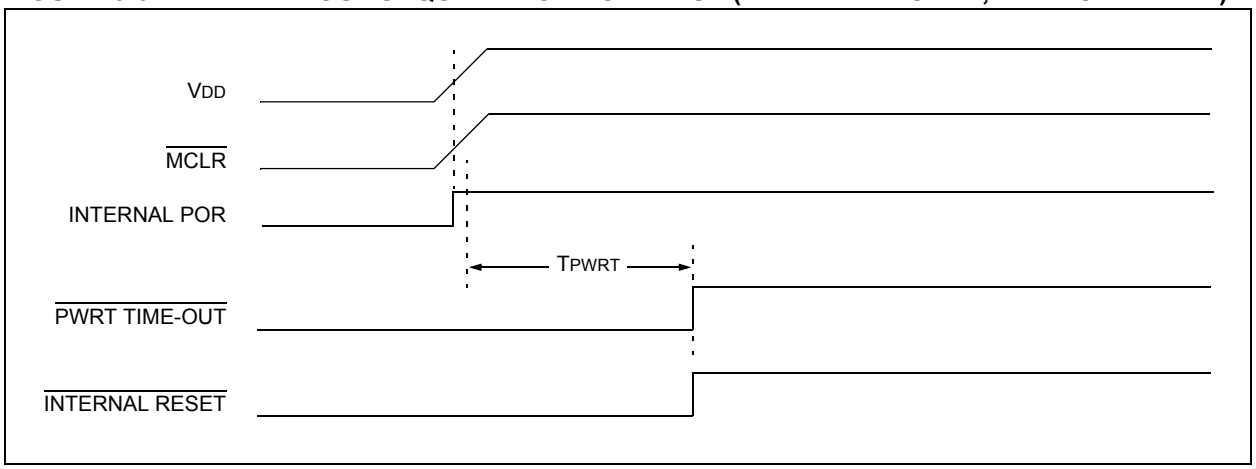
The power-up time delay depends on the LF-INTOSC clock and will vary from chip-to-chip due to temperature and process variation. See DC Parameter 33 for details.

### 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, if $\overline{MCLR}$ is kept low long enough, the PWRT will expire. Bringing $\overline{MCLR}$ high will begin execution immediately (Figure 5-5). This is useful for testing purposes or for synchronizing more than one PIC18 device operating in parallel.

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





© 2009-2018 Microchip Technology Inc.

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

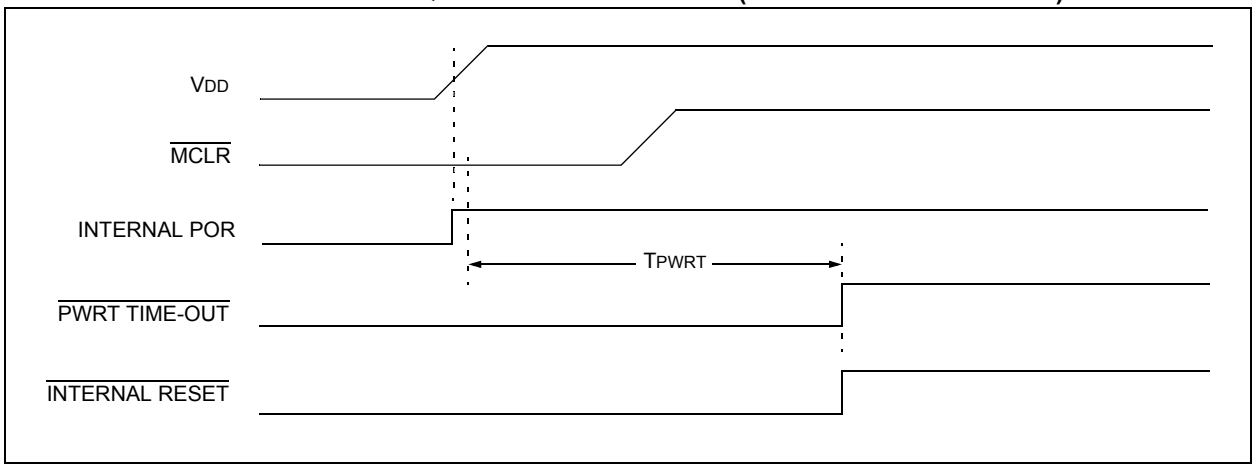



**FIGURE 5-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2**

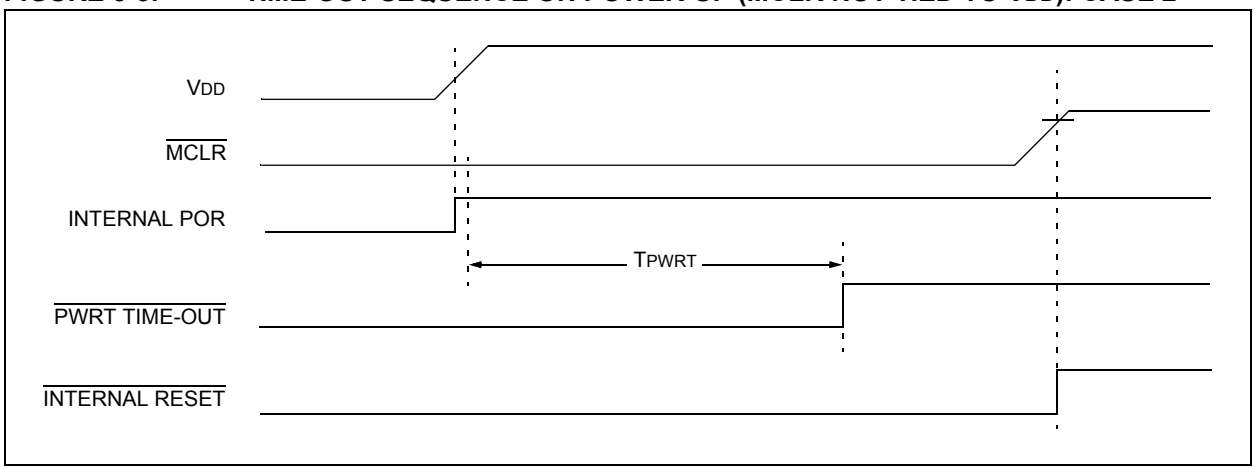
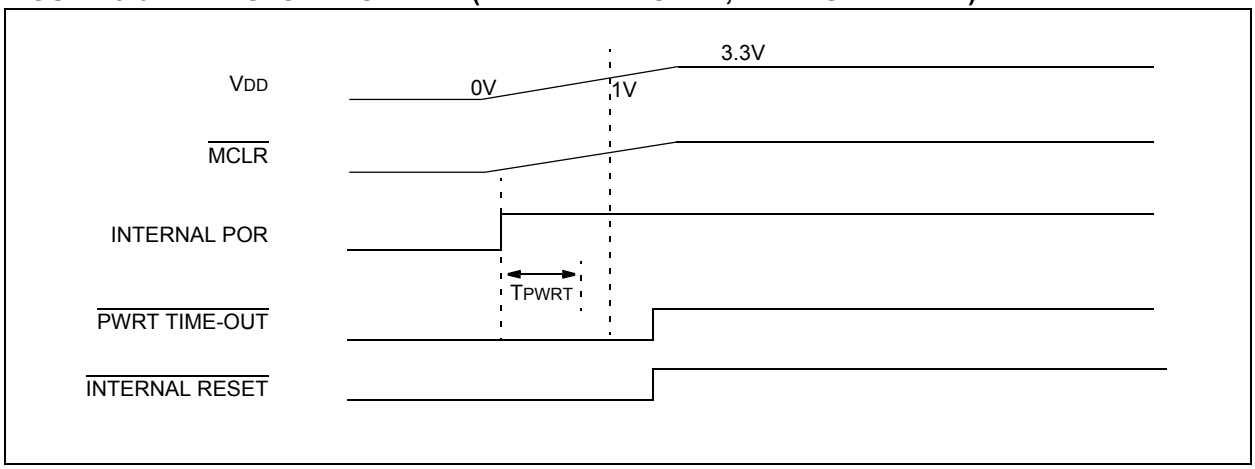



**FIGURE 5-6: SLOW RISE TIME (MCLR TIED TO VDD, VDD RISE > TPWRT)**



## 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 5-1. These bits are used in software to determine the nature of the Reset.

Table 5-2 describes the Reset states for all of the Special Function Registers. These are categorized by Power-on and Brown-out Resets, Master Clear and WDT Resets, and WDT wake-ups.

**TABLE 5-1: 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).


© 2009-2018 Microchip Technology Inc.

**TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS**

| 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 | PIC18F6XK22 | PIC18F8XK22 | ---0 0000 | ---0 0000 | ---0 uuuu(1) |
| TOSH | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu(1) |
| TOSL | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu(1) |
| STKPTR | PIC18F6XK22 | PIC18F8XK22 | 00-0 0000 | uu-0 0000 | uu-u uuuu(1) |
| PCLATU | PIC18F6XK22 | PIC18F8XK22 | ---0 0000 | ---0 0000 | ---u uuuu |
| PCLATH | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCL | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | PC + 2(2) |
| TBLPTRU | PIC18F6XK22 | PIC18F8XK22 | --00 0000 | --00 0000 | --uu uuuu |
| TBLPTRH | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TBLPTRL | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TABLAT | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PRODH | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PRODL | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INTCON | PIC18F6XK22 | PIC18F8XK22 | 0000 000x | 0000 000u | uuuu uuuu(3) |
| INTCON2 | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu(3) |
| INTCON3 | PIC18F6XK22 | PIC18F8XK22 | 1100 0000 | 1100 0000 | uuuu uuuu(3) |
| INDF0 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| POSTINC0 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| POSTDEC0 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| PREINC0 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| PLUSW0 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| FSR0H | PIC18F6XK22 | PIC18F8XK22 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR0L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| WREG | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| INDF1 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| POSTINC1 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| POSTDEC1 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| PREINC1 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| PLUSW1 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| FSR1H | PIC18F6XK22 | PIC18F8XK22 | ---- 0000 | ---- 0000 | ---- uuuu |
| FSR1L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| BSR | PIC18F6XK22 | PIC18F8XK22 | ---- 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 affected (to cause wake-up).

**4:** See Table 5-1 for Reset value for specific condition.

**TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| INDF2 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| POSTINC2 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| POSTDEC2 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| PREINC2 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| PLUSW2 | PIC18F6XK22 | PIC18F8XK22 | N/A | N/A | N/A |
| FSR2H | PIC18F6XK22 | PIC18F8XK22 | ---- xxxx | ----uuuu | ---- uuuu |
| FSR2L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| STATUS | PIC18F6XK22 | PIC18F8XK22 | ---x xxxx | ---u uuuu | ---u uuuu |
| TMR0H | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| TMR0L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T0CON | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| SPBRGH1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| OSCCON | PIC18F6XK22 | PIC18F8XK22 | 0110 q000 | 0110 q000 | uuuu quuu |
| IPR5 | PIC18F65K22 | PIC18F85K22 | ---1 -111 | ---1 -111 | ---u -uuu |
| IPR5 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 1000 0000 | 1000 0000 | uuuu uuuu |
| WDTCON | PIC18F6XK22 | PIC18F8XK22 | 0-x0 -000 | 0-x0 -000 | u-uu -uuu |
| RCON | PIC18F6XK22 | PIC18F8XK22 | 0111 11qq | 0uqq qquu | uuuu qquu |
| TMR1H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR1L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T1CON | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| TMR2 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PR2 | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| T2CON | PIC18F6XK22 | PIC18F8XK22 | -000 0000 | -000 0000 | -uuu uuuu |
| SSP1BUF | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| SSP1ADD | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP1STAT | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP1CON1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP1CON2 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ADRESH | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADRESL | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ADCON0 | PIC18F6XK22 | PIC18F8XK22 | -000 0000 | -000 0000 | -uuu uuuu |
| ADCON1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ADCON2 | PIC18F6XK22 | PIC18F8XK22 | 0-00 0000 | 0-00 0000 | u-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 affected (to cause wake-up).

**4:** See Table 5-1 for Reset value for specific condition.


© 2009-2018 Microchip Technology Inc.

**TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| ECCP1AS | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP1DEL | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CCPR1H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR1L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP1CON | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PIR5 | PIC18F65K22 | PIC18F85K22 | ---0 -000 | ---0 -000 | ---u -uuu |
| PIR5 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PIE5 | PIC18F65K22 | PIC18F85K22 | ---0 0000 | ---0 0000 | ---u uuuu(1) |
| PIE5 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 000 | 0000 0000 | uuuu uuuu(1) |
| IPR4 | PIC18F65K22 | PIC18F85K22 | --11 1111 | --11 1111 | --uu uuuu |
| IPR4 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIR4 | PIC18F65K22 | PIC18F85K22 | --00 0000 | --00 0000 | --uu uuuu(1) |
| PIR4 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0000 | 0000 0000 | uuuu uuuu(1) |
| PIE4 | PIC18F65K22 | PIC18F85K22 | --00 0000 | --00 0000 | --uu uuuu |
| PIE4 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CVRCON | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CMSTAT | PIC18F6XK22 | PIC18F8XK22 | xxx- ---- | xxx- ---- | uuu- ---- |
| TMR3H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR3L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T3CON | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0x00 | uuuu uuuu |
| T3GCON | PIC18F6XK22 | PIC18F8XK22 | 0000 0x00 | 0000 0000 | uuuu uuuu |
| SPBRG1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RCREG1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXREG1 | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| TXSTA1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0010 | 0000 0010 | uuuu uuuu |
| RCSTA1 | PIC18F6XK22 | PIC18F8XK22 | 0000 000x | 0000 000x | uuuu uuuu |
| T1GCON | PIC18F6XK22 | PIC18F8XK22 | 0000 0x00 | 0000 0x00 | uuuu -uuu |
| IPR6 | PIC18F6XK22 | PIC18F8XK22 | ---1 -111 | ---1 -111 | ---u -uuu |
| HLVDCON | PIC18F6XK22 | PIC18F8XK22 | 0000 0101 | 0000 0101 | uuuu uuuu |
| PSPCON | PIC18F6XK22 | PIC18F8XK22 | 0000 ---- | 0000 ---- | uuuu ---- |
| PIR6 | PIC18F6XK22 | PIC18F8XK22 | ---0 -000 | ---0 -000 | ---u -uuu |

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

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

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

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

**4:** See Table 5-1 for Reset value for specific condition.

**TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| IPR3 | PIC18F6XK22 | PIC18F8XK22 | 1-11 1111 | 1-11 1111 | u-uu uuuu |
| PIR3 | PIC18F6XK22 | PIC18F8XK22 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| PIE3 | PIC18F6XK22 | PIC18F8XK22 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| IPR2 | PIC18F6XK22 | PIC18F8XK22 | 1-11 1111 | 1-11 1111 | u-uu uuuu |
| PIR2 | PIC18F6XK22 | PIC18F8XK22 | 0-10 0000 | 0-10 0000 | u-uu uuuu |
| PIE2 | PIC18F6XK22 | PIC18F8XK22 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| IPR1 | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PIR1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PIE1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PSTR1CON | PIC18F6XK22 | PIC18F8XK22 | 00-0 0001 | 00-0 0001 | uu-u uuuu |
| OSCTUNE | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TRISJ | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISH | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISG | PIC18F6XK22 | PIC18F8XK22 | ---1 1111 | ---1 1111 | ---u uuuu |
| TRISF | PIC18F6XK22 | PIC18F8XK22 | 1111 111- | 1111 111- | ---u uuuu |
| TRISE | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISD | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISC | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISB | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| TRISA | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| LATJ | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATH | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATG | PIC18F6XK22 | PIC18F8XK22 | ---x xxxx | ---u uuuu | ---u uuuu |
| LATF | PIC18F6XK22 | PIC18F8XK22 | xxxx xxx- | uuuu uuu- | uuuu uuu- |
| LATE | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATD | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATC | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATB | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| LATA | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTJ | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| PORTH | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| PORTG | PIC18F6XK22 | PIC18F8XK22 | --xx xxxx | --xx xxxx | --uu uuuu |
| PORTF | PIC18F6XK22 | PIC18F8XK22 | xxxx xxx- | xxxx xxx- | uuuu uuu- |
| PORTE | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| PORTD | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | xxxx xxxx | 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 affected (to cause wake-up).

**4:** See Table 5-1 for Reset value for specific condition.


© 2009-2018 Microchip Technology Inc.

**TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| PORTC | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTB | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PORTA | PIC18F6XK22 | PIC18F8XK22 | xx0x 0000 | uu0u 0000 | uuuu uuuu |
| EECON1 | PIC18F6XK22 | PIC18F8XK22 | xx-0 x000 | uu-0 u000 | uu-u uuuu |
| EECON2 | PIC18F6XK22 | PIC18F8XK22 | ---- ---- | ---- ---- | ---- ---- |
| TMR5H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR5L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T5CON | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| T5GCON | PIC18F6XK22 | PIC18F8XK22 | 0000 0x00 | uuuu uuuu | uuuu uuuu |
| CCPR4H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR4L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP4CON | PIC18F6XK22 | PIC18F8XK22 | --00 0000 | --00 0000 | --uu uuuu |
| CCPR5H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR5L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP5CON | PIC18F6XK22 | PIC18F8XK22 | --00 0000 | --00 0000 | --uu uuuu |
| CCPR6H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR6L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP6CON | PIC18F6XK22 | PIC18F8XK22 | --00 0000 | --00 0000 | --uu uuuu |
| CCPR7H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR7L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP7CON | PIC18F6XK22 | PIC18F8XK22 | --00 0000 | --00 0000 | --uu uuuu |
| TMR4 | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| PR4 | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | 1111 1111 |
| T4CON | PIC18F6XK22 | PIC18F8XK22 | -111 1111 | -111 1111 | -uuu uuuu |
| SSP2BUF | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| SSP2ADD | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP2STAT | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP2CON1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SSP2CON2 | PIC18F6XK22 | PIC18F8XK22 | 0100 0000 | 0000 0000 | uuuu uuuu |
| BAUDCON1 | PIC18F6XK22 | PIC18F8XK22 | 0100 0-00 | 0100 0-00 | uuuu u-uu |
| OSCCON2 | PIC18F6XK22 | PIC18F8XK22 | -0-- 0-x0 | -0-- 0-u0 | -u-- u-uu |
| EEADRH | PIC18F6XK22 | PIC18F8XK22 | ---- --00 | ---- --00 | ---- --uu |
| EEADR | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| EEDATA | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PIE6 | PIC18F6XK22 | PIC18F8XK22 | ---0 -000 | ---0 -000 | ---u -uuu |

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

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

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

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

**4:** See Table 5-1 for Reset value for specific condition.

**TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| RTCCFG | PIC18F6XK22 | PIC18F8XK22 | 0-00 0000 | u-uu uuuu | u-uu uuuu |
| RTCCAL | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| RTCVALH | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| RTCVALL | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| ALRMCFG | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| ALRMRPT | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| ALRMVALH | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| ALRMVALL | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CTMUCONH | PIC18F6XK22 | PIC18F8XK22 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| CTMUCONL | PIC18F6XK22 | PIC18F8XK22 | 0000 00xx | 0000 00xx | uuuu uuuu |
| CTMUICONH | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CM1CON | PIC18F6XK22 | PIC18F8XK22 | 0001 1111 | 0001 1111 | uuuu uuuu |
| PADCFG1 | PIC18F6XK22 | PIC18F8XK22 | 00-- -00- | uu-- -uu- | uu-- -uu- |
| PADCFG1 | PIC18F6XK22 | PIC18F8XK22 | 000- -00- | uuu- -uu- | uuu- -uu- |
| ECCP2AS | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP2DEL | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CCPR2H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR2L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP2CON | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP3AS | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ECCP3DEL | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CCPR3H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR3L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP3CON | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CCPR8H | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR8L | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP8CON | PIC18F6XK22 | PIC18F8XK22 | --00 0000 | --00 0000 | --uu uuuu |
| CCPR9H | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR9L | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCP9CON | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | --00 0000 | --00 0000 | --uu uuuu |
| CCPR10H | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| CCPR10L | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 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 affected (to cause wake-up).

**4:** See Table 5-1 for Reset value for specific condition.


© 2009-2018 Microchip Technology Inc.

**TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| CCP10CON | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | --00 0000 | --00 0000 | --uu uuuu |
| TMR7H | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TMR7L | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| T7CON | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| T7GCON | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0x00 | 0000 0x00 | uuuu uuuu |
| TMR6 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PR6 | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| T6CON | PIC18F6XK22 | PIC18F8XK22 | -000 0000 | -000 0000 | -uuu uuuu |
| TMR8 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PR8 | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| T8CON | PIC18F6XK22 | PIC18F8XK22 | -000 0000 | -000 0000 | -uuu uuuu |
| TMR10 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PR10 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| T10CON | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | -000 0000 | -000 0000 | -uuu uuuu |
| TMR12 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PR12 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 1111 1111 | 1111 1111 | uuuu uuuu |
| T12CON | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | -000 0000 | -000 0000 | -uuu uuuu |
| CM2CON | PIC18F6XK22 | PIC18F8XK22 | 0001 1111 | 0001 1111 | uuuu uuuu |
| CM3CON | PIC18F6XK22 | PIC18F8XK22 | 0001 1111 | 0001 1111 | uuuu uuuu |
| CCPTMRS0 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| CCPTMRS1 | PIC18F6XK22 | PIC18F8XK22 | 00-0 -000 | uu-u -uuu | uu-u -uuu |
| CCPTMRS2 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | ---0 -000 | ---u -uuu | ---u -uuu |
| CCPTMRS2 | PIC18F65K22 | PIC18F85K22 | ---- --00 | ---- --uu | ---- --uu |
| REFOCON | PIC18F6XK22 | PIC18F8XK22 | 0-00 0000 | u-uu uuuu | u-uu uuuu |
| ODCON1 | PIC18F6XK22 | PIC18F8XK22 | 000- ---0 | uuu- ---u | uuu- ---u |

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

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

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

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

**4:** See Table 5-1 for Reset value for specific condition.

**TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)**

| Register | Applicable Devices | | Power-on Reset, Brown-out Reset | $\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets, CM Resets | Wake-up via WDT or Interrupt |
|---|---|---|---|---|---|
| ODCON2 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0000 | uuuu uuuu | uuuu uuuu |
| ODCON2 | PIC18F65K22 | PIC18F85K22 | --00 0000 | --uu uuuu | --uu uuuu |
| ODCON3 | PIC18F6XK22 | PIC18F8XK22 | 00-- ---0 | uu-- ---u | uu-- ---u |
| MEMCON | PIC18F6XK22 | PIC18F8XK22 | 0-00 --00 | 0-00 --00 | u-uu --uu |
| ANCON0 | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | uuuu uuuu | uuuu uuuu |
| ANCON1 | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | uuuu uuuu | uuuu uuuu |
| ANCON2 | PIC18F6XK22 | PIC18F8XK22 | 1111 1111 | uuuu uuuu | uuuu uuuu |
| RCSTA2 | PIC18F6XK22 | PIC18F8XK22 | 0000 000x | 0000 000x | uuuu uuuu |
| TXSTA2 | PIC18F6XK22 | PIC18F8XK22 | 0000 0010 | 0000 0010 | uuuu uuuu |
| BAUDCON2 | PIC18F6XK22 | PIC18F8XK22 | 0100 0-00 | 0100 0-00 | uuuu u-uu |
| SPBRGH2 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SPBRG2 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RCREG2 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TXREG2 | PIC18F6XK22 | PIC18F8XK22 | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| PSTR2CON | PIC18F6XK22 | PIC18F8XK22 | 00-0 0001 | 00-0 0001 | uu-u uuuu |
| PSTR3CON | PIC18F6XK22 | PIC18F8XK22 | 00-0 0001 | 00-0 0001 | uu-u uuuu |
| PMD0 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMD1 | PIC18F6XK22 | PIC18F8XK22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMD2 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMD2 | PIC18F65K22 | PIC18F85K22 | -0-0 0000 | -0-0 0000 | -u-u uuuu |
| PMD3 | PIC18F66K22<br>PIC18F67K22 | PIC18F86K22<br>PIC18F87K22 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PMD3 | PIC18F65K22 | PIC18F85K22 | --00 000- | --00 000- | --uu uuu- |

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

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

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

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

**4:** See Table 5-1 for Reset value for specific condition.


© 2009-2018 Microchip Technology Inc.

## 6.0 MEMORY ORGANIZATION

PIC18F87K22 family devices have these types of memory:

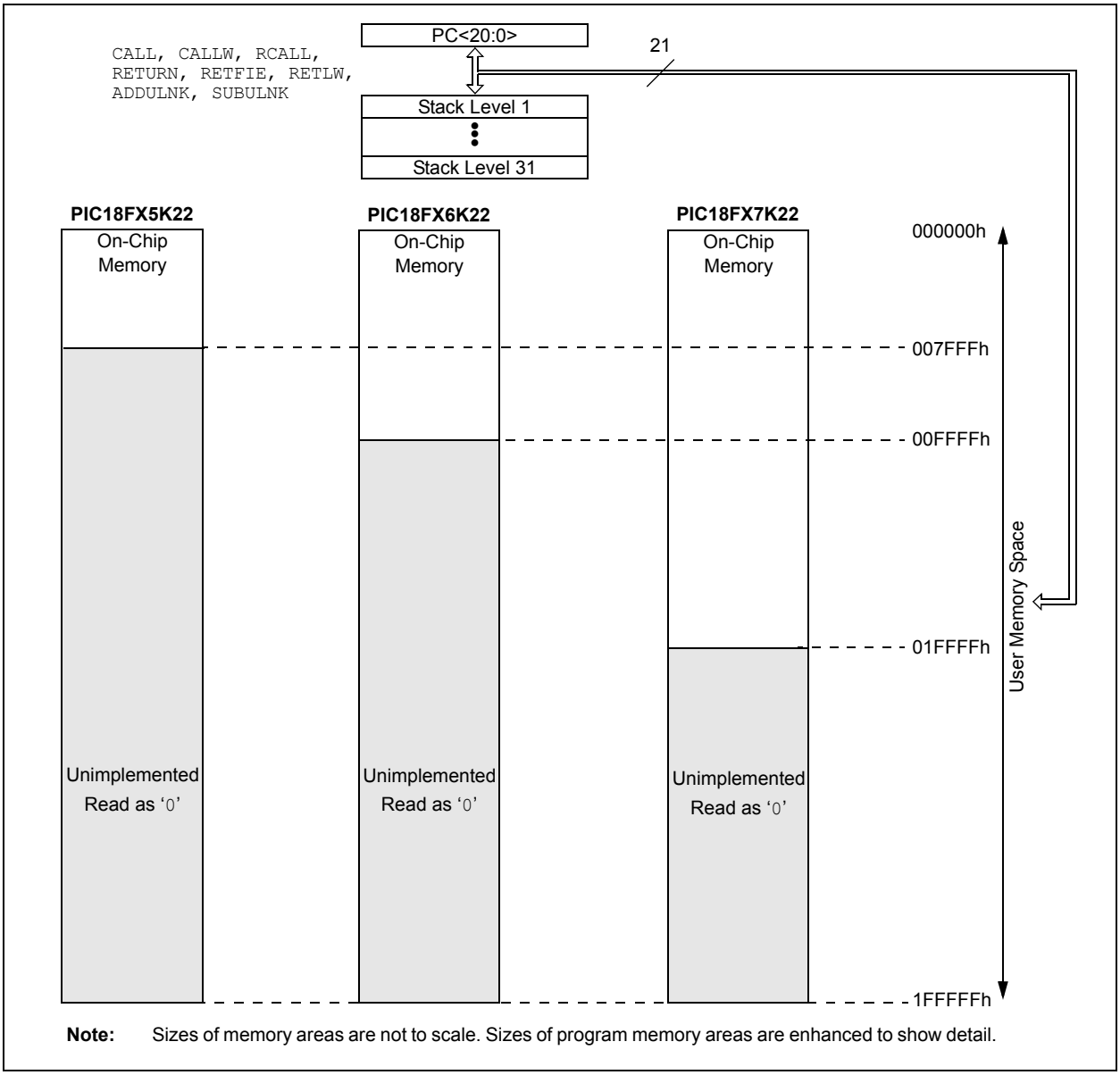
- Program Memory
- Data RAM
- Data EEPROM

As Harvard architecture devices, the data and program memories use separate buses. This enables concurrent access of the two memory spaces.

The data EEPROM, for practical purposes, can be regarded as a peripheral device because it is addressed and accessed through a set of control registers.

Additional detailed information on the operation of the Flash program memory is provided in **Section 7.0 "Flash Program Memory"**. The data EEPROM is discussed separately in **Section 9.0 "Data EEPROM Memory"**.

**FIGURE 6-1: MEMORY MAPS FOR PIC18F87K22 FAMILY DEVICES**




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

## 6.1 Program Memory Organization

PIC18 microcontrollers implement a 21-bit Program Counter that 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 PIC18F87K22 family offers a range of on-chip Flash program memory sizes, from 32 Kbytes (up to 16,384 single-word instructions) to 128 Kbytes (65,536 single-word instructions).

- PIC18F65K22 and PIC18F85K22 – 32 Kbytes of Flash memory, storing up to 16,384 single-word instructions
- PIC18F66K22 and PIC18F86K22 – 64 Kbytes of Flash memory, storing up to 32,768 single-word instructions
- PIC18F67K22 and PIC18F87K22 – 128 Kbytes of Flash memory, storing up to 65,536 single-word instructions

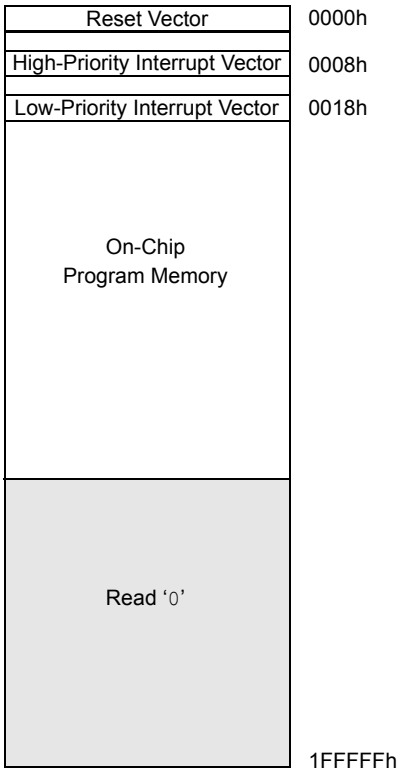
The program memory maps for individual family members are shown in Figure 6-1.

### 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 handling high-priority and low-priority interrupts. The high-priority interrupt vector is located at 0008h and the low-priority interrupt vector is at 0018h. The locations of these vectors are shown, in relation to the program memory map, in Figure 6-2.

**FIGURE 6-2: HARD VECTOR FOR PIC18F87K22 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.


© 2009-2018 Microchip Technology Inc.

### 6.1.2 PROGRAM COUNTER

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and 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 and 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 6.1.5.1 "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 two 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.

### 6.1.3 RETURN ADDRESS STACK

The return address stack enables execution of any combination of up to 31 program calls and interrupts. 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. The value also is pulled off the stack 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.

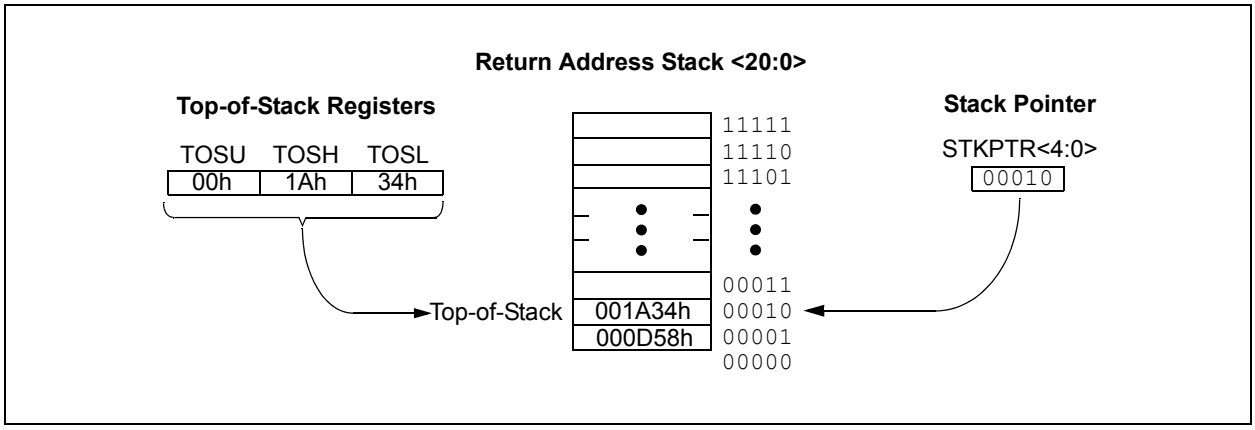
The Stack Pointer is initialized to '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits indicate if the stack is full, has overflowed or has underflowed.

#### 6.1.3.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, holds the contents of the stack location pointed to by the STKPTR register (Figure 6-3). This allows users to implement a software stack, if necessary. After a CALL, RCALL or interrupt (or 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.

While accessing the stack, users must disable the Global Interrupt Enable bits to prevent inadvertent stack corruption.

**FIGURE 6-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS**



### 6.1.3.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 6-1) 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.

What happens when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit. (For a description of the device Configuration bits, see **Section 28.1 "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. |
|---|---|

### 6.1.3.3 PUSH and POP Instructions

Since the Top-of-Stack (TOS) is readable and writable, the ability to push values onto the stack and pull values off of 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 6-1: 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 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)
1 = Stack has become full or overflowed
0 = Stack has not become full or overflowed

bit 6 **STKUNF:** Stack Underflow Flag bit(1)
1 = Stack underflow has occurred
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.


© 2009-2018 Microchip Technology Inc.

#### 6.1.3.4 Stack Full and Underflow Resets

Device Resets on stack overflow and stack underflow conditions are enabled by setting the STVREN bit (CONFIG4L<0>). 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.

### 6.1.4 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
```

### 6.1.5 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

#### 6.1.5.1 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 two (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
        .
        .
        .
```

#### 6.1.5.2 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.

The 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, with the instruction 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 6-4.

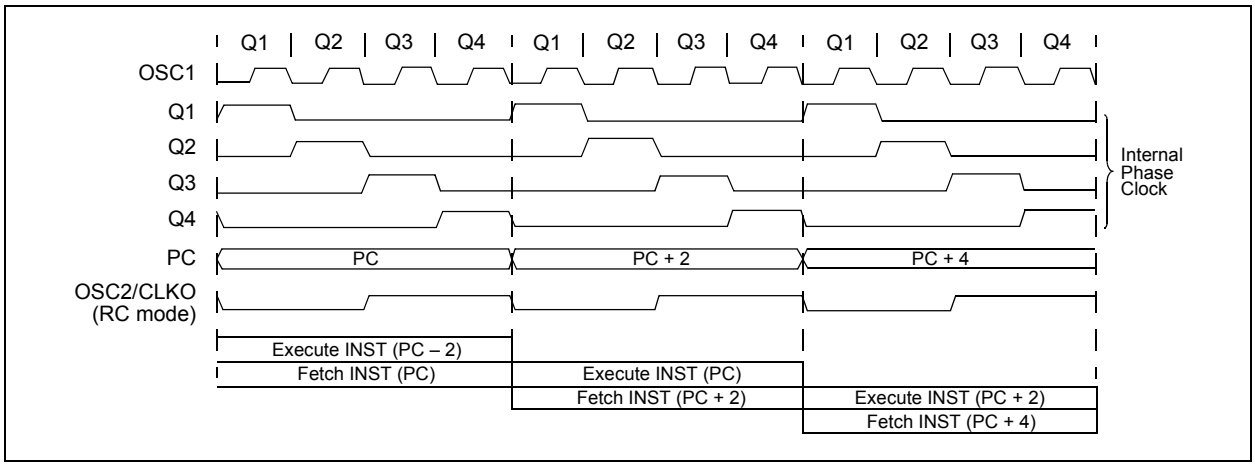
### 6.2.2 INSTRUCTION FLOW/PIPELINING

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 take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction (such as `GOTO`) causes the Program Counter to change, two cycles are required to complete the instruction. (See Example 6-3.)

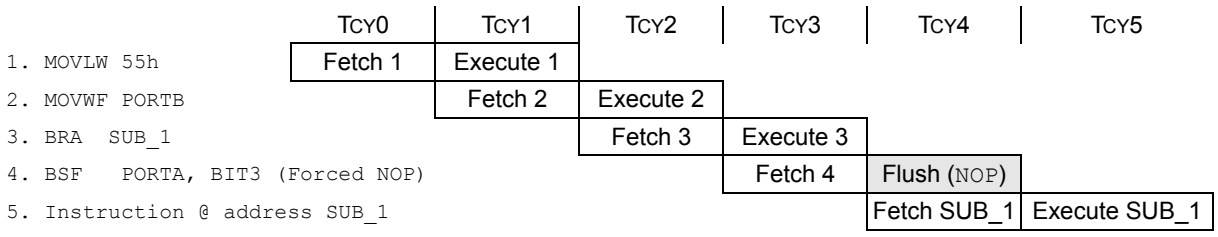
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

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 6-4: CLOCK/INSTRUCTION CYCLE**




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




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.


© 2009-2018 Microchip Technology Inc.

### 6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as two 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 two and the LSB will always read '0' (see **Section 6.1.2 "Program Counter"**).

Figure 6-5 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 6-5 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. For more details on the instruction set, see **Section 29.0 "Instruction Set Summary"**.

**FIGURE 6-5: INSTRUCTIONS IN PROGRAM MEMORY**

| | | | LSB = 1 | LSB = 0 | Word Address ↓ |
|---|---|---|---|---|---|
| Program Memory Byte Locations → | | | | | 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:** For information on two-word instructions in the extended instruction set, see **Section 6.5 "Program Memory and 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 4,096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. PIC18FX6K22 and PIC18FX7K22 devices implement all 16 complete banks, for a total of 4 Kbytes. PIC18FX5K22 devices implement only the first eight complete banks, for a total of 2 Kbytes.

Figure 6-6 and Figure 6-7 show 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 Bank Select Register. For details on the Access RAM, see **Section 6.3.2 "Access Bank"**.

### 6.3.1 BANK SELECT REGISTER

Large areas of data memory require an efficient addressing scheme to make it possible for rapid access to any address. 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 four Most Significant bits of a location's address. The instruction itself includes the eight Least Significant bits. Only the four lower bits of the BSR are implemented (BSR<3:0>). The upper four bits are unused, always read as '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 eight 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 6-7.

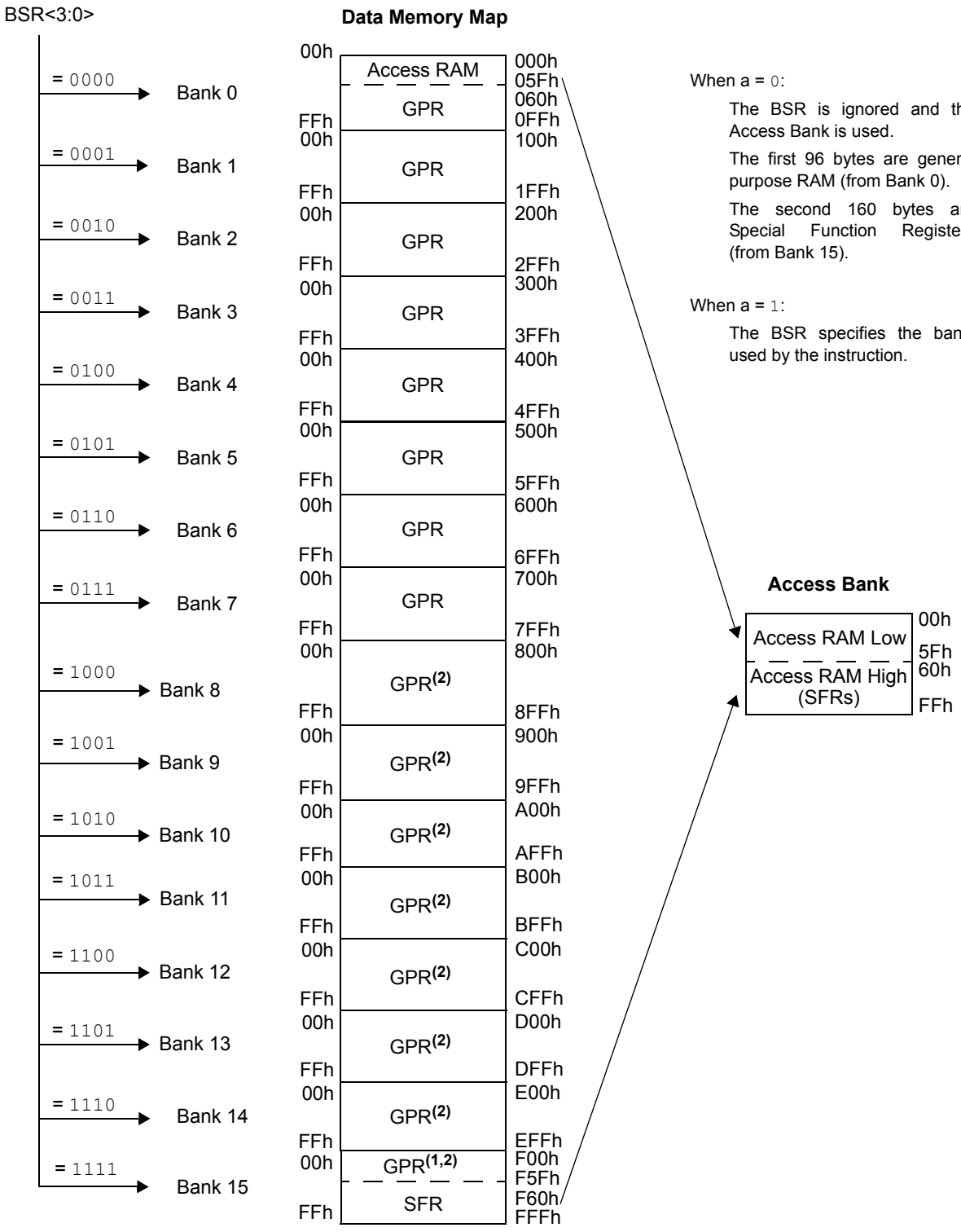
Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an eight-bit address of F9h, while the BSR is 0Fh, will end up resetting 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 6-6 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. When this instruction executes, it ignores the BSR completely. 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.


© 2009-2018 Microchip Technology Inc.

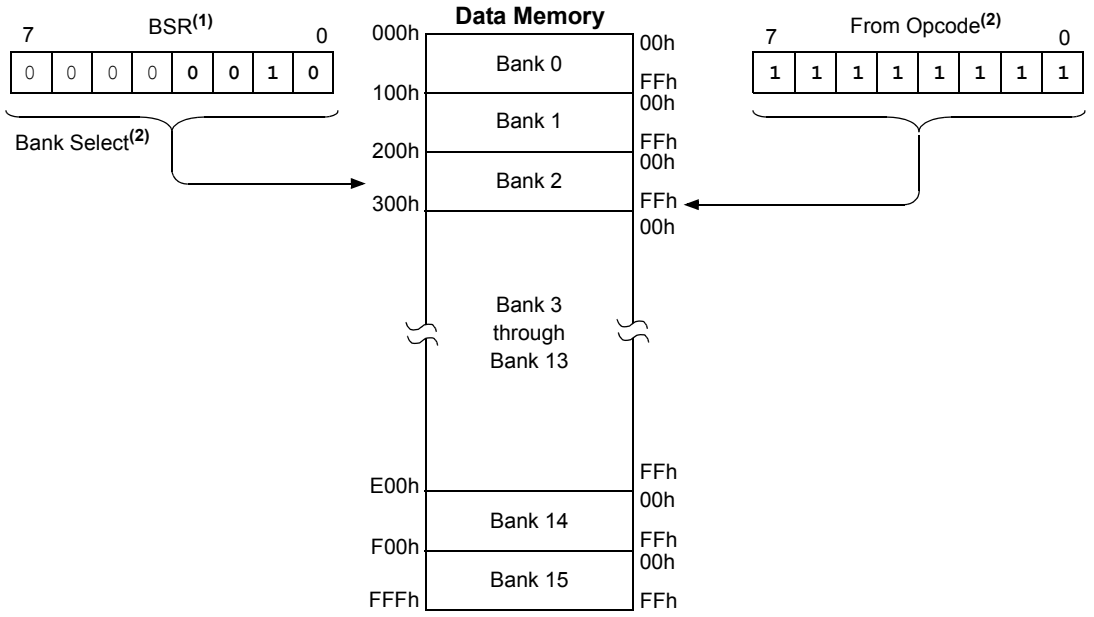
**FIGURE 6-6: DATA MEMORY MAP FOR PIC18FX5K22 AND PIC18FX7K22 DEVICES**




**Note 1:** Addresses, F16h 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:** These addresses are unused for devices with 32 Kbytes of program memory (PIC18FX5K22). For those devices, read these addresses at 00h.

**FIGURE 6-7: 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 ensure that the correct bank is selected. If not, 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 eight-bit address (Figure 6-6).

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. In that case, 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.


© 2009-2018 Microchip Technology Inc.

### 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 all of Bank 15 (F00h to FFFh) and the top part of Bank 14 (EF4h to EFFh).

A list of these registers is given in Table 6-1 and Table 6-2.

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.

**TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F87K22 FAMILY**

| Addr. | Name | Addr. | Name | Addr. | Name | Addr. | Name | Addr. | Name | Addr. | Name(4) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| FFFh | TOSU | FDFh | INDF2(1) | FBFh | ECCP1AS | F9Fh | IPR1 | F7Fh | EECON1 | F5Fh | RTCCFG |
| FFEh | TOSH | FDEh | POSTINC2(1) | FBEh | ECCP1DEL | F9Eh | PIR1 | F7Eh | EECON2 | F5Eh | RTCCAL |
| FFDh | TOSL | FDDh | POSTDEC2(1) | FBDh | CCPR1H | F9Dh | PIE1 | F7Dh | TMR5H | F5Dh | RTCVALH |
| FFCh | STKPTR | FDCh | PREINC2(1) | FBCh | CCPR1L | F9Ch | PSTR1CON | F7Ch | TMR5L | F5Ch | RTCVALL |
| FFBh | PCLATU | FDBh | PLUSW2(1) | FBBh | CCP1CON | F9Bh | OSCTUNE | F7Bh | T5CON | F5Bh | ALRMCFG |
| FFAh | PCLATH | FDAh | FSR2H | FBAh | PIR5 | F9Ah | TRISJ(2) | F7Ah | T5GCON | F5Ah | ALRMRPT |
| FF9h | PCL | FD9h | FSR2L | FB9h | PIE5 | F99h | TRISH(2) | F79h | CCPR4H | F59h | ALRMVALH |
| FF8h | TBLPTRU | FD8h | STATUS | FB8h | IPR4 | F98h | TRISG | F78h | CCPR4L | F58h | ALRMVALL |
| FF7h | TBLPTRH | FD7h | TMR0H | FB7h | PIR4 | F97h | TRISF | F77h | CCP4CON | F57h | CTMUCONH |
| FF6h | TBLPTRL | FD6h | TMR0L | FB6h | PIE4 | F96h | TRISE | F76h | CCPR5H | F56h | CTMUCONL |
| FF5h | TABLAT | FD5h | T0CON | FB5h | CVRCON | F95h | TRISD | F75h | CCPR5L | F55h | CTMUICONH |
| FF4h | PRODH | FD4h | SPBRGH1 | FB4h | CMSTAT | F94h | TRISC | F74h | CCP5CON | F54h | CM1CON |
| FF3h | PRODL | FD3h | OSCCON | FB3h | TMR3H | F93h | TRISB | F73h | CCPR6H | F53h | PADCFG1 |
| FF2h | INTCON | FD2h | IPR5 | FB2h | TMR3L | F92h | TRISA | F72h | CCPR6L | F52h | ECCP2AS |
| FF1h | INTCON2 | FD1h | WDTCON | FB1h | T3CON | F91h | LATJ(2) | F71h | CCP6CON | F51h | ECCP2DEL |
| FF0h | INTCON3 | FD0h | RCON | FB0h | T3GCON | F90h | LATH(2) | F70h | CCPR7H | F50h | CCPR2H |
| FEFh | INDF0(1) | FCFh | TMR1H | FAFh | SPBRG1 | F8Fh | LATG | F6Fh | CCPR7L | F4Fh | CCPR2L |
| FEEh | POSTINC0(1) | FCEh | TMR1L | FAEh | RCREG1 | F8Eh | LATF | F6Eh | CCP7CON | F4Eh | CCP2CON |
| FEDh | POSTDEC0(1) | FCDh | T1CON | FADh | TXREG1 | F8Dh | LATE | F6Dh | TMR4 | F4Dh | ECCP3AS |
| FECh | PREINC0(1) | FCCh | TMR2 | FACh | TXSTA1 | F8Ch | LATD | F6Ch | PR4 | F4Ch | ECCP3DEL |
| FEBh | PLUSW0(1) | FCBh | PR2 | FABh | RCSTA1 | F8Bh | LATC | F6Bh | T4CON | F4Bh | CCPR3H |
| FEAh | FSR0H | FCAh | T2CON | FAAh | T1GCON | F8Ah | LATB | F6Ah | SSP2BUF | F4Ah | CCPR3L |
| FE9h | FSR0L | FC9h | SSP1BUF | FA9h | IPR6 | F89h | LATA | F69h | SSP2ADD | F49h | CCP3CON |
| FE8h | WREG | FC8h | SSP1ADD | FA8h | HLVDCON | F88h | PORTJ(2) | F68h | SSP2STAT | F48h | CCPR8H |
| FE7h | INDF1(1) | FC7h | SSP1STAT | FA7h | PSPCON | F87h | PORTH(2) | F67h | SSP2CON1 | F47h | CCPR8L |
| FE6h | POSTINC1(1) | FC6h | SSP1CON1 | FA6h | PIR6 | F86h | PORTG | F66h | SSP2CON2 | F46h | CCP8CON |
| FE5h | POSTDEC1(1) | FC5h | SSP1CON2 | FA5h | IPR3 | F85h | PORTF | F65h | BAUDCON1 | F45h | CCPR9H(3) |
| FE4h | PREINC1(1) | FC4h | ADRESH | FA4h | PIR3 | F84h | PORTE | F64h | OSCCON2 | F44h | CCPR9L(3) |
| FE3h | PLUSW1(1) | FC3h | ADRESL | FA3h | PIE3 | F83h | PORTD | F63h | EEADRH | F43h | CCP9CON(3) |
| FE2h | FSR1H | FC2h | ADCON0 | FA2h | IPR2 | F82h | PORTC | F62h | EEADR | F42h | CCPR10H(3) |
| FE1h | FSR1L | FC1h | ADCON1 | FA1h | PIR2 | F81h | PORTB | F61h | EEDATA | F41h | CCPR10L(3) |
| FE0h | BSR | FC0h | ADCON2 | FA0h | PIE2 | F80h | PORTA | F60h | PIE6 | F40h | CCP10CON(3) |

**Note 1:** This is not a physical register.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as ‘0’.
**3:** This register is not available on devices with a program memory of 32 Kbytes (PIC18FX5K22).
**4:** Addresses, F16h through F5Fh, are also used by SFRs, but are not part of the Access RAM. To access these registers, users must always load the proper BSR value.

### TABLE 6-1: SPECIAL FUNCTION REGISTER MAP FOR PIC18F87K22 FAMILY (CONTINUED)

| Addr. | Name | Addr. | Name | Addr. | Name | Addr. | Name |
|---|---|---|---|---|---|---|---|
| F3Fh | TMR7H(3) | F32h | TMR12(3) | F25h | ANCON0 | F18h | PMD1 |
| F3Eh | TMR7L(3) | F31h | PR12(3) | F24h | ANCON1 | F17h | PMD2 |
| F3Dh | T7CON(3) | F30h | T12CON(3) | F23h | ANCON2 | F16h | PMD3 |
| F3Ch | T7GCON(3) | F2Fh | CM2CON | F22h | RCSTA2 | | |
| F3Bh | TMR6 | F2Eh | CM3CON | F21h | TXSTA2 | | |
| F3Ah | PR6 | F2Dh | CCPTMRS0 | F20h | BAUDCON2 | | |
| F39H | T6CON | F2Ch | CCPTMRS1 | F1Fh | SPBRGH2 | | |
| F38h | TMR8 | F2Bh | CCPTMRS2 | F1Eh | SPBRG2 | | |
| F37h | PR8 | F2Ah | REFOCON | F1Dh | RCREG2 | | |
| F36h | T8CON | F29H | ODCON1 | F1Ch | TXREG2 | | |
| F35h | TMR10(3) | F28h | ODCON2 | F1Bh | PSTR2CON | | |
| F34h | PR10(3) | F27h | ODCON3 | F1Ah | PSTR3CON | | |
| F33h | T10CON(3) | F26h | MEMCON(3) | F19h | PMD0 | | |

**Note 1:** This is not a physical register.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.
**3:** This register is not available on devices with a program memory of 32 Kbytes (PIC18FX5K22).
**4:** Addresses, F16h through F5Fh, are also used by SFRs, but are not part of the Access RAM. To access these registers, users must always load the proper BSR value.

### TABLE 6-2: PIC18F87K22 FAMILY REGISTER FILE SUMMARY

| Address | File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---|---|---|---|---|---|---|---|---|---|---|
| FFFh | TOSU | — | — | — | Top-of-Stack Upper Byte (TOS<20:16>) | | | | | ---0 0000 |
| FFEh | TOSH | Top-of-Stack High Byte (TOS<15:8>) | | | | | | | | 0000 0000 |
| FFDh | TOSL | Top-of-Stack Low Byte (TOS<7:0>) | | | | | | | | 0000 0000 |
| FFCh | STKPTR | STKFUL | STKUNF | — | Return Stack Pointer | | | | | uu-0 0000 |
| FFBh | PCLATU | — | — | — | Holding Register for PC<20:16> | | | | | ---0 0000 |
| FFAh | PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000 0000 |
| FF9h | PCL | PC Low Byte (PC<7:0>) | | | | | | | | 0000 0000 |
| FF8h | TBLPTRU | — | — | bit 21 | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | | --00 0000 |
| FF7h | TBLPTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | | 0000 0000 |
| FF6h | TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | | 0000 0000 |
| FF5h | TABLAT | Program Memory Table Latch | | | | | | | | 0000 0000 |
| FF4h | PRODH | Product Register High Byte | | | | | | | | xxxx xxxx |
| FF3h | PRODL | Product Register Low Byte | | | | | | | | xxxx xxxx |
| FF2h | INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF | 0000 000x |
| FF1h | INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | INTEDG3 | TMR0IP | INT3IP | RBIP | 1111 1111 |
| FF0h | INTCON3 | INT2IP | INT1IP | INT3IE | INT2IE | INT1IE | INT3IF | INT2IF | INT1IF | 1100 0000 |
| FEFh | INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) | | | | | | | | ---- ---- |
| FEEh | POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | ---- ---- |
| FEDh | POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) | | | | | | | | ---- ---- |
| FECh | PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) | | | | | | | | ---- ---- |
| FEBh | PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W | | | | | | | | ---- ---- |
| FEAh | FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0 High | | | | ---- 0000 |
| FE9h | FSR0L | Indirect Data Memory Address Pointer 0 Low Byte | | | | | | | | xxxx xxxx |
| FE8h | WREG | Working Register | | | | | | | | xxxx xxxx |
| FE7h | INDF1 | Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register) | | | | | | | | ---- ---- |

**Note 1:** This bit is available when Master Clear is disabled (MCLRE = 0). When MCLRE is set, the bit is unimplemented.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.
**3:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).


© 2009-2018 Microchip Technology Inc.

**TABLE 6-2: PIC18F87K22 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

| Address | File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---|---|---|---|---|---|---|---|---|---|---|
| FE6h | POSTINC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | ---- ---- |
| FE5h | POSTDEC1 | Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register) | | | | | | | | ---- ---- |
| FE4h | PREINC1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) | | | | | | | | ---- ---- |
| FE3h | PLUSW1 | Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by W | | | | | | | | ---- ---- |
| FE2h | FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1 High | | | | ---- xxxx |
| FE1h | FSR1L | Indirect Data Memory Address Pointer 1 Low Byte | | | | | | | | xxxx xxxx |
| FE0h | BSR | — | — | — | — | Bank Select Register | | | | ---- 0000 |
| FDFh | INDF2 | Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register) | | | | | | | | ---- ---- |
| FDEh | POSTINC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | ---- ---- |
| FDDh | POSTDEC2 | Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register) | | | | | | | | ---- ---- |
| FDCh | PREINC2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) | | | | | | | | ---- ---- |
| FDBh | PLUSW2 | Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by W | | | | | | | | ---- ---- |
| FDAh | FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2 High | | | | ---- xxxx |
| FD9h | FSR2L | Indirect Data Memory Address Pointer 2 Low Byte | | | | | | | | xxxx xxxx |
| FD8h | STATUS | — | — | — | N | OV | Z | DC | C | ---x xxxx |
| FD7h | TMR0H | Timer0 Register High Byte | | | | | | | | 0000 0000 |
| FD6h | TMR0L | Timer0 Register Low Byte | | | | | | | | xxxx xxxx |
| FD5h | T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | TOPS2 | TOPS1 | TOPS0 | 1111 1111 |
| FD4h | SPBRGH1 | USART1 Baud Rate Generator High Byte | | | | | | | | 0000 0000 |
| FD3h | OSCCON | IDLEN | IRCF2 | IRCF1 | IRCF0 | OSTS | HFIOFS | SCS1 | SCS0 | 0110 q000 |
| FD2h | IPR5 | TMR7GIP(3) | TMR12IP(3) | TMR10IP(3) | TMR8IP | TMR7IP(3) | TMR6IP | TMR5IP | TMR4IP | 1111 1111 |
| FD1h | WDTCON | REGSLP | — | ULPLVL | SRETEN | — | ULPEN | ULPSINK | SWDTEN | 0-x0 -000 |
| FD0h | RCON | IPEN | SBOREN | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ | 0111 11qq |
| FCFh | TMR1H | Timer1 Register High Byte | | | | | | | | xxxx xxxx |
| FCEh | TMR1L | Timer1 Register Low Byte | | | | | | | | xxxx xxxx |
| FCDh | T1CON | TMR1CS1 | TMR1CS0 | T1CKPS1 | T1CKPS0 | SOSCEN | $\overline{T1SYNC}$ | RD16 | TMR1ON | 0000 0000 |
| FCCh | TMR2 | Timer2 Register | | | | | | | | 0000 0000 |
| FCBh | PR2 | Timer2 Period Register | | | | | | | | 1111 1111 |
| FCAh | T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 | -000 0000 |
| FC9h | SSP1BUF | MSSP Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx |
| FC8h | SSP1ADD | MSSP Address Register in $I^2C$ Slave Mode. SSP1 Baud Rate Reload Register in $I^2C$ Master Mode. | | | | | | | | 0000 0000 |
| FC7h | SSP1STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF | 0000 0000 |
| FC6h | SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 |
| FC5h | SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0000 0000 |
| FC4h | ADRESH | A/D Result Register High Byte | | | | | | | | xxxx xxxx |
| FC3h | ADRESL | A/D Result Register Low Byte | | | | | | | | xxxx xxxx |
| FC2h | ADCON0 | — | CHS4 | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{DONE}$ | ADON | -000 0000 |
| FC1h | ADCON1 | TRIGSEL1 | TRIGSEL0 | VCFG1 | VCFG0 | VNCFG | CHSN2 | CHSN1 | CHSN0 | 0000 0000 |
| FC0h | ADCON2 | ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 | 0-00 0000 |
| FBFh | ECCP1AS | ECCP1ASE | ECCP1AS2 | ECCP1AS1 | ECCP1AS0 | PSS1AC1 | PSS1AC0 | PSS1BD1 | PSS1BD0 | 0000 0000 |
| FBEh | ECCP1DEL | P1RSEN | P1DC6 | P1DC5 | P1DC4 | P1DC3 | P1DC2 | P1DC1 | P1DC0 | 0000 0000 |
| FBDh | CCPR1H | Capture/Compare/PWM Register1 High Byte | | | | | | | | xxxx xxxx |
| FBCh | CCPR1L | Capture/Compare/PWM Register1 Low Byte | | | | | | | | xxxx xxxx |
| FBBh | CCP1CON | P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 | 0000 0000 |
| FBAh | PIR5 | TMR7GIF(3) | TMR12IF(3) | TMR10IF(3) | TMR8IF | TMR7IF(3) | TMR6IF | TMR5IF | TMR4IF | 0000 0000 |
| FB9h | PIE5 | TMR7GIE(3) | TMR12IE(3) | TMR10IE(3) | TMR8IE | TMR7IE(3) | TMR6IE | TMR5IE | TMR4IE | 0000 0000 |
| FB8h | IPR4 | CCP10IP(3) | CCP9IP(3) | CCP8IP | CCP7IP | CCP6IP | CCP5IP | CCP4IP | CCP3IP | 1111 1111 |
| FB7h | PIR4 | CCP10IF(3) | CCP9IF(3) | CCP8IF | CCP7IF | CCP6IF | CCP5IF | CCP4IF | CCP3IF | 0000 0000 |

**Note 1:** This bit is available when Master Clear is disabled (MCLRE = 0). When MCLRE is set, the bit is unimplemented.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.
**3:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

**TABLE 6-2: PIC18F87K22 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

| Address | File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---|---|---|---|---|---|---|---|---|---|---|
| FB6h | PIE4 | CCP10IE(3) | CCP9IE(3) | CCP8IE | CCP7IE | CCP6IE | CCP5IE | CCP4IE | CCP3IE | 0000 0000 |
| FB5h | CVRCON | CVREN | CVROE | CVRSS | CVR4 | CVR3 | CVR2 | CVR1 | CVR0 | 0000 0000 |
| FB4h | CMSTAT | CMP3OUT | CMP2OUT | CMP1OUT | — | — | — | — | — | xxx- ---- |
| FB3h | TMR3H | Timer3 Register High Byte | | | | | | | | xxxx xxxx |
| FB2h | TMR3L | Timer3 Register Low Byte | | | | | | | | xxxx xxxx |
| FB1h | T3CON | TMR3CS1 | TMR3CS0 | T3CKPS1 | T3CKPS0 | SOSCEN | $\overline{\text{T3SYNC}}$ | RD16 | TMR3ON | 0000 0000 |
| FB0h | T3GCON | TMR3GE | T3GPOL | T3GTM | T3GSPM | $\overline{\text{T3GGO}}$/$\overline{\text{T3DONE}}$ | T3GVAL | T3GSS1 | T3GSS0 | 0000 0x00 |
| FAFh | SPBRG1 | USART1 Baud Rate Generator | | | | | | | | 0000 0000 |
| FAEh | RCREG1 | USART1 Receive Register | | | | | | | | 0000 0000 |
| FADh | TXREG1 | USART1 Transmit Register | | | | | | | | xxxx xxxx |
| FACh | TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 |
| FABh | RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x |
| FAAh | T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | $\overline{\text{T1GGO}}$/$\overline{\text{T1DONE}}$ | T1GVAL | T1GSS1 | T1GSS0 | 0000 0x00 |
| FA9h | IPR6 | — | — | — | EEIP | — | CMP3IP | CMP2IP | CMP1IP | ---1 -111 |
| FA8h | HLVDCON | VDIRMAG | BGVST | IRVST | HLVDEN | HLVDL3 | HLVDL2 | HLVDL1 | HLVDL0 | 0000 0000 |
| FA7h | PSPCON | IBF | OBF | IBOV | PSPMODE | — | — | — | — | 0000 ---- |
| FA6h | PIR6 | — | — | — | EEIF | — | CMP3IF | CMP2IF | CMP1IF | ---0 -000 |
| FA5h | IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP | 1-11 1111 |
| FA4h | PIR3 | TMR5GiF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF | 0-00 0000 |
| FA3h | PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE | 0-00 0000 |
| FA2h | IPR2 | OSCFIP | — | SSP2IP | BCL2IP | BCL1IP | HLVDIP | TMR3IP | TMR3GIP | 1-11 1111 |
| FA1h | PIR2 | OSCFIF | — | SSP2IF | BCL2IF | BCL1IF | HLVDIF | TMR3IF | TMR3GIF | 0-00 0000 |
| FA0h | PIE2 | OSCFIE | — | SSP2IE | BCL2IE | BCL1IE | HLVDIE | TMR3IE | TMR3GIE | 0-00 0000 |
| F9Fh | IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP | 1111 1111 |
| F9Eh | PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF | 0000 0000 |
| F9Dh | PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE | 0000 0000 |
| F9Ch | PSTR1CON | CMPL1 | CMPL0 | — | STRSYNC | STRD | STRC | STRB | STRA | 00-0 0001 |
| F9Bh | OSCTUNE | INTSRC | PLLEN | TUN5 | TUN4 | TUN3 | TUN2 | TUN1 | TUN0 | 0000 0000 |
| F9Ah | TRISJ(2) | TRISJ7 | TRISJ6 | TRISJ5 | TRISJ4 | TRISJ3 | TRISJ2 | TRISJ1 | TRISJ0 | 1111 1111 |
| F99h | TRISH(2) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 | 1111 1111 |
| F98h | TRISG | — | — | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 | ---1 1111 |
| F97h | TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — | 1111 111- |
| F96h | TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | — | 1111 111- |
| F95h | TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 1111 1111 |
| F94h | TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 1111 1111 |
| F93h | TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 1111 1111 |
| F92h | TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 1111 1111 |
| F91h | LATJ(2) | LATJ7 | LATJ6 | LATJ5 | LATJ4 | LATJ3 | LATJ2 | LATJ1 | LATJ0 | xxxx xxxx |
| F90h | LATH(2) | LATH7 | LATH6 | LATH5 | LATH4 | LATH3 | LATH2 | LATH1 | LATH0 | xxxx xxxx |
| F8Fh | LATG | — | — | — | LATG4 | LATG3 | LATG2 | LATG1 | LATG0 | ---x xxxx |
| F8Eh | LATF | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | — | xxxx xxx- |
| F8Dh | LATE | LATE7 | LATE6 | LATE5 | LATE4 | LATE3 | LATE2 | LATE1 | LATE0 | xxxx xxxx |
| F8Ch | LATD | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | xxxx xxxx |
| F8Bh | LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | xxxx xxxx |
| F8Ah | LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | xxxx xxxx |
| F89h | LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | xxxx xxxx |
| F88h | PORTJ(2) | RJ7 | RJ6 | RJ5 | RJ4 | RJ3 | RJ2 | RJ1 | RJ0 | xxxx xxxx |
| F87h | PORTH(2) | RH7 | RH6 | RH5 | RH4 | RH3 | RH2 | RH1 | RH0 | xxxx xxxx |

**Note 1:** This bit is available when Master Clear is disabled (MCLRE = 0). When MCLRE is set, the bit is unimplemented.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.
**3:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).


© 2009-2018 Microchip Technology Inc.

**TABLE 6-2: PIC18F87K22 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

| Address | File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---|---|---|---|---|---|---|---|---|---|---|
| F86h | PORTG | — | — | RG5(1) | RG4 | RG3 | RG2 | RG1 | RG0 | --xx xxxx |
| F85h | PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | — | xxxx xxx- |
| F84h | PORTE | RE7 | RE6 | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 | xxxx xxxx |
| F83h | PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxx xxxx |
| F82h | PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | xxxx xxxx |
| F81h | PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | xxxx xxxx |
| F80h | PORTA | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | xxxx xxxx |
| F7Fh | EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD | xx-0 x000 |
| F7Eh | EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | | ---- ---- |
| F7Dh | TMR5H | Timer5 Register High Byte | | | | | | | | xxxx xxxx |
| F7Ch | TMR5L | Timer5 Register Low Byte | | | | | | | | xxxx xxxx |
| F7Bh | T5CON | TMR5CS1 | TMR5CS0 | T5CKPS1 | T5CKPS0 | SOSCEN | $\overline{\text{T5SYNC}}$ | RD16 | TMR5ON | 0000 0000 |
| F7Ah | T5GCON | TMR5GE | T5GPOL | T5GTM | T5GSPM | T5GGO/$\overline{\text{T5DONE}}$ | T5GVAL | T5GSS1 | T5GSS0 | 0000 0x00 |
| F79h | CCPR4H | Capture/Compare/PWM Register 4 High Byte | | | | | | | | xxxx xxxx |
| F78h | CCPR4L | Capture/Compare/PWM Register 4 Low Byte | | | | | | | | xxxx xxxx |
| F77h | CCP4CON | — | — | DC4B1 | DC4B0 | CCP4M3 | CCP4M2 | CCP4M1 | CCP4M0 | --00 0000 |
| F76h | CCPR5H | Capture/Compare/PWM Register 5 High Byte | | | | | | | | xxxx xxxx |
| F75h | CCPR5L | Capture/Compare/PWM Register 5 Low Byte | | | | | | | | xxxx xxxx |
| F74h | CCP5CON | — | — | DC5B1 | DC5B0 | CCP5M3 | CCP5M2 | CCP5M1 | CCP5M0 | --00 0000 |
| F73h | CCPR6H | Capture/Compare/PWM Register 6 High Byte | | | | | | | | xxxx xxxx |
| F72h | CCPR6L | Capture/Compare/PWM Register 6 Low Byte | | | | | | | | xxxx xxxx |
| F71h | CCP6CON | — | — | DC6B1 | DC6B0 | CCP6M3 | CCP6M2 | CCP6M1 | CCP6M0 | --00 0000 |
| F70h | CCPR7H | Capture/Compare/PWM Register 7 High Byte | | | | | | | | xxxx xxxx |
| F6Fh | CCPR7L | Capture/Compare/PWM Register 7 Low Byte | | | | | | | | xxxx xxxx |
| F6Eh | CCP7CON | — | — | DC7B1 | DC7B0 | CCP7M3 | CCP7M2 | CCP7M1 | CCP7M0 | --00 0000 |
| F6Dh | TMR4 | Timer4 Register | | | | | | | | xxxx xxxx |
| F6Ch | PR4 | Timer4 Period Register | | | | | | | | 1111 1111 |
| F6Bh | T4CON | — | T4OUTPS3 | T4OUTPS2 | T4OUTPS1 | T4OUTPS0 | TMR4ON | T4CKPS1 | T4CKPS0 | -111 1111 |
| F6Ah | SSP2BUF | MSSP Receive Buffer/Transmit Register | | | | | | | | xxxx xxxx |
| F69h | SSP2ADD | MSSP Address Register in $I^2C$ Slave Mode. MSSP1 Baud Rate Reload Register in $I^2C$ Master Mode. | | | | | | | | 0000 0000 |
| F68h | SSP2STAT | SMP | CKE | D/$\overline{\text{A}}$ | P | S | R/$\overline{\text{W}}$ | UA | BF | 0000 0000 |
| F67h | SSP2CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 | 0000 0000 |
| F66h | SSP2CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0100 0000 |
| F65h | BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 0100 0-00 |
| F64h | OSCCON2 | — | SOSCRUN | — | — | SOSCGO | — | MFIOFS | MFIOSEL | -0-- 0-x0 |
| F63h | EEADRH | EEPROM Address Register High Byte | | | | | | | | 0000 0000 |
| F62h | EEADR | EEPROM Address Register Low Byte | | | | | | | | 0000 0000 |
| F61h | EEDATA | EEPROM Data Register | | | | | | | | 0000 0000 |
| F60h | PIE6 | — | — | — | EEIE | — | CMP3IE | CMP2IE | CMP1IE | ---0 -000 |
| F5Fh | RTCCFG | RTCEN | — | RTCWREN | RTCSYNC | HALFSEC | RTCOE | RTCPTR1 | RTCPTR0 | 0-00 0000 |
| F5Eh | RTCCAL | CAL7 | CAL6 | CAL5 | CAL4 | CAL3 | CAL2 | CAL1 | CAL0 | 0000 0000 |
| F5Dh | RTCVALH | RTCC Value High Register Window Based on RTCPTR<1:0> | | | | | | | | xxxx xxxx |

**Note 1:** This bit is available when Master Clear is disabled (MCLRE = 0). When MCLRE is set, the bit is unimplemented.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.
**3:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

**TABLE 6-2: PIC18F87K22 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

| Address | File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---|---|---|---|---|---|---|---|---|---|---|
| F5Ch | RTCVALL | RTCC Value Low Register Window Based on RTCPTR<1:0> | | | | | | | | 0000 0000 |
| F5Bh | ALRMCFG | ALRMEN | CHIME | AMASK3 | AMASK2 | AMASK1 | AMASK0 | ALRMPTR1 | ALRMPTR0 | 0000 0000 |
| F5Ah | ALRMRPT | ARPT7 | ARPT6 | ARPT5 | ARPT4 | ARPT3 | ARPT2 | ARPT1 | ARPT0 | 0000 0000 |
| F59h | ALRMVALH | Alarm Value High Register Window Based on APTR<1:0> | | | | | | | | xxxx xxxx |
| F58h | ALRMVALL | Alarm Value Low Register Window Based on APTR<1:0> | | | | | | | | xxxx xxxx |
| F57h | CTMUCONH | CTMUEN | — | CTMUSIDL | TGEN | EDGEN | EDGSEQEN | IDISSEN | CTTRIG | 0-00 0000 |
| F56h | CTMUCONL | EDG2POL | EDG2SEL1 | EDG2SEL0 | EDG1POL | EDG1SEL1 | EDG1SEL0 | EDG2STAT | EDG1STAT | 0000 0000 |
| F55h | CTMUICONH | ITRIM5 | ITRIM4 | ITRIM3 | ITRIM2 | ITRIM1 | ITRIM0 | IRNG1 | IRNG0 | 0000 0000 |
| F54h | CM1CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 | 0001 1111 |
| F53h | PADCFG1 | RDPU | REPU | RJPU(2) | — | — | RTSEC-SEL1 | RTSEC-SEL0 | — | 000- -00- |
| F52h | ECCP2AS | ECCP2ASE | ECCP2AS2 | ECCP2AS1 | ECCP2AS0 | PSS2AC1 | PSS2AC0 | PSS2BD1 | PSS2BD0 | 0000 0000 |
| F51h | ECCP2DEL | P2RSEN | P2DC6 | P2DC5 | P2DC4 | P2DC3 | P2DC2 | P2DC1 | P2DC0 | 0000 0000 |
| F50h | CCPR2H | Capture/Compare/PWM Register 2 High Byte | | | | | | | | xxxx xxxx |
| F4Fh | CCPR2L | Capture/Compare/PWM Register 2 Low Byte | | | | | | | | xxxx xxxx |
| F4Eh | CCP2CON | P2M1 | P2M0 | DC2B1 | DC2B0 | CCP2M3 | CCP2M2 | CCP2M1 | CCP2M0 | 0000 0000 |
| F4Dh | ECCP3AS | ECCP3ASE | ECCP3AS2 | ECCP3AS1 | ECCP3AS0 | PSS3AC1 | PSS3AC0 | PSS3BD1 | PSS3BD0 | 0000 0000 |
| F4Ch | ECCP3DEL | P3RSEN | P3DC6 | P3DC5 | P3DC4 | P3DC3 | P3DC2 | P3DC1 | P3DC0 | 0000 0000 |
| F4Bh | CCPR3H | Capture/Compare/PWM Register 3 High Byte | | | | | | | | xxxx xxxx |
| F4Ah | CCPR3L | Capture/Compare/PWM Register 3 Low Byte | | | | | | | | xxxx xxxx |
| F49h | CCP3CON | P3M1 | P3M0 | DC3B1 | DC3B0 | CCP3M3 | CCP3M2 | CCP3M1 | CCP3M0 | 0000 0000 |
| F48h | CCPR8H | Capture/Compare/PWM Register 8 High Byte | | | | | | | | xxxx xxxx |
| F47h | CCPR8L | Capture/Compare/PWM Register 8 Low Byte | | | | | | | | xxxx xxxx |
| F46h | CCP8CON | — | — | DC8B1 | DC8B0 | CCP8M3 | CCP8M2 | CCP8M1 | CCP8M0 | --00 0000 |
| F45h | CCPR9H(3) | Capture/Compare/PWM Register 9 High Byte | | | | | | | | xxxx xxxx |
| F44h | CCPR9L(3) | Capture/Compare/PWM Register 9 Low Byte | | | | | | | | xxxx xxxx |
| F43h | CCP9CON(3) | — | — | DC9B1 | DC9B0 | CCP9M3 | CCP9M2 | CCP9M1 | CCP9M0 | --00 0000 |
| F42h | CCPR10H(3) | Capture/Compare/PWM Register 10 High Byte | | | | | | | | xxxx xxxx |
| F41h | CCPR10L(3) | Capture/Compare/PWM Register 10 Low Byte | | | | | | | | xxxx xxxx |
| F40h | CCP10CON(3) | — | — | DC10B1 | DC10B0 | CCP10M3 | CCP10M2 | CCP10M1 | CCP10M0 | --00 0000 |
| F3Fh | TMR7H(3) | Timer7 Register High Byte | | | | | | | | xxxx xxxx |
| F3Eh | TMR7L(3) | Timer7 Register Low Byte | | | | | | | | 0000 0000 |
| F3Dh | T7CON(3) | TMR7CS1 | TMR7CS0 | T7CKPS1 | T7CKPS0 | SOSCEN | $\overline{\text{T7SYNC}}$ | RD16 | TMR7ON | 0000 0000 |
| F3Ch | T7GCON(3) | TMR7GE | T7GPOL | T7GTM | T7GSPM | $\overline{\text{T7GGO}}$/$\overline{\text{T7DONE}}$ | T7GVAL | T7GSS1 | T7GSS0 | 0000 0x00 |
| F3Bh | TMR6 | Timer6 Register | | | | | | | | 0000 0000 |
| F3Ah | PR6 | Timer6 Period Register | | | | | | | | 1111 1111 |
| F39h | T6CON | — | T6OUTPS3 | T6OUTPS2 | T6OUTPS1 | T6OUTPS0 | TMR6ON | T6CKPS1 | T6CKPS0 | -000 0000 |
| F38h | TMR8 | Timer8 Register | | | | | | | | 0000 0000 |
| F37h | PR8 | Timer8 Period Register | | | | | | | | 1111 1111 |
| F36h | T8CON | — | T8OUTPS3 | T8OUTPS2 | T8OUTPS1 | T8OUTPS0 | TMR8ON | T8CKPS1 | T8CKPS0 | -000 0000 |
| F35h | TMR10(3) | TMR10 Register | | | | | | | | 0000 0000 |
| F34h | PR10(3) | Timer10 Period Register | | | | | | | | 1111 1111 |
| F33h | T10CON(3) | — | T10OUTPS3 | T10OUTPS2 | T10OUTPS1 | T10OUTPS0 | TMR10ON | T10CKPS1 | T10CKPS0 | -000 0000 |
| F32h | TMR12(3) | TMR12 Register | | | | | | | | 0000 0000 |
| F31h | PR12(3) | Timer12 Period Register | | | | | | | | 1111 1111 |
| F30h | T12CON(3) | — | T12OUTPS3 | T12OUTPS2 | T12OUTPS1 | T12OUTPS0 | TMR12ON | T12CKPS1 | T12CKPS0 | -000 0000 |
| F2Fh | CM2CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 | 0001 1111 |

**Note 1:** This bit is available when Master Clear is disabled (MCLRE = 0). When MCLRE is set, the bit is unimplemented.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.
**3:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).


© 2009-2018 Microchip Technology Inc.

**TABLE 6-2: PIC18F87K22 FAMILY REGISTER FILE SUMMARY (CONTINUED)**

| Address | File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---|---|---|---|---|---|---|---|---|---|---|
| F2Eh | CM3CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 | 0001 1111 |
| F2Dh | CCPTMRS0 | C3TSEL1 | C3TSEL0 | C2TSEL2 | C2TSEL1 | C2TSEL0 | C1TSEL2 | C1TSEL1 | C1TSEL0 | 0000 0000 |
| F2Ch | CCPTMRS1 | C7TSEL1 | C7TSEL0 | — | C6TSEL0 | — | C5TSEL0 | C4TSEL1 | C4TSEL0 | 00-0 -000 |
| F2Bh | CCPTMRS2 | — | — | — | C10TSEL0(3) | — | C9TSEL0(3) | C8TSEL1 | C8TSEL0 | ---0 -000 |
| F2Ah | REFOCON | ROON | — | ROSSLP | ROSEL | RODIV3 | RODIV2 | RODIV1 | RODIV0 | 0-00 0000 |
| F29h | ODCON1 | SSP1OD | CCP2OD | CCP1OD | — | — | — | — | SSP2OD | 000- ---0 |
| F28h | ODCON2 | CCP10OD(3) | CCP9OD(3) | CCP8OD | CCP7OD | CCP6OD | CCP5OD | CCP4OD | CCP3OD | 0000 0000 |
| F27h | ODCON3 | U2OD | U1OD | — | — | — | — | — | CTMUDS | 00-- ---0 |
| F26h | MEMCON(2) | EBDIS | — | WAIT1 | WAIT0 | — | — | WM1 | WM0 | 0-00 --00 |
| F25h | ANCON0 | ANSEL7 | ANSEL6 | ANSEL5 | ANSEL4 | ANSEL3 | ANSEL2 | ANSEL1 | ANSEL0 | 1111 1111 |
| F24h | ANCON1 | ANSEL15 | ANSEL14 | ANSEL13 | ANSEL12 | ANSEL11 | ANSEL10 | ANSEL9 | ANSEL8 | 1111 1111 |
| F23h | ANCON2 | ANSEL23 | ANSEL22 | ANSEL21 | ANSEL20 | ANSEL19 | ANSEL18 | ANSEL17 | ANSEL16 | 1111 1111 |
| F22h | RCSTA2 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x |
| F21h | TXSTA2 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 0000 0010 |
| F20h | BAUDCON2 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN | 0100 0-00 |
| F1Fh | SPBRGH2 | USART2 Baud Rate Generator High Byte | | | | | | | | 0000 0000 |
| F1Eh | SPBRG2 | USART2 Baud Rate Generator | | | | | | | | 0000 0000 |
| F1Dh | RCREG2 | Receive Data FIFO | | | | | | | | 0000 0000 |
| F1Ch | TXREG2 | Transmit Data FIFO | | | | | | | | xxxx xxxx |
| F1Bh | PSTR2CON | CMPL1 | CMPL0 | — | STRSYNC | STRD | STRC | STRB | STRA | 00-0 0001 |
| F1Ah | PSTR3CON | CMPL1 | CMPL0 | — | STRSYNC | STRD | STRC | STRB | STRA | 00-0 0001 |
| F19h | PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD | 0000 0000 |
| F18h | PMD1 | PSPMD | CTMUMD | RTCCMD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | EMBMD | 0000 0000 |
| F17h | PMD2 | TMR10MD(3) | TMR8MD | TMR7MD(3) | TMR6MD | TMR5MD | CMP3MD | CMP2MD | CMP1MD | 0000 0000 |
| F16h | PMD3 | CCP10MD(3) | CCP9MD(3) | CCP8MD | CCP7MD | CCP6MD | CCP5MD | CCP4MD | TMR12MD(3) | 0000 0000 |

**Note 1:** This bit is available when Master Clear is disabled (MCLRE = 0). When MCLRE is set, the bit is unimplemented.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.
**3:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

### 6.3.5 STATUS REGISTER

The STATUS register, shown in Register 6-2, 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, 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 be 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 29-2 and Table 29-3.

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

**REGISTER 6-2: 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 seven-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.


© 2009-2018 Microchip Technology Inc.

## 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. For more information, see **Section 6.6 "Data Memory and the Extended Instruction Set"**.

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). For details on this mode's operation, see **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 of this mode include SLEEP, RESET and DAW.

Other instructions work in a similar way, but require an additional explicit argument in the opcode. This method is known as the Literal Addressing mode because the instructions require some literal value as an argument. Examples of this 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 the instruction's data source as either a register address in one of the banks of data RAM (see **Section 6.3.3 "General Purpose Register File"**) or a location in the Access Bank (see **Section 6.3.2 "Access Bank"**).

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, 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

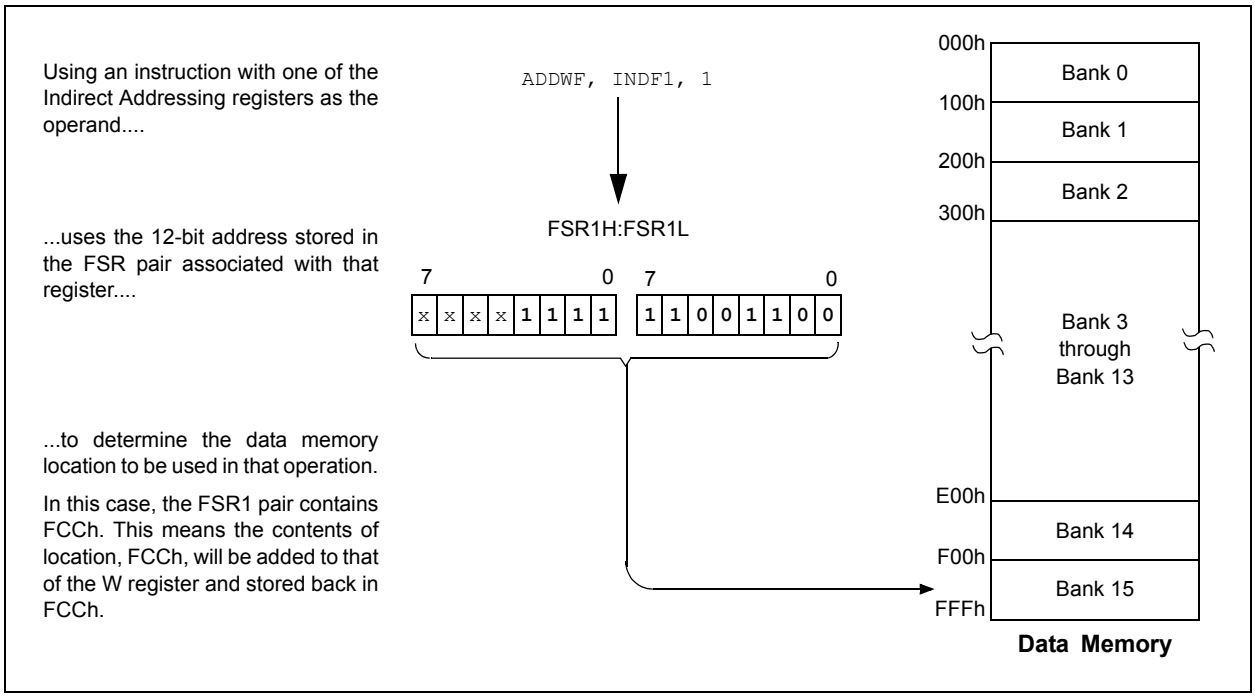
At the core of Indirect Addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers: FSRnH and FSRnL. The four upper bits of the FSRnH register are not used, so each FSR pair holds a 12-bit value. This represents a value that can address the entire range of 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. The operands 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 6-8: INDIRECT ADDRESSING**





© 2009-2018 Microchip Technology Inc.

#### 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.

These operands are:

- POSTDEC – Accesses the FSR value, then automatically decrements it by ‘1’ afterwards
- POSTINC – Accesses the FSR value, then automatically increments it by ‘1’ afterwards
- PREINC – Increments the FSR value by ‘1’, then uses it in the operation
- PLUSW – Adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the new value in the operation

In this context, accessing an INDF register uses the value in the FSR registers without changing them. Similarly, accessing a PLUSW register gives the FSR value, offset by the value in the W register, with neither value 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. Rollovers of the FSRnL register, from FFh to 00h, carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (for example, Z, N and OV bits).

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 the FSR0H:FSR0L registers contain 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, however, 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, so 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. Using the Access Bank for many of the core PIC18 instructions introduces 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 – that is, most bit-oriented and byte-oriented instructions – can invoke a form of Indexed Addressing using an offset specified in the instruction. This special addressing mode is known as Indexed Addressing with Literal Offset or the Indexed Literal Offset mode.

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

- Use of the Access Bank (‘a’ = 0)
- A file address argument that 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 = 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 6-9.

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 29.2.1 “Extended Instruction Syntax”**.


© 2009-2018 Microchip Technology Inc.

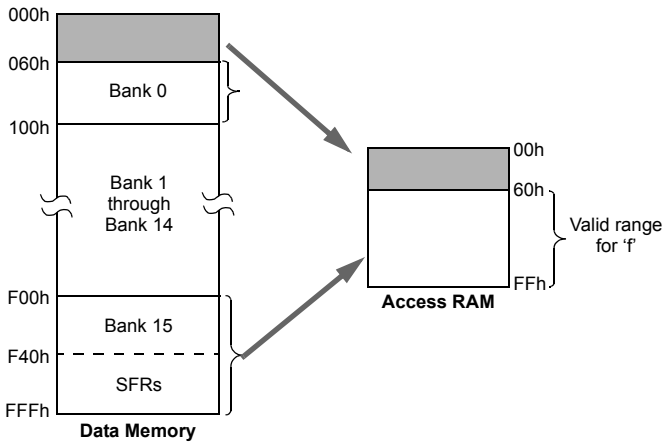
**FIGURE 6-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)**

**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.

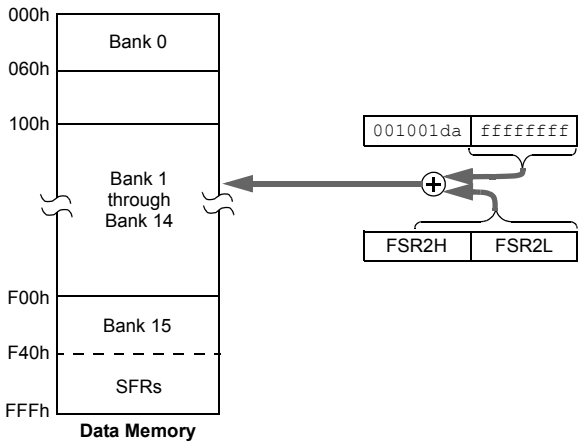
Locations below 060h are not available in this addressing mode.




**When a = 0 and f ≤ 5Fh:**

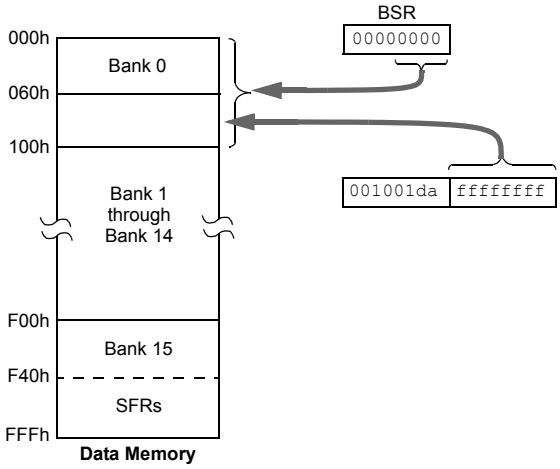
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

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.

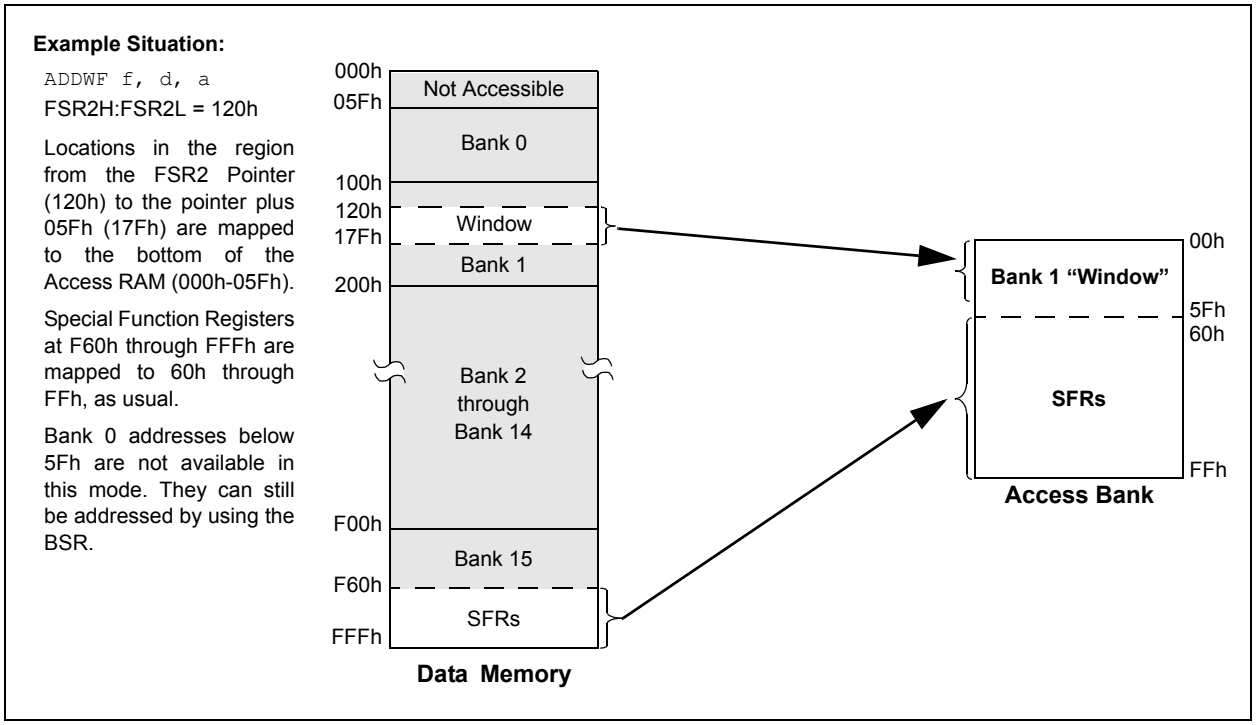
The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described. (See **Section 6.3.2 "Access Bank"**.) An example of Access Bank remapping in this addressing mode is shown in Figure 6-10.

Remapping the Access Bank applies *only* to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit = `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 6-10: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING**





© 2009-2018 Microchip Technology Inc.

## 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. For execution of a write to, or erasure of, program memory:

- Memory of 32 Kbytes and 64 Kbytes (PIC18FX-5K22 and PIC18FX6K22 devices) – Blocks of 64 bytes
- Memory of 128 Kbytes (PIC18FX7K22 devices) – Blocks of 128 bytes

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)

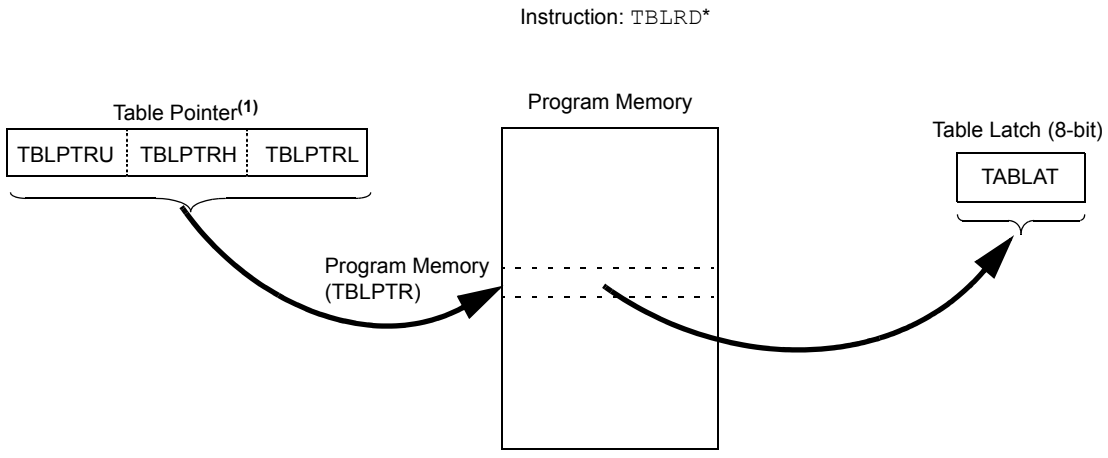
The program memory space is 16 bits wide, while the data RAM space is eight bits wide. Table reads and table writes move data between these two memory spaces through an eight-bit register (TABLAT).

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.

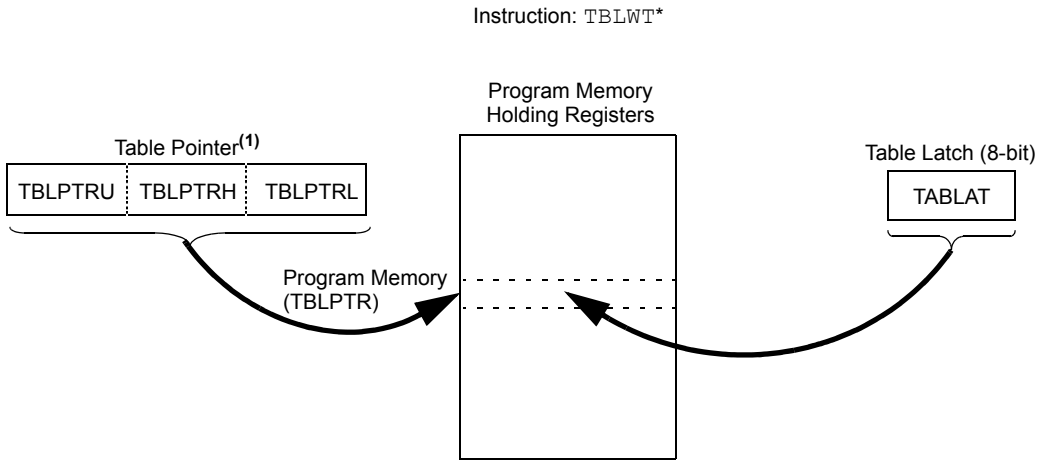
Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word-aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions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

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




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

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




**Note 1:** The 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, not a physical register, is used exclusively in the memory write and erase sequences. Reading EECON2 will read all '0's.

The EEPGD control bit determines if the access is a program or data EEPROM memory access. When clear, any subsequent operations operate on the data EEPROM memory. When set, any subsequent operations operate on the program memory.

The CFGS control bit determines if the access is to the Configuration/Calibration registers or to program memory/data EEPROM memory. When set, subsequent operations operate on Configuration registers regardless of EEPGD (see **Section 28.0 "Special Features of the CPU"**). When clear, memory selection access is determined by EEPGD.

The FREE bit, when set, allows 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, allows 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.

| **Note:** | The EEIF interrupt flag bit (PIR6<4>) is set when the write is complete. It must be cleared in software. |
|---|---|


© 2009-2018 Microchip Technology Inc.

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

| R/W-x | R/W-x | U-0 | R/W-0 | R/W-x | R/W-0 | R/S-0 | R/S-0 |
|---|---|---|---|---|---|---|---|
| EEPGD | CFGS | — | FREE | WRERR(1) | WREN | WR | RD |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | S = Settable 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 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
- 1 = Access Flash program memory
- 0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
- 1 = Access Configuration registers
- 0 = Access Flash program or data EEPROM memory

bit 5 **Unimplemented:** Read as '0'

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

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit(1)
- 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/Data EEPROM Write Enable bit
- 1 = Allows write cycles to Flash program/data EEPROM
- 0 = Inhibits write cycles to Flash program/data EEPROM

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

bit 0 **RD:** Read Control bit
- 1 = Initiates an EEPROM read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. The RD bit cannot be set when EEPGD = 1 or CFGS = 1.)
- 0 = Does not initiate an EEPROM read

**Note 1:** When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.

### 7.2.2 TABLAT – TABLE LATCH REGISTER

The Table Latch (TABLAT) is an eight-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 TBLPTR – TABLE POINTER REGISTER

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 and only affect the low-order 21 bits.

### 7.2.4 TABLE POINTER BOUNDARIES

The 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 the TABLAT.

When a `TBLWT` is executed, the six LSbs of the Table Pointer register (TBLPTR<5:0>) 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 16 MSbs of the TBLPTR (TBLPTR<21:6>) determine which program memory block of 64 bytes is written to. For more detail, see **Section 7.5 "Writing to Flash Program Memory"**.

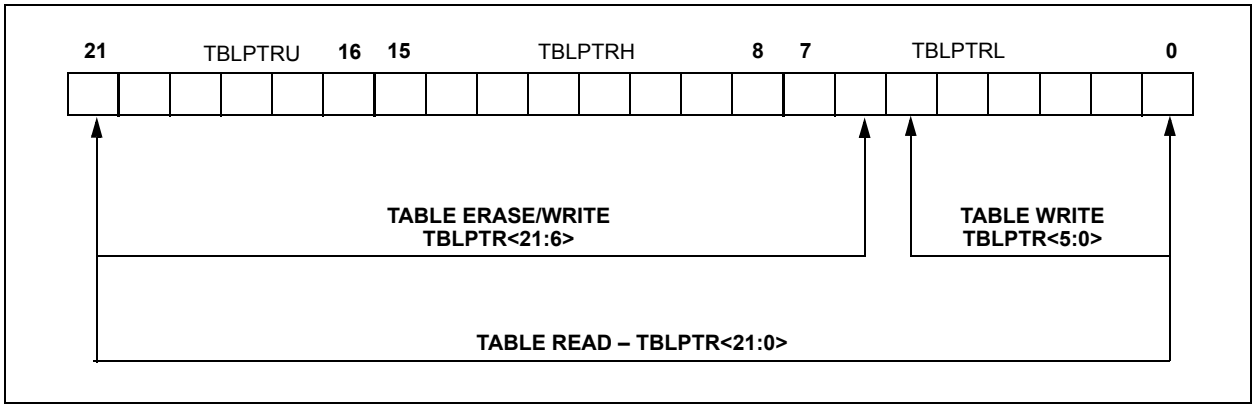
When an erase of program memory is executed, the 16 MSbs of the Table Pointer register (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

Figure 7-3 describes the relevant boundaries of the 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**





© 2009-2018 Microchip Technology Inc.

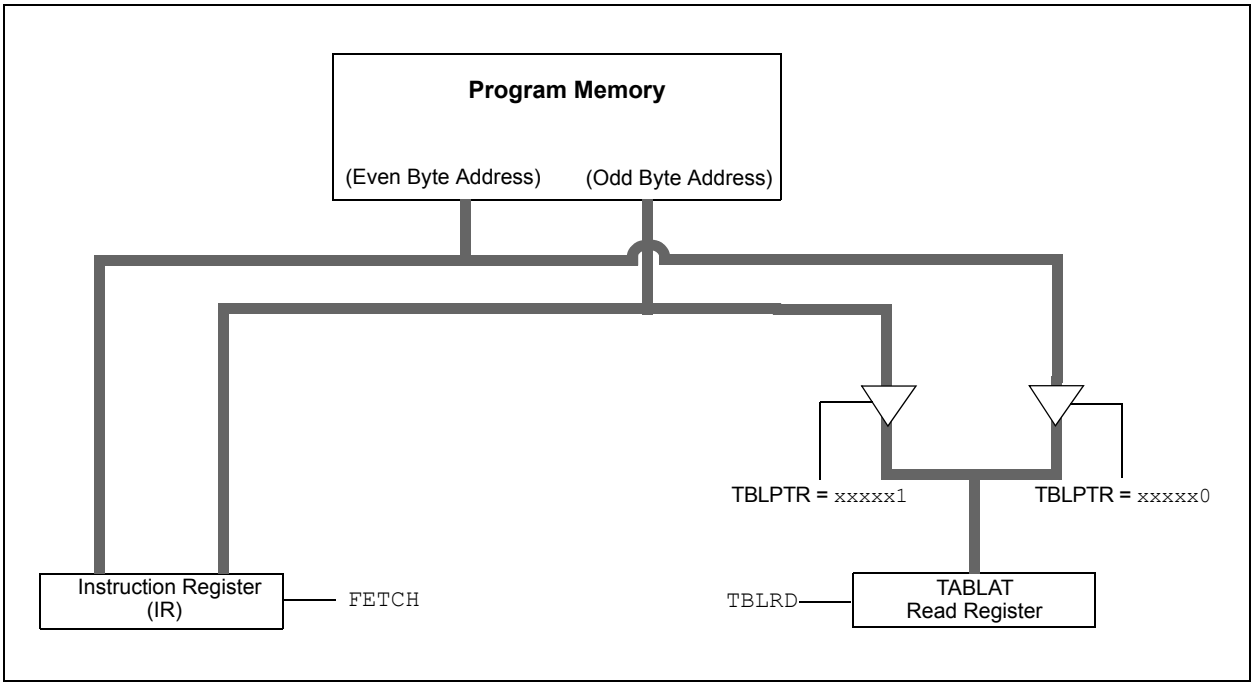
## 7.3 Reading the Flash Program Memory

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, the 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**

```
            BCF     EECON1, CFGS            ; point to Flash program memory
            BSF     EECON1, EEPGD           ; access Flash program memory
            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
            MOVF    WORD_ODD
```

## 7.4 Erasing Flash Program Memory

The erase blocks are:

- PIC18FX5K22 and PIC18FX6K22 – 32 words or 64 bytes
- PIC18FX7K22 – 64 words or 128 bytes

Word erase in the Flash array is not supported.

When initiating an erase sequence from the microcontroller itself, a block of 64 or 128 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. The TBLPTR<5:0> bits are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. 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 the Table Pointer register with the address of the row to be erased.
2. Set the EECON1 register for the erase operation:
   - Set the EEPGD bit to point to program memory
   - Clear the CFGS bit to access program memory
   - Set the WREN bit to enable writes
   - Set the FREE bit to enable the erase
3. Disable the interrupts.
4. Write 0x55 to EECON2.
5. Write 0xAA to EECON2.
6. Set the WR bit.

   This begins the row erase cycle.

   The CPU will stall for the duration of the erase for TIW. (See Parameter D133A.)
7. 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, EEPGD           ; point to Flash program memory
                    BCF     EECON1, CFGS            ; access Flash program memory
                    BSF     EECON1, WREN            ; enable write to memory
                    BSF     EECON1, FREE            ; enable Row Erase operation
                    BCF     INTCON, GIE             ; disable interrupts
    Required        MOVLW   0x55
    Sequence        MOVWF   EECON2                  ; write 55h
                    MOVLW   0xAA
                    MOVWF   EECON2                  ; write 0AAh
                    BSF     EECON1, WR              ; start erase (CPU stall)
                    BSF     INTCON, GIE             ; re-enable interrupts
```


© 2009-2018 Microchip Technology Inc.

## 7.5 Writing to Flash Program Memory

The programming blocks are:

- PIC18FX5K22 and PIC18FX6K22 – 32 words or 64 bytes
- PIC18FX7K22 – 64 words or 128 bytes

Word or byte programming is not supported.

Table writes are used internally to load the holding registers needed to program the Flash memory. The number of holding registers used for programming by the table writes are:

- PIC18FX5K22 and PIC18FX6K22 – 64
- PIC18FX7K22 – 128

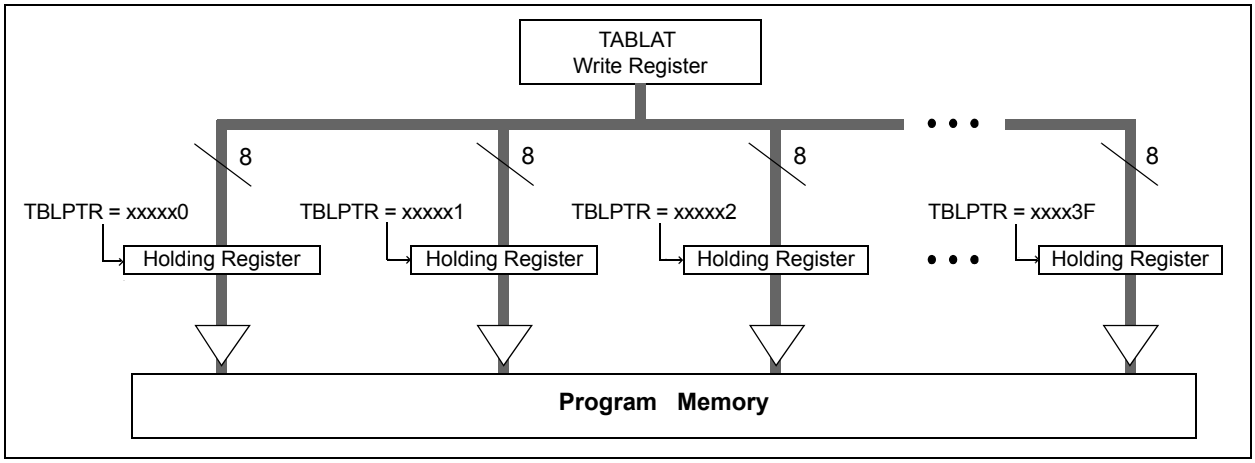
Since the Table Latch (TABLAT) is only a single byte, the `TBLWT` instruction may need to be executed 64 times for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. At the end of updating the 64 or 128 holding 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 is terminated by the internal programming timer.

The EEPROM 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:** | The default value of the holding registers on device Resets, and after write operations, is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all 64 or 128 holding registers before executing a write operation. |
|---|---|

**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 the 64 or 128 bytes into RAM.
2. Update the data values in RAM as necessary.
3. Load the Table Pointer register with the address being erased.
4. Execute the row erase procedure.
5. Load the Table Pointer register with the address of the first byte being written.
6. Write the 64 or 128 bytes into the holding registers with auto-increment.
7. Set the EECON1 register for the write operation:
   - Set the EEPGD bit to point to program memory
   - Clear the CFGS bit to access program memory
   - Set the WREN to enable byte writes
8. Disable the interrupts.
9. Write 0x55 to EECON2.
10. Write 0xAA to EECON2.
11. Set the WR bit. This will begin the write cycle. The CPU will stall for duration of the write for TIW (see Parameter D133A).
12. Re-enable the interrupts.
13. Verify the memory (table read).

An example of the required code is shown in Example 7-3.

| | |
|---|---|
| **Note:** | Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the 64 or 128 bytes in the holding register. |

| | |
|---|---|
| **Note:** | Self-write execution to Flash and EEPROM memory cannot be done while running in LP Oscillator mode (Low-Power mode). Therefore, executing a self-write will put the device into High-Power mode. |


© 2009-2018 Microchip Technology Inc.

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

```
                MOVLW   SIZE_OF_BLOCK          ; number of bytes in erase block
                MOVWF   COUNTER
                MOVLW   BUFFER_ADDR_HIGH       ; point to buffer
                MOVWF   FSR0H
                MOVLW   BUFFER_ADDR_LOW
                MOVWF   FSR0L
                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
READ_BLOCK
                TBLRD*+                        ; read into TABLAT, and inc
                MOVF    TABLAT, W              ; get data
                MOVWF   POSTINC0               ; store data
                DECFSZ  COUNTER                ; done?
                BRA     READ_BLOCK             ; repeat
MODIFY_WORD
                MOVLW   DATA_ADDR_HIGH         ; point to buffer
                MOVWF   FSR0H
                MOVLW   DATA_ADDR_LOW
                MOVWF   FSR0L
                MOVLW   NEW_DATA_LOW           ; update buffer word
                MOVWF   POSTINC0
                MOVLW   NEW_DATA_HIGH
                MOVWF   INDF0
ERASE_BLOCK
                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
                BSF     EECON1, EEPGD          ; point to Flash program memory
                BCF     EECON1, CFGS           ; access Flash program memory
                BSF     EECON1, WREN           ; enable write to memory
                BSF     EECON1, FREE           ; enable Row Erase operation
                BCF     INTCON, GIE            ; disable interrupts
                MOVLW   0x55
Required        MOVWF   EECON2                 ; write 55h
Sequence        MOVLW   0xAA
                MOVWF   EECON2                 ; write 0AAh
                BSF     EECON1, WR             ; start erase (CPU stall)
                BSF     INTCON, GIE            ; re-enable interrupts
                TBLRD*-                        ; dummy read decrement
                MOVLW   BUFFER_ADDR_HIGH       ; point to buffer
                MOVWF   FSR0H
                MOVLW   BUFFER_ADDR_LOW
                MOVWF   FSR0L
WRITE_BUFFER_BACK
                MOVLW   SIZE_OF_BLOCK          ; 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
```

**EXAMPLE 7-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)**

```
PROGRAM_MEMORY
                BSF     EECON1, EEPGD       ; point to Flash program memory
                BCF     EECON1, CFGS        ; access Flash program memory
                BSF     EECON1, WREN        ; enable write to memory
                BCF     INTCON, GIE         ; disable interrupts
                MOVLW   0x55
   Required     MOVWF   EECON2              ; write 55h
   Sequence     MOVLW   0xAA
                MOVWF   EECON2              ; write 0AAh
                BSF     EECON1, WR          ; start program (CPU stall)
                BSF     INTCON, GIE         ; re-enable interrupts
                BCF     EECON1, WREN        ; disable write to memory
```

### 7.5.2 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.

### 7.5.3 UNEXPECTED TERMINATION OF WRITE OPERATION

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

### 7.5.4 PROTECTION AGAINST SPURIOUS WRITES

To protect against spurious writes to Flash program memory, the write initiate sequence must also be followed. See **Section 28.0 "Special Features of the CPU"** for more details.

## 7.6 Flash Program Operation During Code Protection

See **Section 28.6 "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 |
|---|---|---|---|---|---|---|---|---|
| TBLPTRU | — | — | bit 21(1) | Program Memory Table Pointer Upper Byte (TBLPTR<20:16>) | | | | |
| TBPLTRH | Program Memory Table Pointer High Byte (TBLPTR<15:8>) | | | | | | | |
| TBLPTRL | Program Memory Table Pointer Low Byte (TBLPTR<7:0>) | | | | | | | |
| TABLAT | Program Memory Table Latch | | | | | | | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD |
| IPR6 | — | — | — | EEIP | — | CMP3IP | CMP2IP | CMP1IP |
| PIR6 | — | — | — | EEIF | — | CMP3IF | CMP2IF | CMP1IF |
| PIE6 | — | — | — | EEIE | — | CMP3IE | CMP2IE | CMP1IE |

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

**Note 1:** Bit 21 of the TBLPTRU allows access to the device Configuration bits.


© 2009-2018 Microchip Technology Inc.

## 8.0 EXTERNAL MEMORY BUS

**Note:** The External Memory Bus is not implemented on 64-pin devices.

The External Memory Bus (EMB) allows the device to access external memory devices (such as Flash, EPROM or SRAM) 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: PIC18F87K22 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.

© 2009-2018 Microchip Technology Inc.

## 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 WAIT 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 WM bits select the particular operating mode used when the bus is operating in 16-Bit Data Width mode. These bits 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.

**REGISTER 8-1: MEMCON: EXTERNAL MEMORY BUS CONTROL REGISTER(1)**

| 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 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; $\overline{WRH}$ is active when TABLAT is written
- 01 = Byte Select mode: TABLAT data is copied on both MSB and LSB; $\overline{WRH}$ and ($\overline{UB}$ or $\overline{LB}$) will activate
- 00 = Byte Write mode: TABLAT data is copied on both MSB and LSB; $\overline{WRH}$ or $\overline{WRL}$ will activate

**Note 1:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.


© 2009-2018 Microchip Technology Inc.

## 8.2 Address and Data Width

The PIC18F87K22 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 ABW<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 is 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 (ABW<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 ABW 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 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 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 is 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 PADCFG1 register (PADCFG1<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 PIC18F87K22 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 the ABW 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 the 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 $\overline{CE}$, $\overline{OE}$, $\overline{WRH}$, $\overline{WRL}$, $\overline{UB}$ and $\overline{LB}$ 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 (PMP) 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 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 ($\overline{OE}$) signal will enable both bytes of program memory at once to form a 16-bit instruction word. The Chip Enable ($\overline{CE}$ signal) 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 $\overline{UB}$ or $\overline{LB}$ signals for byte selection.

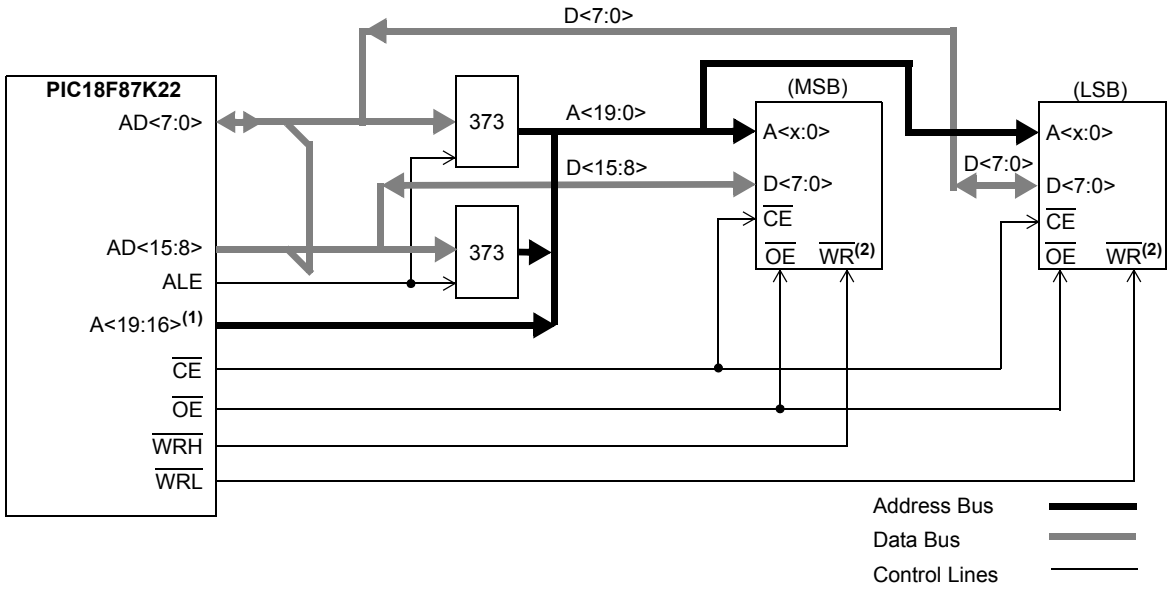

© 2009-2018 Microchip Technology Inc.

### 8.6.1 16-BIT BYTE WRITE MODE

Figure 8-1 shows an example of 16-Bit Byte Write mode for PIC18F87K22 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 WRH or 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

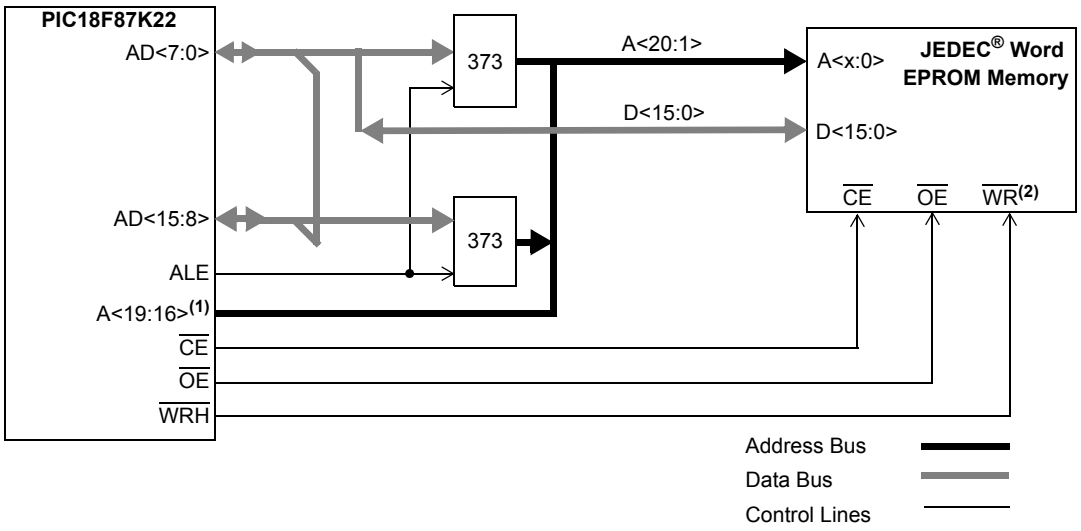
Figure 8-2 shows an example of 16-Bit Word Write mode for PIC18F87K22 family devices. This mode is used for word-wide memories, which includes some of the EPROM and Flash type memories. This mode allows opcode fetches and table reads from all forms of 16-bit memory, and table writes to any type of word-wide 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 $\overline{WRH}$ signal is strobed for each write cycle; the $\overline{WRL}$ pin is unused. The signal on the BA0 pin indicates the LSb of the TBLPTR, but it is left unconnected. Instead, the $\overline{UB}$ and $\overline{LB}$ 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"**.


© 2009-2018 Microchip Technology Inc.

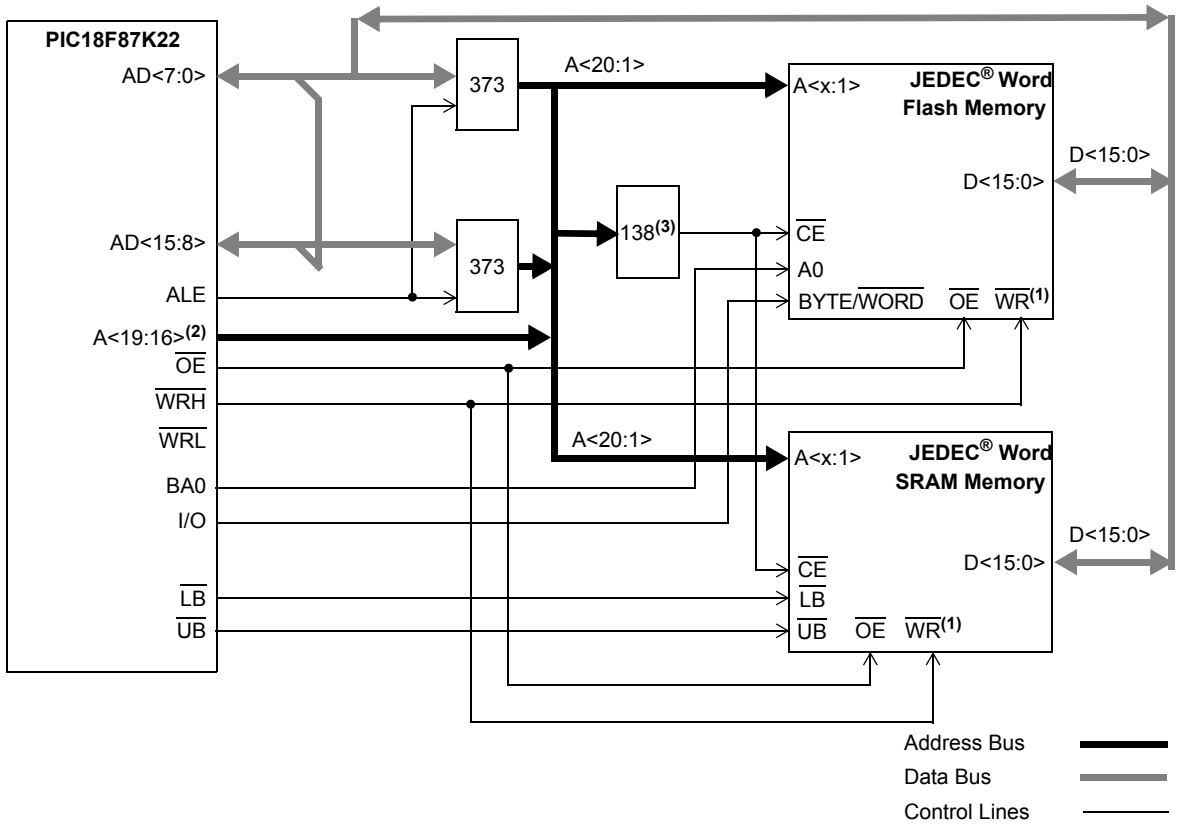
### 8.6.3 16-BIT BYTE SELECT MODE

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 $\overline{WRH}$ signal is strobed for each write cycle; the $\overline{WRL}$ pin is not used. The BA0 or $\overline{UB}$/$\overline{LB}$ 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/$\overline{WORD}$ 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 $\overline{UB}$ or $\overline{LB}$ 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)**

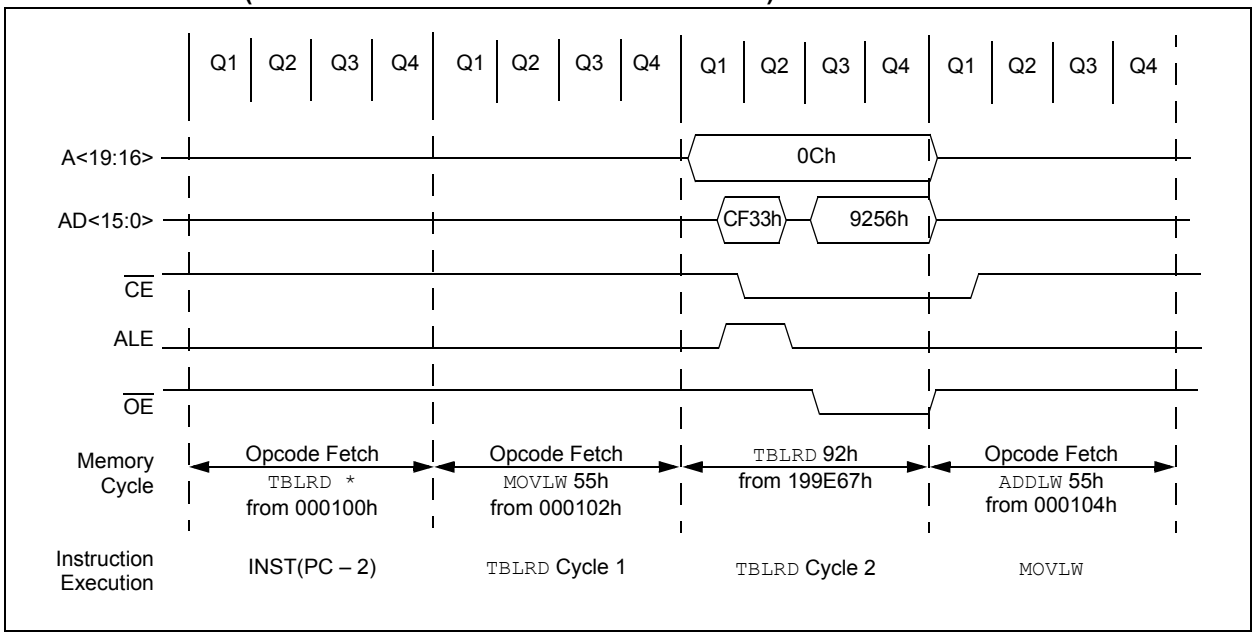
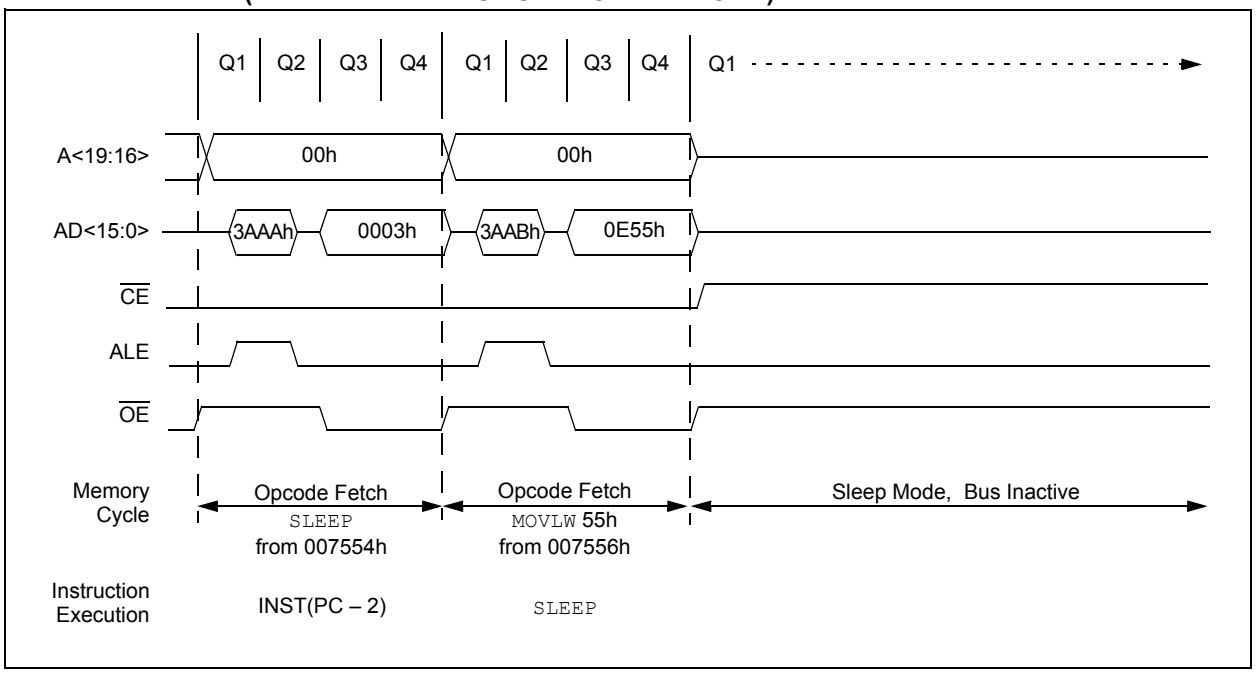



**FIGURE 8-5: EXTERNAL MEMORY BUS TIMING FOR SLEEP (EXTENDED MICROCONTROLLER MODE)**





© 2009-2018 Microchip Technology Inc.

## 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.

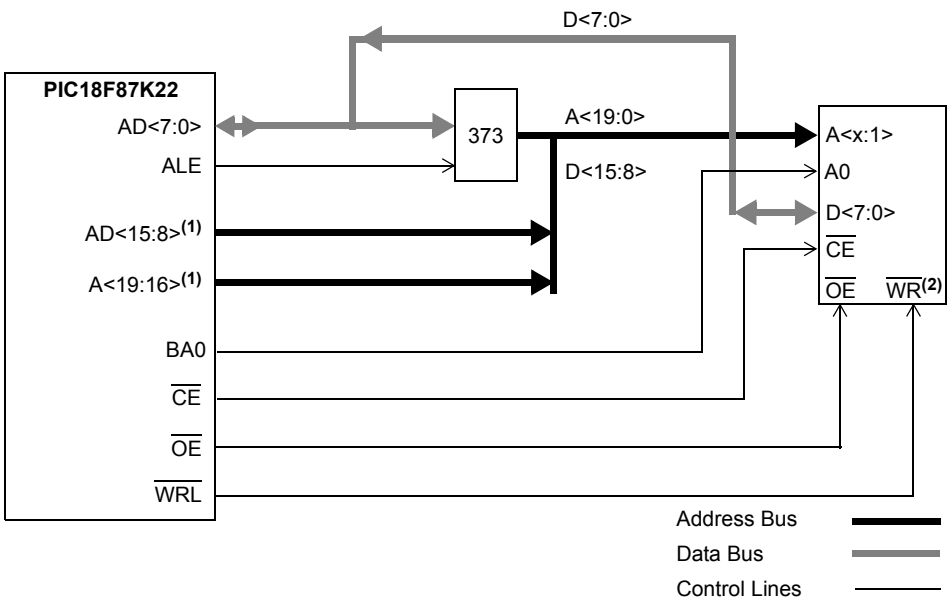
Figure 8-6 shows an example of 8-Bit Multiplexed mode for PIC18F8XK22 devices. This mode is used for a single, 8-bit memory connected for 16-bit operation. The instructions will be fetched as two 8-bit bytes on a shared data/address bus. The two bytes are sequentially fetched within one instruction cycle (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 Bus interface. The Output Enable ($\overline{OE}$) signal 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 ($\overline{CE}$) signal 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)**

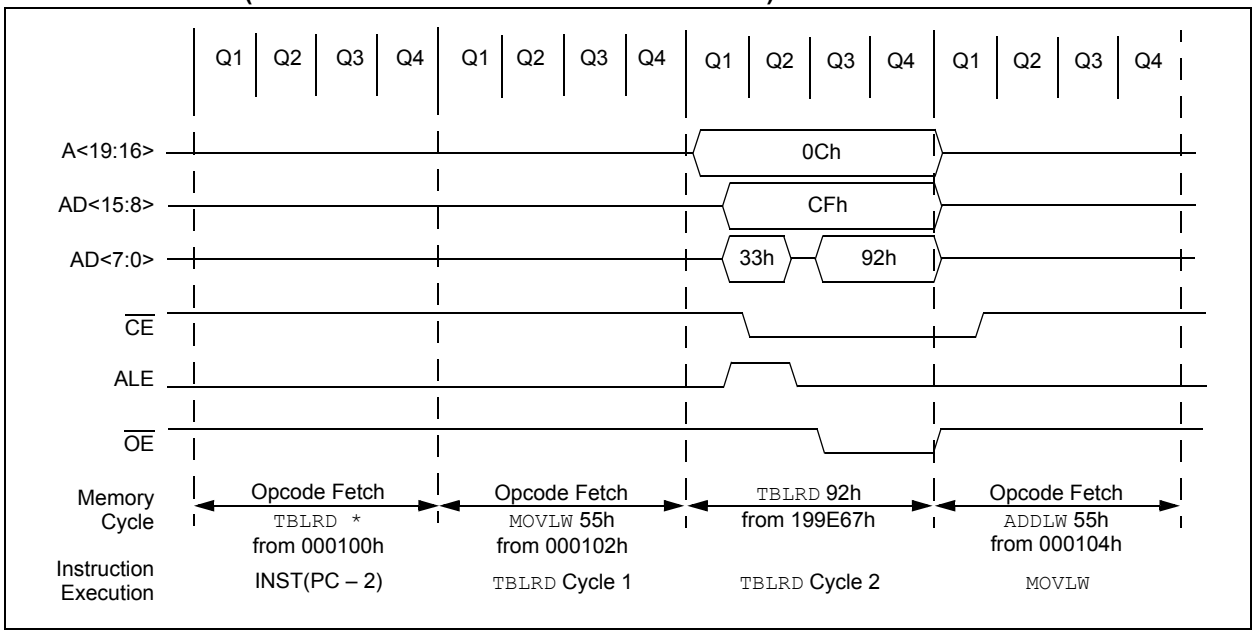
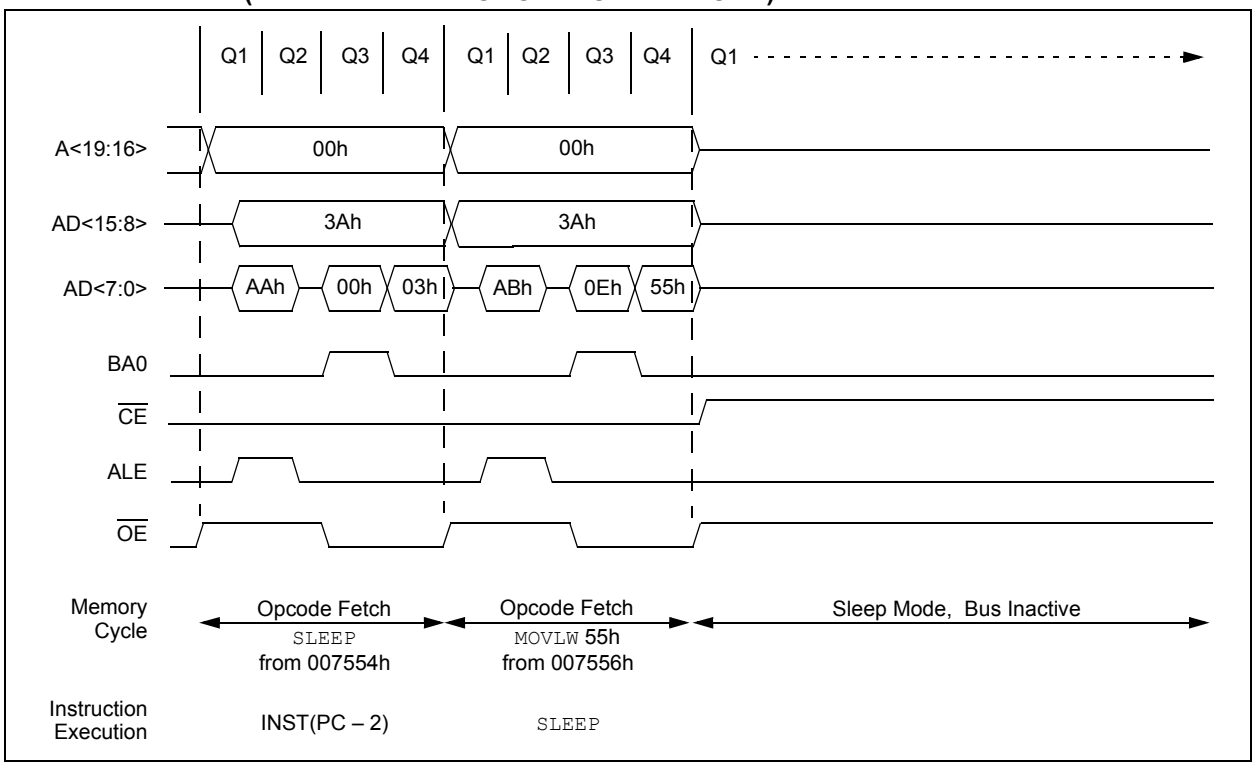



**FIGURE 8-8: EXTERNAL MEMORY BUS TIMING FOR SLEEP (EXTENDED MICROCONTROLLER MODE)**





© 2009-2018 Microchip Technology Inc.

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

**TABLE 8-3: REGISTERS ASSOCIATED WITH THE EXTERNAL MEMORY BUS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MEMCON(1) | EBDIS | — | WAIT1 | WAIT0 | — | — | WM1 | WM0 |
| PADCFG1 | RDPU | REPU | RJPU(1) | — | — | RTSECSEL1 | RTSECSEL0 | — |
| PMD1 | PSPMD | CTMUMD | RTCCMD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | EMBMD |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used during External Memory Bus access.

**Note 1:** Unimplemented in 64-pin devices (PIC18F6XK22), read as '0'.

## 9.0 DATA EEPROM MEMORY

The data EEPROM is a nonvolatile memory array, separate from the data RAM and program memory, that is used for long-term storage of program data. It is not directly mapped in either the register file or program memory space, but is indirectly addressed through the Special Function Registers (SFRs). The EEPROM is readable and writable during normal operation over the entire VDD range.

Five SFRs are used to read and write to the data EEPROM, as well as the program memory. They are:

- EECON1
- EECON2
- EEDATA
- EEADR
- EEADRH

The data EEPROM allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and the EEADRH:EEADR register pair holds the address of the EEPROM location being accessed.

The EEPROM data memory is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer; it will vary with voltage and temperature, as well as from chip-to-chip. Please refer to Parameter D122 (Table 31-1 in **Section 31.0 "Electrical Characteristics"**) for exact limits.

### 9.1 EEADR and EEADRH Registers

The EEADRH:EEADR register pair is used to address the data EEPROM for read and write operations. EEADRH holds the two MSbs of the address; the upper 6 bits are ignored. The 10-bit range of the pair can address a memory range of 1024 bytes (00h to 3FFh).

### 9.2 EECON1 and EECON2 Registers

Access to the data EEPROM is controlled by two registers: EECON1 and EECON2. These are the same registers which control access to the program memory and are used in a similar manner for the data EEPROM.

The EECON1 register (Register 9-1) is the control register for data and program memory access. Control bit, EEPGD, determines if the access will be to program memory or data EEPROM memory. When clear, operations will access the data EEPROM memory. When set, program memory is accessed.

Control bit, CFGS, determines if the access will be to the Configuration registers or to program memory/data EEPROM memory. When set, subsequent operations access Configuration registers. When CFGS is clear, the EEPGD bit selects either program Flash or data EEPROM memory.

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 WREN 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.

| | |
|---|---|
| **Note:** | The EEIF interrupt flag bit (PIR6<4>) is set when the write is complete; it must be cleared in software. |

Control bits, RD and WR, start read and erase/write operations, respectively. These bits are set by firmware and cleared by hardware at the completion of the operation.

The RD bit cannot be set when accessing program memory (EEPGD = 1). Program memory is read using table read instructions. See **Section 7.1 "Table Reads and Table Writes"** regarding table reads.

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.


© 2009-2018 Microchip Technology Inc.

**REGISTER 9-1: EECON1: DATA EEPROM CONTROL REGISTER 1**

| R/W-x | R/W-x | U-0 | R/W-0 | R/W-x | R/W-0 | R/S-0 | R/S-0 |
|---|---|---|---|---|---|---|---|
| EEPGD | CFGS | — | FREE | WRERR(1) | WREN | WR | RD |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | S = Settable 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 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
- 1 = Access Flash program memory
- 0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
- 1 = Access Configuration registers
- 0 = Access Flash program or data EEPROM memory

bit 5 **Unimplemented:** Read as '0'

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

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit(1)
- 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 has completed

bit 2 **WREN:** Flash Program/Data EEPROM Write Enable bit
- 1 = Allows write cycles to Flash program/data EEPROM
- 0 = Inhibits write cycles to Flash program/data EEPROM

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

bit 0 **RD:** Read Control bit
- 1 = Initiates an EEPROM read (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. The RD bit cannot be set when EEPGD = 1 or CFGS = 1.)
- 0 = Does not initiate an EEPROM read

**Note 1:** When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.

## 9.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADRH:EEADR register pair, clear the EEPGD control bit (EECON1<7>) and then set control bit, RD (EECON1<0>). The data is available in the EEDATA register after one cycle; therefore, it can be read after one `NOP` instruction. EEDATA will hold this value until another read operation or until it is written to by the user (during a write operation).

The basic process is shown in Example 9-1.

## 9.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADRH:EEADR register pair and the data written to the EEDATA register. The sequence in Example 9-2 must be followed to initiate the write cycle.

The write will not begin if this sequence is not exactly followed (write 0x55 to EECON2, write 0xAA to EECON2, then set WR bit) for each byte. It is strongly recommended that interrupts be disabled during this code segment.

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

After a write sequence has been initiated, EECON1, EEADRH:EEADR and EEDATA cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. The WREN bit must be set on a previous instruction. Both WR and WREN cannot be set with the same instruction.

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

## 9.5 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.

| | |
|---|---|
| **Note:** | Self-write execution to Flash and EEPROM memory cannot be done while running in LP Oscillator mode (Low-Power mode). Therefore, executing a self-write will put the device into High-Power mode. |


© 2009-2018 Microchip Technology Inc.

**EXAMPLE 9-1: DATA EEPROM READ**

```
        MOVLW   DATA_EE_ADDRH      ;
        MOVWF   EEADRH             ; Upper bits of Data Memory Address to read
        MOVLW   DATA_EE_ADDR       ;
        MOVWF   EEADR              ; Lower bits of Data Memory Address to read
        BCF     EECON1, EEPGD      ; Point to DATA memory
        BCF     EECON1, CFGS       ; Access EEPROM
        BSF     EECON1, RD         ; EEPROM Read
        NOP
        MOVF    EEDATA, W          ; W = EEDATA
```

**EXAMPLE 9-2: DATA EEPROM WRITE**

```
                MOVLW   DATA_EE_ADDRH      ;
                MOVWF   EEADRH             ; Upper bits of Data Memory Address to write
                MOVLW   DATA_EE_ADDR       ;
                MOVWF   EEADR              ; Lower bits of Data Memory Address to write
                MOVLW   DATA_EE_DATA       ;
                MOVWF   EEDATA             ; Data Memory Value to write
                BCF     EECON1, EEPGD      ; Point to DATA memory
                BCF     EECON1, CFGS       ; Access EEPROM
                BSF     EECON1, WREN       ; Enable writes

                BCF     INTCON, GIE        ; Disable Interrupts
                MOVLW   0x55               ;
Required        MOVWF   EECON2             ; Write 55h
Sequence        MOVLW   0xAA               ;
                MOVWF   EECON2             ; Write 0AAh
                BSF     EECON1, WR         ; Set WR bit to begin write
                BTFSC   EECON1, WR         ; Wait for write to complete GOTO $-2
                BSF     INTCON, GIE        ; Enable Interrupts

                                           ; User code execution
                BCF     EECON1, WREN       ; Disable writes on write complete (EEIF set)
```

## 9.6 Operation During Code-Protect

Data EEPROM memory has its own code-protect bits in the Configuration Words. External read and write operations are disabled if code protection is enabled.

The microcontroller itself can both read and write to the internal data EEPROM, regardless of the state of the code-protect Configuration bit. Refer to **Section 28.0 "Special Features of the CPU"** for additional information.

## 9.7 Protection Against Spurious Write

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been implemented. On power-up, the WREN bit is cleared. In addition, writes to the EEPROM are blocked during the Power-up Timer period (TPWRT, Parameter 33 in Table 31-14).

The write initiate sequence, and the WREN bit together, help prevent an accidental write during brown-out, power glitch or software malfunction.

## 9.8 Using the Data EEPROM

The data EEPROM is a high-endurance, byte address-able array that has been optimized for the storage of frequently changing information (e.g., program variables or other data that is updated often). Frequently changing values will typically be updated more often than Specification D124. If this is the case, an array refresh must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in Flash program memory.

A simple data EEPROM refresh routine is shown in Example 9-3.

| | |
|---|---|
| **Note:** | If data EEPROM is only used to store constants and/or data that changes often, an array refresh is likely not required. See Specification D124. |

**EXAMPLE 9-3: DATA EEPROM REFRESH ROUTINE**

```
        CLRF    EEADR             ; Start at address 0
        CLRF    EEADRH            ;
        BCF     EECON1, CFGS      ; Set for memory
        BCF     EECON1, EEPGD     ; Set for Data EEPROM
        BCF     INTCON, GIE       ; Disable interrupts
        BSF     EECON1, WREN      ; Enable writes
LOOP                              ; Loop to refresh array
        BSF     EECON1, RD        ; Read current address
        MOVLW   0x55              ;
        MOVWF   EECON2            ; Write 55h
        MOVLW   0xAA              ;
        MOVWF   EECON2            ; Write 0AAh
        BSF     EECON1, WR        ; Set WR bit to begin write
        BTFSC   EECON1, WR        ; Wait for write to complete
        BRA     $-2
        INCFSZ  EEADR, F          ; Increment address
        BRA     LOOP              ; Not zero, do it again
        INCFSZ  EEADRH, F         ; Increment the high address
        BRA     LOOP              ; Not zero, do it again

        BCF     EECON1, WREN      ; Disable writes
        BSF     INTCON, GIE       ; Enable interrupts
```


© 2009-2018 Microchip Technology Inc.

**TABLE 9-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| EEADRH | EEPROM Address Register High Byte | | | | | | | |
| EEADR | EEPROM Address Register Low Byte | | | | | | | |
| EEDATA | EEPROM Data Register | | | | | | | |
| EECON2 | EEPROM Control Register 2 (not a physical register) | | | | | | | |
| EECON1 | EEPGD | CFGS | — | FREE | WRERR | WREN | WR | RD |
| IPR6 | — | — | — | EEIP | — | CMP3IP | CMP2IP | CMP1IP |
| PIR6 | — | — | — | EEIF | — | CMP3IF | CMP2IF | CMP1IF |
| PIE6 | — | — | — | EEIE | — | CMP3IE | CMP2IE | CMP1IE |

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

## 10.0 8 x 8 HARDWARE MULTIPLIER

### 10.1 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 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 10-1.

### 10.2 Operation

Example 10-1 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 10-2 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 10-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE**

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

**EXAMPLE 10-2: 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 10-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS**

| Routine | Multiply Method | Program Memory (Words) | Cycles (Max) | Time | | | |
|---|---|---|---|---|---|---|---|
| | | | | @ 64 MHz | @ 48 MHz | @ 10 MHz | @ 4 MHz |
| 8 x 8 Unsigned | Without Hardware Multiply | 13 | 69 | 4.3 µs | 5.7 µs | 27.6 µs | 69 µs |
| | Hardware Multiply | 1 | 1 | 62.5 ns | 83.3 ns | 400 ns | 1 µs |
| 8 x 8 Signed | Without Hardware Multiply | 33 | 91 | 5.6 µs | 7.5 µs | 36.4 µs | 91 µs |
| | Hardware Multiply | 6 | 6 | 375 ns | 500 ns | 2.4 µs | 6 µs |
| 16 x 16 Unsigned | Without Hardware Multiply | 21 | 242 | 15.1 µs | 20.1 µs | 96.8 µs | 242 µs |
| | Hardware Multiply | 28 | 28 | 1.7 µs | 2.3 µs | 11.2 µs | 28 µs |
| 16 x 16 Signed | Without Hardware Multiply | 52 | 254 | 15.8 µs | 21.2 µs | 101.6 µs | 254 µs |
| | Hardware Multiply | 35 | 40 | 2.5 µs | 3.3 µs | 16.0 µs | 40 µs |


© 2009-2018 Microchip Technology Inc.

Example 10-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 10-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES3:RES0).

**EQUATION 10-1: 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 10-3: 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 10-4 shows the sequence to do a 16 x 16 signed multiply. Equation 10-2 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 10-2: 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 10-4: 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
   :
```

## 11.0 INTERRUPTS

Members of the PIC18F87K22 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.

The registers for controlling interrupt operation 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** – Indicating that an interrupt event occurred
- **Enable bit** – Enabling program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** – Specifying 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 that 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>) and GIEH bit (INTCON<7>) 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 PIC® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit that enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit that 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.

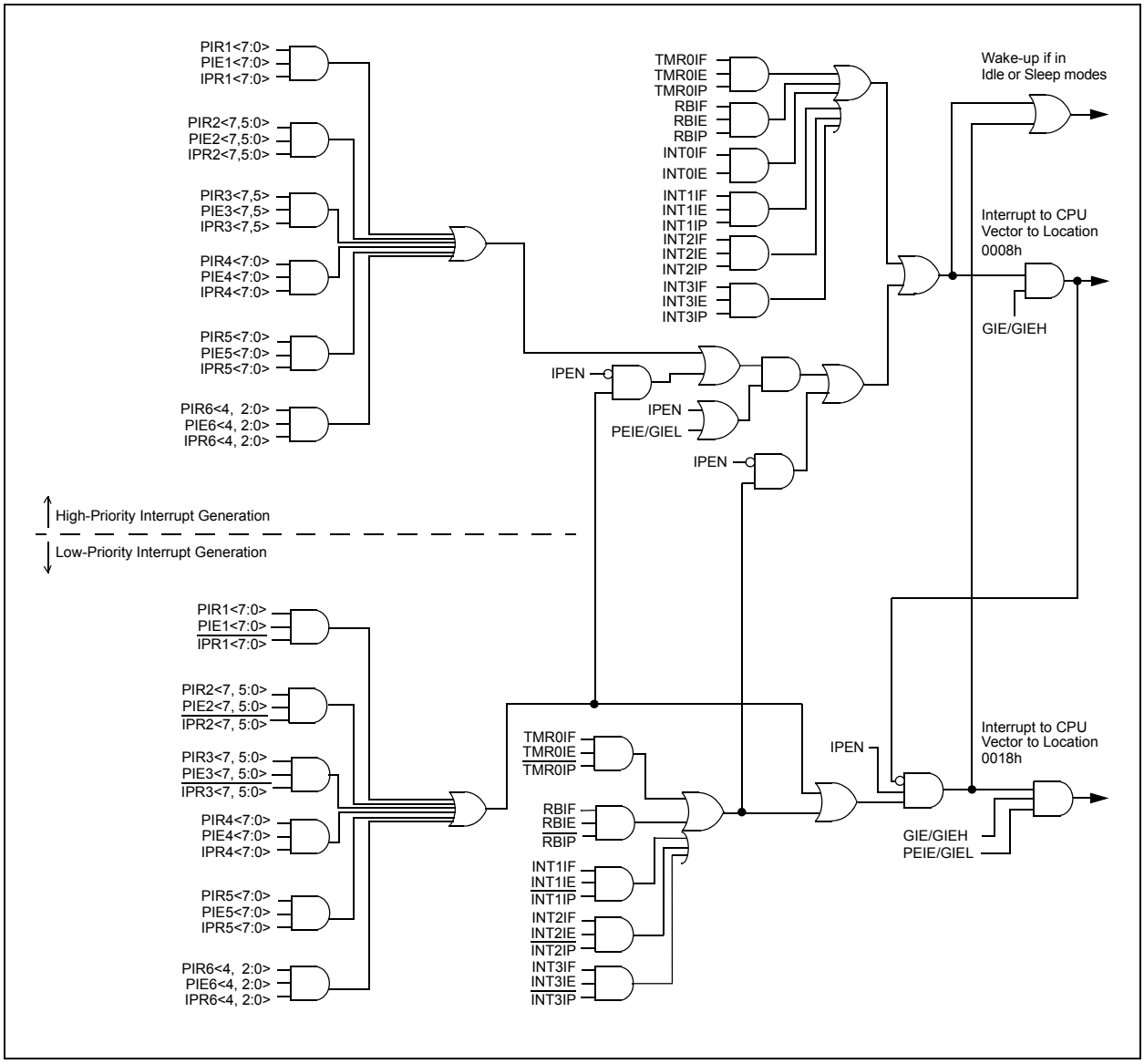
The “return from interrupt” instruction, `RETFIE`, exits the interrupt routine and sets the GIE bit (GIEH or GIEL if priority levels are used) that re-enables interrupts.

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. |
|---|---|


© 2009-2018 Microchip Technology Inc.

**FIGURE 11-1: PIC18F87K22 FAMILY INTERRUPT LOGIC**



## 11.1 INTCON Registers

The INTCON registers are readable and writable registers that 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 11-1: 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
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.


© 2009-2018 Microchip Technology Inc.

**REGISTER 11-2: 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 TRIS register 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 11-3: 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.


© 2009-2018 Microchip Technology Inc.

## 11.2 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are six Peripheral Interrupt Request (Flag) registers (PIR1 through PIR6).

**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 11-4: 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 |
|---|---|---|---|---|---|---|---|
| PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | 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 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or write operation has taken place (must be cleared in software)
0 = No read or write operation 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:** EUSART Receive Interrupt Flag bit
1 = The EUSART receive buffer, RCREG1, is full (cleared when RCREG1 is read)
0 = The EUSART receive buffer is empty

bit 4 **TX1IF:** EUSART Transmit Interrupt Flag bit
1 = The EUSART transmit buffer, TXREG1, is empty (cleared when TXREG1 is written)
0 = The EUSART transmit buffer is full

bit 3 **SSP1IF:** Master Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

bit 2 **TMR1GIF:** Timer1 Gate Interrupt Flag bit
1 = Timer gate interrupt occurred (must be cleared in software)
0 = No timer gate interrupt occurred

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 11-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2**

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| OSCFIF | — | SSP2IF | BCL2IF | BCL1IF | HLVDIF | TMR3IF | TMR3GIF |
| 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 is operating

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

bit 5 **SSP2IF:** Master Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception has been completed (must be cleared in software)
0 = Waiting to transmit/receive

bit 4 **BCL2IF:** Bus Collision Interrupt Flag bit
1 = A bus collision occurred (must be cleared in software)
0 = No bus collision occurred

bit 3 **BCL1IF:** Bus Collision Interrupt Flag bit
1 = A bus collision occurred (must be cleared in software)
0 = No bus collision occurred

bit 2 **HLVDIF:** High/Low-Voltage Detect Interrupt Flag bit
1 = A low-voltage condition occurred (must be cleared in software)
0 = The device voltage is above the regulator's low-voltage trip point

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 **TMR3GIF:** TMR3 Gate Interrupt Flag bit
1 = Timer gate interrupt occurred (must be cleared in software)
0 = No timer gate interrupt occurred


© 2009-2018 Microchip Technology Inc.

**REGISTER 11-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3**

| R/W-0 | U-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| 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 **TMR5GIF:** Timer5 Gate Interrupt Flag bit
1 = Timer gate interrupt occurred (must be cleared in software)
0 = No timer gate interrupt occurred

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

bit 5 **RC2IF:** EUSART Receive Interrupt Flag bit
1 = The EUSART receive buffer, RCREG2, is full (cleared when RCREG2 is read)
0 = The EUSART receive buffer is empty

bit 4 **TX2IF:** EUSART Transmit Interrupt Flag bit
1 = The EUSART transmit buffer, TXREG2, is empty (cleared when TXREG2 is written)
0 = The EUSART transmit buffer is full

bit 3 **CTMUIF:** CTMU Interrupt Flag bit
1 = CTMU interrupt occurred (must be cleared in software)
0 = No CTMU interrupt occurred

bit 2 **CCP2IF:** ECCP2 Interrupt Flag bit
Capture mode:
1 = A TMR register capture occurred (must be cleared in software)
0 = No TMR register capture occurred
Compare mode:
1 = A TMR register compare match occurred (must be cleared in software)
0 = No TMR register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **CCP1IF:** ECCP1 Interrupt Flag bit
Capture mode:
1 = A TMR register capture occurred (must be cleared in software)
0 = No TMR register capture occurred
Compare mode:
1 = A TMR register compare match occurred (must be cleared in software)
0 = No TMR register compare match occurred
PWM mode:
Unused in this mode.

bit 0 **RTCCIF:** RTCC Interrupt Flag bit
1 = RTCC interrupt occurred (must be cleared in software)
0 = No RTCC interrupt occurred

**REGISTER 11-7: PIR4: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 4**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CCP10IF(1) | CCP9IF(1) | CCP8IF | CCP7IF | CCP6IF | 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 7-1 **CCP<10:4>IF:** CCP<10:4> Interrupt Flag bits(1)

Capture Mode:
1 = A TMR register capture occurred (must be cleared in software)
0 = No TMR register capture occurred
Compare Mode:
1 = A TMR register compare match occurred (must be cleared in software)
0 = No TMR register compare match occurred
PWM Mode:
Not used in PWM mode.

bit 0 **CCP3IF:** ECCP3 Interrupt Flag bit

Capture Mode:
1 = A TMR register capture occurred (must be cleared in software)
0 = No TMR register capture occurred
Compare Mode:
1 = A TMR register compare match occurred (must be cleared in software)
0 = No TMR register compare match occurred
PWM Mode:
Not used in PWM mode.

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).


© 2009-2018 Microchip Technology Inc.

**REGISTER 11-8: PIR5: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 5**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| TMR7GIF(1) | TMR12IF(1) | TMR10IF(1) | TMR8IF | TMR7IF(1) | TMR6IF | TMR5IF | TMR4IF |
| 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 **TMR7GIF:** TMR7 Gate Interrupt Flag bits(1)
1 = TMR gate interrupt occurred (bit must be cleared in software)
0 = No TMR gate interrupt occurred

bit 6 **TMR12IF:** TMR12 to PR12 Match Interrupt Flag bit(1)
1 = TMR12 to PR12 match occurred (must be cleared in software)
0 = No TMR12 to PR12 match occurred

bit 5 **TMR10IF:** TMR10 to PR10 Match Interrupt Flag bit(1)
1 = TMR10 to PR10 match occurred (must be cleared in software)
0 = No TMR10 to PR10 match occurred

bit 4 **TMR8IF:** TMR8 to PR8 Match Interrupt Flag bit
1 = TMR8 to PR8 match occurred (must be cleared in software)
0 = No TMR8 to PR8 match occurred

bit 3 **TMR7IF:** TMR7 Overflow Interrupt Flag bit(1)
1 = TMR7 register overflowed (must be cleared in software)
0 = TMR7 register did not overflow

bit 2 **TMR6IF:** TMR6 to PR6 Match Interrupt Flag bit
1 = TMR6 to PR6 match occurred (must be cleared in software)
0 = No TMR6 to PR6 match occurred

bit 1 **TMR5IF:** TMR5 Overflow Interrupt Flag bit
1 = TMR5 register overflowed (must be cleared in software)
0 = TMR5 register did not overflow

bit 0 **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

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

**REGISTER 11-9: PIR6: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 6**

| U-0 | U-0 | U-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | EEIF | — | CMP3IF | CMP2IF | CMP1IF |
| 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 **EEIF:** Data EEDATA/Flash Write Operation Interrupt Flag bit
1 = The write operation is complete (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **Unimplemented:** Read as '0'

bit 2 **CMP3IF:** CMP3 Interrupt Flag bit
1 = CMP3 interrupt occurred (must be cleared in software)
0 = No CMP3 interrupt occurred

bit 1 **CMP2IF:** CMP2 Interrupt Flag bit
1 = CMP2 interrupt occurred (must be cleared in software)
0 = No CMP2 interrupt occurred

bit 0 **CMP1IF:** CM1 Interrupt Flag bit
1 = CMP1 interrupt occurred (must be cleared in software)
0 = No CMP1 interrupt occurred


© 2009-2018 Microchip Technology Inc.

## 11.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are six Peripheral Interrupt Enable registers (PIE1 through PIE6). When IPEN (RCON<7>) = 0, the PEIE bit must be set to enable any of these peripheral interrupts.

**REGISTER 11-10: 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 |
|---|---|---|---|---|---|---|---|
| PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | 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 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP 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:** EUSART Receive Interrupt Enable bit
1 = Enables the EUSART receive interrupt
0 = Disables the EUSART receive interrupt

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

bit 3 **SSP1IE:** Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **TMR1GIE:** TMR1 Gate Interrupt Enable bit
1 = Enables the gate
0 = Disabled the gate

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 11-11: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2**

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| OSCFIE | — | SSP2IE | BCL2IE | BCL1IE | HLVDIE | TMR3IE | TMR3GIE |
| 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 **Unimplemented:** Read as '0'

bit 5 **SSP2IE:** Master Synchronous Serial Port 2 Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 4 **BCL2IE:** Bus Collision Interrupt Enable bit
1 = Enables the bus collision interrupt
0 = Disables the bus collision interrupt

bit 3 **BCL1IE:** Bus Collision Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **HLVDIE:** High/Low-Voltage Detect Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR3IE:** TMR3 Overflow Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR3GIE:** Timer3 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled


© 2009-2018 Microchip Technology Inc.

**REGISTER 11-12: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3**

| R/W-0 | U-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| 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 **TMR5GIE:** Timer5 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

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

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

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

bit 3 **CTMUIE:** CTMU Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **CCP2IE:** ECCP2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **CCP1IE:** ECCP1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **RTCCIE:** RTCC Interrupt Enable bit
1 = Enabled
0 = Disabled

**REGISTER 11-13: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CCP10IE(1) | CCP9IE(1) | CCP8IE | CCP7IE | CCP6IE | 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 7-0 **CCP<10:3>IE:** CCP<10:3> Interrupt Enable bits(1)
1 = Enabled
0 = Disabled

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

**REGISTER 11-14: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| TMR7GIE(1) | TMR12IE(1) | TMR10IE(1) | TMR8IE | TMR7IE(1) | TMR6IE | TMR5IE | TMR4IE |
| 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 **TMR7GIE:** TMR7 Gate Interrupt Enable bit(1)
1 = Enabled
0 = Disabled

bit 6 **TMR12IE:** TMR12 to PR12 Match Interrupt Enable bit(1)
1 = Enables the TMR12 to PR12 match interrupt
0 = Disables the TMR12 to PR12 match interrupt

bit 5 **TMR10IE:** TMR10 to PR10 Match Interrupt Enable bit(1)
1 = Enables the TMR10 to PR10 match interrupt
0 = Disables the TMR10 to PR10 match interrupt

bit 4 **TMR8IE:** TMR8 to PR8 Match Interrupt Enable bit
1 = Enables the TMR8 to PR8 match interrupt
0 = Disables the TMR8 to PR8 match interrupt

bit 3 **TMR7IE:** TMR7 Overflow Interrupt Enable bit(1)
1 = Enables the TMR7 overflow interrupt
0 = Disables the TMR7 overflow interrupt

bit 2 **TMR6IE:** TMR6 to PR6 Match Interrupt Enable bit
1 = Enables the TMR6 to PR6 match interrupt
0 = Disables the TMR6 to PR6 match interrupt

bit 1 **TMR5IE:** TMR5 Overflow Interrupt Enable bit
1 = Enables the TMR5 overflow interrupt
0 = Disables the TMR5 overflow interrupt

bit 0 **TMR4IE:** TMR4 to PR4 Match Interrupt Enable bit
1 = Enables the TMR4 to PR4 match interrupt
0 = Disables the TMR4 to PR4 match interrupt

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).


© 2009-2018 Microchip Technology Inc.

**REGISTER 11-15: PIE6: PERIPHERAL INTERRUPT ENABLE REGISTER 6**

| U-0 | U-0 | U-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | EEIE | — | CMP3IE | CMP2IE | CMP1IE |
| 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 **EEIE:** Data EEDATA/Flash Write Operation Enable bit
1 = Interrupt is enabled
0 = interrupt is disabled

bit 3 **Unimplemented:** Read as '0'

bit 2 **CMP3IE:** CMP3 Enable bit
1 = Interrupt is enabled
0 = interrupt is disabled

bit 1 **CMP2E:** CMP2 Enable bit
1 = Interrupt is enabled
0 = interrupt is disabled

bit 0 **CMP1IE:** CMP1 Enable bit
1 = Interrupt is enabled
0 = interrupt is disabled

## 11.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are six Peripheral Interrupt Priority registers (IPR1 through IPR6). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit (RCON<7>) be set.

**REGISTER 11-16: 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 |
|---|---|---|---|---|---|---|---|
| PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | 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 **PSPIP:** Parallel Slave 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:** EUSART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TX1IP:** EUSART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSP1IP:** Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **TMR1GIP:** Timer1 Gate 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


© 2009-2018 Microchip Technology Inc.

**REGISTER 11-17: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2**

| R/W-1 | U-0 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| OSCFIP | — | SSP2IP | BCL2IP | BCL1IP | HLVDIP | TMR3IP | TMR3GIP |
| 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 **Unimplemented:** Read as '0'

bit 5 **SSP2IP:** Master Synchronous Serial Port 2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **BCL2IP:** Bus Collision Interrupt priority bit (MSSP)
1 = High priority
0 = Low priority

bit 3 **BCL1IP:** Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **HLVDIP:** High/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 **TMR3GIP:** TMR3 Gate Interrupt Priority bit
1 = High priority
0 = Low priority

**REGISTER 11-18: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3**

| R/W-1 | U-0 | R-1 | R-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| 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 **TMR5GIP:** Timer5 Gate interrupt Priority bit
1 = High priority
0 = Low priority

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

bit 5 **RC2IP:** EUSART Receive Priority Flag bit
1 = High priority
0 = Low priority

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

bit 3 **CTMUIP:** CTMU Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP2IP:** ECCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **CCP1IP:** ECCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **RTCCIP:** RTCC Interrupt Priority bit
1 = High priority
0 = Low priority

**REGISTER 11-19: IPR4: PERIPHERAL INTERRUPT PRIORITY REGISTER 4**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| CCP10IP(1) | CCP9IP(1) | CCP8IP | CCP7IP | CCP6IP | 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 7-0 **CCP<10:3>IP:** CCP<10:3> Interrupt Priority bits(1)
1 = High priority
0 = Low priority

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).


© 2009-2018 Microchip Technology Inc.

**REGISTER 11-20: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| TMR7GIP(1) | TMR12IP(1) | TMR10IP(1) | TMR8IP | TMR7IP(1) | TMR6IP | TMR5IP | TMR4IP |
| 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 **TMR7GIP:** TMR7 Gate Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 6 **TMR12IP:** TMR12 to PR12 Match Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 5 **TMR10IP:** TMR10 to PR10 Match Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 4 **TMR8IP:** TMR8 to PR8 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **TMR7IP:** TMR7 Overflow Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 2 **TMR6IP:** TMR6 to PR6 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR5IP:** TMR5 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR4IP:** TMR4 to PR4 Match Interrupt Priority bit
1 = High priority
0 = Low priority

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

**REGISTER 11-21: IPR6: PERIPHERAL INTERRUPT PRIORITY REGISTER 6**

| U-0 | U-0 | U-0 | R/W-1 | U-0 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | EEIP | — | CMP3IP | CMP2IP | CMP1IP |
| 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 **EEIP:** EE Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **Unimplemented:** Read as '0'

bit 2 **CMP3IP:** CMP3 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **CMP2IP:** CMP2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CMP1IP:** CMP1 Interrupt Priority bit
1 = High priority
0 = Low priority


© 2009-2018 Microchip Technology Inc.

## 11.5 RCON Register

The RCON register contains the 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 11-22: RCON: RESET CONTROL REGISTER**

| R/W-0 | R/W-1 | R/W-1 | R/W-1 | R-1 | R-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| IPEN | SBOREN | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| 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 **SBOREN:** Software BOR Enable bit
For details of bit operation, see Register 5-1.

bit 5 **$\overline{CM}$:** Configuration Mismatch Flag bit
1 = A Configuration Mismatch Reset has not occurred
0 = A Configuration Mismatch Reset has occurred (must be subsequently set in software)

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

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

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

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

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

## 11.6 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 that 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. Before re-enabling the interrupt, the flag bit (INTxIF) must be cleared in software in the Interrupt Service Routine.

All external interrupts (INT0, INT1, INT2 and INT3) can wake up the processor from the power-managed modes if bit, INTxIE, was set prior to going into the power-managed modes. If the Global Interrupt Enable bit (GIE) is set, the processor will branch to the interrupt vector following wake-up.

The 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.

## 11.7 TMR0 Interrupt

In 8-bit mode (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>). For further details on the Timer0 module, see **Section 13.0 "Timer0 Module"**.

## 11.8 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>).

## 11.9 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 (ISR). Depending on the user's application, other registers may also need to be saved.

Example 11-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

**EXAMPLE 11-1: 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
```


© 2009-2018 Microchip Technology Inc.

**TABLE 11-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | INTEDG3 | TMR0IP | INT3IP | RBIP |
| INTCON3 | INT2IP | INT1IP | INT3IE | INT2IE | INT1IE | INT3IF | INT2IF | INT1IF |
| PIR1 | PSPIP | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIR2 | OSCFIF | — | SSP2IF | BCL2IF | BCL1IF | HLVDIF | TMR3IF | TMR3GIF |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIR4 | CCP10IF(1) | CCP9IF(1) | CCP8IF | CCP7IF | CCP6IF | CCP5IF | CCP4IF | CCP3IF |
| PIR5 | TMR7GIF(1) | TMR12IF(1) | TMR10IF(1) | TMR8IF | TMR7IF(1) | TMR6IF | TMR5IF | TMR4IF |
| PIR6 | — | — | — | EEIF | — | CMP3IF | CMP2IF | CMP1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| PIE2 | OSCFIE | — | SSP2IE | BCL2IE | BCL1IE | HLVDIE | TMR3IE | TMR3GIE |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| PIE4 | CCP10IE(1) | CCP9IE(1) | CCP8IE | CCP7IE | CCP6IE | CCP5IE | CCP4IE | CCP3IE |
| PIE5 | TMR7GIE(1) | TMR12IE(1) | TMR10IE(1) | TMR8IE | TMR7IE(1) | TMR6IE | TMR5IE | TMR4IE |
| PIE6 | — | — | — | EEIE | — | CMP3IE | CMP2IE | CMP1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| IPR2 | OSCFIP | — | SSP2IP | BCL2IP | BCL1IP | HLVDIP | TMR3IP | TMR3GIP |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| IPR4 | CCP10IP(1) | CCP9IP(1) | CCP8IP | CCP7IP | CCP6IP | CCP5IP | CCP4IP | CCP3IP |
| IPR5 | TMR7GIP(1) | TMR12IP(1) | TMR10IP(1) | TMR8IP | TMR7IP(1) | TMR6IP | TMR5IP | TMR4IP |
| IPR6 | — | — | — | EEIP | — | CMP3IP | CMP2IP | CMP1IP |
| RCON | IPEN | SBOREN | $\overline{\text{CM}}$ | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |

**Legend:** Shaded cells are not used by the interrupts.

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

## 12.0 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)

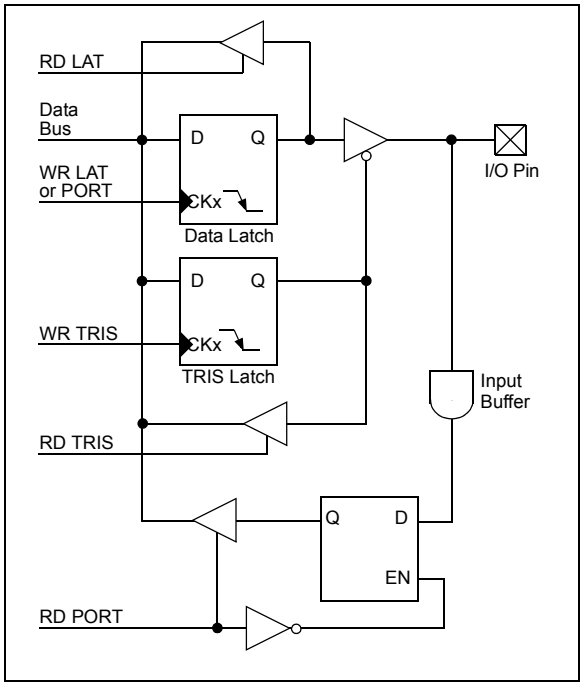
Reading the PORT register reads the current status of the pins, whereas writing to the PORT register writes to the Output Latch (LAT) register.

Setting a TRIS bit (= 1) makes the corresponding port pin an input (putting 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 12-1.

**FIGURE 12-1: GENERIC I/O PORT OPERATION**




## 12.1 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.

All of the digital ports are 5.5V input tolerant. The analog ports have the same tolerance – having clamping diodes implemented internally.

### 12.1.1 PIN OUTPUT DRIVE

When used as digital I/O, the output pin drive strengths vary, according to the pins' grouping, to meet the needs for a variety of applications. In general, there are two classes of output pins, in terms of drive capability:

- Outputs designed to drive higher current loads, such as LEDs:
  - PORTA
  - PORTB
  - PORTC
- Outputs with lower drive levels, but capable of driving normal digital circuit loads with a high input impedance. Able to drive LEDs, but only those with smaller current requirements:
  - PORTD
  - PORTE
  - PORTF
  - PORTG
  - PORTH(†)
  - PORTJ(†)

  † These ports are not available on 64-pin devices.

### 12.1.2 PULL-UP CONFIGURATION

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

The pull-ups are enabled with a single bit for each of the ports: RBPU (INTCON2<7>) for PORTB, and RDPU, REPU and RJPU (PADCFG1<7:5>) for the other ports.


© 2009-2018 Microchip Technology Inc.

**REGISTER 12-1: PADCFG1: PAD CONFIGURATION REGISTER**

| R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|
| RDPU | REPU | RJPU(2) | — | — | RTSECSEL1(1) | RTSECSEL0(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 **RDPU:** PORTD Pull-up Enable bit
1 = PORTD pull-up resistors are enabled by individual port latch values
0 = All PORTD pull-up resistors are disabled

bit 6 **REPU:** PORTE Pull-up Enable bit
1 = PORTE pull-up resistors are enabled by individual port latch values
0 = All PORTE pull-up resistors are disabled

bit 5 **RJPU:** PORTJ Pull-up Enable bit(2)
1 = PORTJ pull-up resistors are enabled by individual port latch values
0 = All PORTJ pull-up resistors are disabled

bit 4-3 **Unimplemented:** Read as '0'

bit 2-1 **RTSECSEL<1:0>:** RTCC Seconds Clock Output Select bits(1)
11 = Reserved; do not use
10 = RTCC source clock is selected for the RTCC pin (the pin can be LF-INTOSC or SOSC, depending on the RTCOSC (CONFIG3L<1>) bit setting)
01 = RTCC seconds clock is selected for the RTCC pin
00 = RTCC alarm pulse is selected for the RTCC pin

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

**Note 1:** To enable the actual RTCC output, the RTCOE (RTCCFG<2>) bit must be set.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.

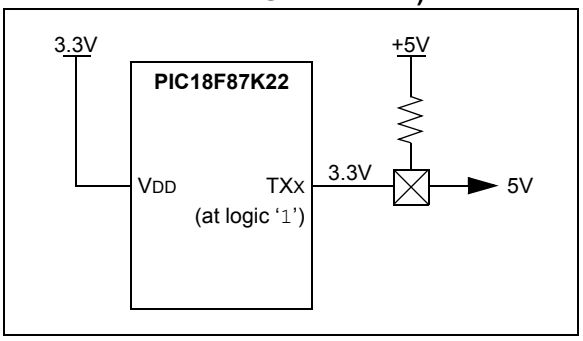
### 12.1.3 OPEN-DRAIN OUTPUTS

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 USARTs, the MSSP module (in SPI mode) and the CCP modules. This option is selectively enabled by setting the open-drain control bits in the registers, ODCON1, ODCON2 and ODCON3.

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 (Figure 12-2). When a digital logic high signal is output, it is pulled up to the higher voltage level.

**FIGURE 12-2: USING THE OPEN-DRAIN OUTPUT (USART SHOWN AS EXAMPLE)**




**REGISTER 12-2: ODCON1: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 1**

| R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| SSP1OD | CCP2OD | CCP1OD | — | — | — | — | SSP2OD |
| 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 **SSP1OD:** MSSP1 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 6 **CCP2OD:** ECCP2 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 5 **CCP1OD:** ECCP1 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 4-1 **Unimplemented:** Read as '0'

bit 0 **SSP2OD:** MSSP2 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled


© 2009-2018 Microchip Technology Inc.

**REGISTER 12-3: ODCON2: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 2**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CCP10OD(1) | CCP9OD(1) | CCP8OD | CCP7OD | CCP6OD | CCP5OD | CCP4OD | CCP3OD |
| 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 **CCP10OD:** CCP10 Open-Drain Output Enable bit(1)
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 6 **CCP9OD:** CCP9 Open-Drain Output Enable bit(1)
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 5 **CCP8OD:** CCP8 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 4 **CCP7OD:** CCP7 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 3 **CCP6OD:** CCP6 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 2 **CCP5OD:** CCP5 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 1 **CCP4OD:** CCP4 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 0 **CCP3OD:** ECCP3 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

**Note 1:** Not implemented on devices with 32-byte program memory (PIC18FX5K22).

**REGISTER 12-4: ODCON3: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 3**

| R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| U2OD | U1OD | — | — | — | — | — | CTMUDS |
| 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 **U2OD:** EUSART2 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 6 **U1OD:** EUSART1 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 5-1 **Unimplemented:** Read as '0'

bit 0 **CTMUDS:** CTMU Pulse Delay Enable bit
1 = Pulse delay input for CTMU is enabled on pin, RF1
0 = Pulse delay input for CTMU is disabled on pin, RF1

### 12.1.4 ANALOG AND DIGITAL PORTS

Many of the ports multiplex analog and digital functionality, providing a lot of flexibility for hardware designers. PIC18F87K22 family devices can make any analog pin analog or digital, depending on an application's needs. The ports' analog/digital functionality is controlled by registers: ANCON0, ANCON1 and ANCON2.

Setting these registers makes the corresponding pins analog and clearing the registers makes the ports digital. For details on these registers, see **Section 23.0 "12-Bit Analog-to-Digital Converter (A/D) Module"**.


© 2009-2018 Microchip Technology Inc.

## 12.2 PORTA, TRISA and LATA Registers

PORTA is an 8-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISA and LATA.

RA4/T0CKI is a Schmitt Trigger input. All other PORTA pins have TTL input levels and full CMOS output drivers.

RA5 and RA<3:0> are multiplexed with analog inputs for the A/D Converter.

The operation of the analog inputs as A/D Converter inputs is selected by clearing or setting the ANSEL control bits in the ANCON1 register. The corresponding TRISA bits control the direction of these 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.

| **Note:** | 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. |
|---|---|

OSC2/CLKO/RA6 and OSC1/CLKI/RA7 normally serve as the external circuit connections for the external (primary) oscillator circuit (HS Oscillator modes), or the external clock input and output (EC 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'. When the device is configured to use HF-INTOSC, MF-INTOSC or LF-INTOSC as the default oscillator mode, RA6 and RA7 are automatically configured as digital I/O; the oscillator and clock in/clock out functions are disabled.

RA5 has additional functionality for Timer1 and Timer3. It can be configured as the Timer1 clock input or the Timer3 external clock gate input.

**EXAMPLE 12-1: INITIALIZING PORTA**

```
CLRF     PORTA    ;  Initialize PORTA by
                  ; clearing output latches
CLRF     LATA     ; Alternate method to
                  ; clear output data latches
BANKSEL  ANCON1   ; Select bank with ANCON1 register
MOVLW    00h      ; Configure A/D
MOVWF    ANCON1   ; for digital inputs
BANKSEL  TRISA    ; Select bank with TRISA register
MOVLW    0BFh     ; Value used to initialize
                  ; data direction
MOVWF    TRISA    ; Set RA<7, 5:0> as inputs,
                  ; RA<6> as output
```

**TABLE 12-1: PORTA FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RA0/AN0/ULPWU | 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. |
| | ULPWU | 1 | I | ANA | Ultra Low-Power Wake-up input. |
| 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. |
| | | 1 | I | TTL | PORTA<2> data input; disabled when analog functions are enabled. |
| | AN2 | 1 | I | ANA | A/D Input Channel 2. Default input configuration on POR. |
| | VREF- | 1 | I | ANA | A/D and comparator 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 and comparator high reference voltage input. |
| RA4/T0CKI | RA4 | 0 | O | DIG | LATA<4> data output. |
| | | 1 | I | ST | PORTA<4> data input. Default configuration on POR. |
| | T0CKI | x | I | ST | Timer0 clock input. |
| RA5/AN4/T1CKI/ T3G/HLVDIN | 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. |
| | AN4 | 1 | I | ANA | A/D Input Channel 4. Default configuration on POR. |
| | T1CKI | x | I | ST | Timer1 clock input. |
| | T3G | x | I | ST | Timer3 external clock gate input. |
| | HLVDIN | 1 | I | ANA | High/Low-Voltage Detect (HLVD) external trip point input. |
| OSC2/CLKO/RA6 | OSC2 | x | O | ANA | Main oscillator feedback output connection (HS, XT and LP modes). |
| | CLKO | x | O | DIG | System cycle clock output (FOSC/4, EC and INTOSC 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, XT and LP modes). |
| | CLKI | x | I | ANA | Main external clock source input (EC 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 Trigger Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**TABLE 12-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTA | RA7(1) | RA6(1) | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 |
| LATA | LATA7(1) | LATA6(1) | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 |
| TRISA | TRISA7(1) | TRISA6(1) | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| ANCON0 | ANSEL7 | ANSEL6 | ANSEL5 | ANSEL4 | ANSEL3 | ANSEL2 | ANSEL1 | ANSEL0 |

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

**Note 1:** These bits are enabled depending on the oscillator mode selected. When not enabled as PORTA pins, they are disabled and read as 'x'.


© 2009-2018 Microchip Technology Inc.

## 12.3 PORTB, TRISB and LATB Registers

PORTB is an eight-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISB and LATB. All pins on PORTB are digital only.

**EXAMPLE 12-2: INITIALIZING PORTB**

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

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. Any RB<7:4> pin configured as an output will be excluded from the interrupt-on-change comparison.

Comparisons with the input pins (of RB<7:4>) are made 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. To clear the interrupt in the Interrupt Service Routine:

a) Any read or write of PORTB (except with the `MOVFF (ANY), PORTB` instruction). This will end the mismatch condition.
b) Wait one instruction cycle (such as executing a `NOP` instruction).
c) 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 one TCY 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.

The RB<3:2> pins are multiplexed as CTMU edge inputs. RB5 has an additional function for Timer3 and Timer1. It can be configured for Timer3 clock input or Timer1 external clock gate input.

**TABLE 12-3: 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 | x | I | ST | Enhanced PWM Fault input for ECCPx. |
| RB1/INT1 | 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. |
| RB2/INT2/CTED1 | 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. |
| | CTED1 | x | I | ST | CTMU Edge 1 input. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger 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 when the CCP2MX Configuration bit is cleared and in Extended Microcontroller mode.

© 2009-2018 Microchip Technology Inc.

**TABLE 12-3: PORTB FUNCTIONS (CONTINUED)**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RB3/INT3/CTED2/ 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. |
| | CTED2 | x | I | ST | CTMU Edge 2 input. |
| | ECCP2(1) | 0 | O | DIG | ECCP2 compare output and ECCP2 PWM output. Takes priority over port data. |
| | | 1 | I | ST | ECCP2 capture input. |
| | P2A | 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 | 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 | 1 | I | TTL | Interrupt-on-pin change. |
| RB5/KBI1/T3CKI/ T1G | 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 | 1 | I | TTL | Interrupt-on-pin change. |
| | T3CKI | x | I | ST | Timer3 clock input. |
| | T1G | x | I | ST | Timer1 external clock gate input. |
| 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 operations. |
| 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 operations. |
| | | x | I | ST | Serial execution data input for ICSP and ICD operations. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger 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 when the CCP2MX Configuration bit is cleared and in Extended Microcontroller mode.

**TABLE 12-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| INTCON2 | $\overline{\text{RBPU}}$ | INTEDG0 | INTEDG1 | INTEDG2 | INTEDG3 | TMR0IP | INT3IP | RBIP |
| INTCON3 | INT2IP | INT1IP | INT3IE | INT2IE | INT1IE | INT3IF | INT2IF | INT1IF |
| ODCON1 | SSP1OD | CCP2OD | CCP1OD | — | — | — | — | SSP2OD |

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


© 2009-2018 Microchip Technology Inc.

## 12.4 PORTC, TRISC and LATC Registers

PORTC is an eight-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISC and LATC. Only PORTC pins, RC2 through RC7, are digital only pins.

PORTC is multiplexed with ECCP, MSSP and EUSART peripheral functions (Table 12-5). The pins have Schmitt Trigger input buffers. The pins for ECCP, SPI and EUSART are also configurable for open-drain output whenever these functions are active. Open-drain configuration is selected by setting the SPIOD, CCPxOD and U1OD control bits in the registers, ODCON1 and ODCON3.

RC1 is normally configured as the default peripheral pin for the ECCP2 module. The assignment of ECCP2 is controlled by Configuration bit, CCP2MX (default state, CCP2MX = 1).

When enabling peripheral functions, use care in defining TRIS bits for each PORTC pin. Some peripherals can override the TRIS bit to make a pin an output or input. Consult 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 12-3: INITIALIZING PORTC**

```
CLRF    PORTC   ; Initialize PORTC by
                ; clearing output
                ; data latches
CLRF    LATC    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISC   ; Set RC<3:0> as inputs
                ; RC<5:4> as outputs
                ; RC<7:6> as inputs
```

**TABLE 12-5: PORTC FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RC0/SOSCO/ SCLKI/ | RC0 | 0 | O | DIG | LATC<0> data output. |
| | | 1 | I | ST | PORTC<0> data input. |
| | SOSCO | 1 | I | ST | SOSC oscillator output. |
| | SCLKI | 1 | I | ST | Digital clock input; enabled when SOSC oscillator is disabled. |
| RC1/SOSCI/ ECCP2/P2A | RC1 | 0 | O | DIG | LATC<1> data output. |
| | | 1 | I | ST | PORTC<1> data input. |
| | SOSCI | x | I | ANA | SOSC oscillator input. |
| | ECCP2(1) | 0 | O | DIG | ECCP2 compare output and ECCP2 PWM output; takes priority over port data. |
| | | 1 | I | ST | ECCP2 capture input. |
| | P2A | 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. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, TTL = TTL Buffer Input, $I^2C$ = $I^2C$/SMBus 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 when the CCP2MX Configuration bit is set.

**TABLE 12-5: PORTC FUNCTIONS (CONTINUED)**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| 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 (MSSP module); takes priority over port data. |
| | | 1 | I | ST | SPI clock input (MSSP module). |
| | SCL1 | 0 | O | DIG | $I^2C$ clock output (MSSP module); takes priority over port data. |
| | | 1 | I | $I^2C$ | $I^2C$ clock input (MSSP 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 | | I | ST | SPI data input (MSSP module). |
| | SDA1 | 1 | O | DIG | $I^2C$ data output (MSSP module); takes priority over port data. |
| | | 1 | I | $I^2C$ | $I^2C$ data input (MSSP 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 (MSSP module). |
| 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 (EUSART module); takes priority over port data. |
| | CK1 | 1 | O | DIG | Synchronous serial data input (EUSART module); user must configure as an input. |
| | | 1 | I | ST | Synchronous serial clock input (EUSART 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 (EUSART module). |
| | DT1 | 1 | O | DIG | Synchronous serial data output (EUSART module); takes priority over port data. |
| | | 1 | I | ST | Synchronous serial data input (EUSART module); user must configure as an input. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, TTL = TTL Buffer Input, $I^2C$ = $I^2C$/SMBus 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 when the CCP2MX Configuration bit is set.

**TABLE 12-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 |
| LATC | LATC7 | LATBC6 | LATC5 | LATCB4 | LATC3 | LATC2 | LATC1 | LATC0 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| ODCON1 | SSP1OD | CCP2OD | CCP1OD | — | — | — | — | SSP2OD |
| ODCON3 | U2OD | U1OD | — | — | — | — | — | CTMUDS |

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


© 2009-2018 Microchip Technology Inc.

## 12.5 PORTD, TRISD and LATD Registers

PORTD is an 8-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISD and LATD.

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.

Each of the PORTD pins has a weak internal pull-up. A single control bit can turn off all the pull-ups. This is performed by setting bit, RDPU (PADCFG1<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.

On 80-pin devices, PORTD is multiplexed with the system bus as part of the external memory interface. The 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 can also be configured as an 8-bit wide microprocessor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL. For additional information, see **Section 12.11 "Parallel Slave Port"**.

The PORTD also has the $I^2C$ and SPI functionality on RD4, RD5 and RD6. The pins for SPI are also configurable for open-drain output. Open-drain configuration is selected by setting bit, SSP2OD (ODCON1<0>).

RD0 has a CTMU functionality. RD1 has the functionality for the Timer5 clock input and Timer7 external clock gate input.

**EXAMPLE 12-4: INITIALIZING PORTD**

```
CLRF    PORTD   ; Initialize PORTD by
                ; clearing output
                ; data latches
CLRF    LATD    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISD   ; Set RD<3:0> as inputs
                ; RD<5:4> as outputs
                ; RD<7:6> as inputs
```

**TABLE 12-7: PORTD FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RD0/PSP0/ AD0/CTPLS | RD0 | 0 | O | DIG | LATD<0> data output. |
| | | 1 | I | ST | PORTD<0> data input. |
| | PSP0(1) | x | I/O | TTL | Parallel Slave Port data. |
| | AD0(2) | x | I/O | TTL | External Memory Address/Data 0. |
| | CTPLS | x | O | DIG | CTMU pulse generator output. |
| RD1/PSP1/ AD1/T5CKI/ T7G | RD1 | 0 | O | DIG | LATD<1> data output. |
| | | 1 | I | ST | PORTD<1> data input. |
| | PSP1(1) | x | I/O | TTL | Parallel Slave Port data. |
| | AD1(2) | x | I/O | TTL | External Memory Address/Data 1. |
| | T5CKI | x | I | ST | Timer5 clock input. |
| | T7G | x | I | ST | Timer7 external clock gate input. |
| RD2/PSP2/AD2 | RD2 | 0 | O | DIG | LATD<2> data output. |
| | | 1 | I | ST | PORTD<2> data input. |
| | PSP2(1) | x | I/O | TTL | Parallel Slave Port data. |
| | AD2(2) | x | I/O | TTL | External Memory Address/Data 2. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, $I^2C$ = $I^2C$/SMBus Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** The Parallel Slave Port (PSP) is available only in Microcontroller mode.

**2:** This feature is available only on PIC18F8XK22 devices.

© 2009-2018 Microchip Technology Inc.

**TABLE 12-7: PORTD FUNCTIONS (CONTINUED)**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RD3/PSP3/AD3 | RD3 | 0 | O | DIG | LATD<3> data output. |
| | | 1 | I | ST | PORTD<3> data input. |
| | PSP3(1) | x | I/O | TTL | Parallel Slave Port data. |
| | AD3(2) | x | I/O | TTL | External Memory Address/Data 3. |
| RD4/PSP4/ AD4/SDO2 | RD4 | 0 | O | DIG | LATD<4> data output. |
| | | 1 | I | ST | PORTD<4> data input. |
| | PSP4(1) | x | I/O | TTL | Parallel Slave Port data. |
| | AD4(2) | x | I/O | TTL | External Memory Address/Data 4. |
| | SDO2 | 0 | P | DOG | SPI data output (MSSP module). |
| RD5/PSP5/ AD5/SDI2/ SDA2 | RD5 | 0 | O | DIG | LATD<5> data output. |
| | | 1 | I | ST | PORTD<5> data input. |
| | PSP5(1) | x | I/O | TTL | Parallel Slave Port data. |
| | AD5(2) | x | I/O | TTL | External Memory Address/Data 5. |
| | SDI2 | 1 | I | ST | SPI data input (MSSP module). |
| | SDA2 | 0 | O | $I^2C$ | $I^2C$ data input (MSSP module). Input type depends on module setting. |
| RD6/PSP6/ AD6/SCK2/ SCL2 | RD6 | 0 | O | DIG | LATD<6> data output. |
| | | 1 | I | ST | PORTD<6> data input. |
| | PSP6(1) | x | I/O | TTL | Parallel Slave Port data. |
| | AD6(2) | x | I/O | TTL | External Memory Address/Data 6. |
| | SCK2 | 0 | O | DIG | SPI clock output (MSSP module); takes priority over port data. |
| | | 1 | I | ST | SPI clock input (MSSP module). |
| | SCL2 | 0 | O | DIG | $I^2C$ clock output (MSSP module); takes priority over port data. |
| | | 1 | I | $I^2C$ | $I^2C$ clock input (MSSP module). Input type depends on module setting. |
| RD7/PSP7/ AD7/$\overline{SS2}$ | RD7 | 0 | O | DIG | LATD<7> data output. |
| | | 1 | I | ST | PORTD<7> data input. |
| | PSP7(1) | x | I/O | TTL | Parallel Slave Port data. |
| | AD7(2) | x | I/O | TTL | External Memory Address/Data 7. |
| | $\overline{SS2}$ | 1 | I | TTL | Slave select input for MSSP module. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, $I^2C$ = $I^2C$/SMBus Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**Note 1:** The Parallel Slave Port (PSP) is available only in Microcontroller mode.

**2:** This feature is available only on PIC18F8XK22 devices.

**TABLE 12-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 |
| LATD | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 |
| PADCFG1 | RDPU | REPU | RJPU(1) | — | — | RTSECSEL1 | RESECSEL0 | — |
| ODCON1 | SSP1OD | CCP2OD | CCP1OD | — | — | — | — | SSP2OD |

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

**Note 1:** Unimplemented on PIC18F6XK22 devices, read as '0'.


© 2009-2018 Microchip Technology Inc.

## 12.6 PORTE, TRISE and LATE Registers

PORTE is an eight-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISE and LATE.

All pins on PORTE are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. The RE7 pin is also configurable for open-drain output when ECCP2 is active on this pin. Open-drain configuration is selected by setting the CCP2OD control bit (ODCON1<6>)

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

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 setting bit, REPU (PADCFG1<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, for 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 re-assigns 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 Slave Port address lines. RE1 and RE0 are multiplexed with the control signals, $\overline{WR}$ and $\overline{RD}$.

RE3 can also be configured as the Reference Clock Output (REFO) from the system clock. For further details, see **Section 3.7 "Reference Clock Output"**.

**EXAMPLE 12-5: 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 12-9: PORTE FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RE0/$\overline{RD}$/P2D AD8 | RE0 | 0 | O | DIG | LATE<0> data output. |
| | | 1 | I | ST | PORTE<0> data input. |
| | $\overline{RD}$ | x | O | DIG | Parallel Slave Port read strobe pin. |
| | | x | I | TTL | Parallel Slave Port read pin. |
| | P2D | 0 | O | — | ECCP2 PWM Output D.<br>May be configured for tri-state during Enhanced PWM shutdown events. |
| | AD8(2) | x | O | DIG | External memory interface, Data Bit 8 output. |
| | | x | I | TTL | External memory interface, Data Bit 8 input. |
| RE1/P2C/$\overline{WR}$/ AD9 | RE1 | 0 | O | DIG | LATE<1> data output. |
| | | 1 | I | ST | PORTE<1> data input. |
| | P2C | 0 | O | — | ECCP2 PWM Output C.<br>May be configured for tri-state during Enhanced PWM shutdown events. |
| | $\overline{WR}$ | x | O | DIG | Parallel Slave Port write strobe pin. |
| | | x | I | TTL | Parallel Slave Port write pin. |
| | AD9(2) | x | O | DIG | External memory interface, Data Bit 9 output. |
| | | x | I | TTL | External memory interface, Data Bit 9 input. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger 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 when the CCP2MX Configuration bit is cleared and in Microcontroller mode.

**2:** This feature is only available on PIC18F8XKXX devices.

**TABLE 12-9: PORTE FUNCTIONS (CONTINUED)**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RE2/$\overline{CS}$/P2B/ CCP10/AD10 | RE2 | 0 | O | DIG | LATE<2> data output. |
| | | 1 | I | ST | PORTE<2> data input. |
| | $\overline{CS}$ | x | I | TTL | Parallel Slave Port chip select. |
| | P2B | 0 | O | — | ECCP2 PWM Output B. May be configured for tri-state during Enhanced PWM shutdown events. |
| | CCP10 | 1 | I/O | ST | Capture 10 input/Compare 10 output/PWM10 output. |
| | AD10(2) | x | O | DIG | External memory interface, Address/Data Bit 10 output. |
| | | x | I | TTL | External memory interface, Data Bit 10 input. |
| RE3/P3C/ CCP9/REFO/ AD11 | RE3 | 0 | O | DIG | LATE<3> data output. |
| | | 1 | I | ST | PORTE<3> data input. |
| | P3C | 0 | O | — | ECCP3 PWM Output C. May be configured for tri-state during Enhanced PWM shutdown events. |
| | CCP9 | 0 | O | DIG | CCP9 Compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP9 capture input. |
| | REFO | x | O | DIG | Reference output clock. |
| | AD11(2) | x | O | DIG | External memory interface, Address/Data Bit 11 output. |
| | | x | I | TTL | External memory interface, Data Bit 11 input. |
| RE4/P3B/ CCP8/AD12 | RE4 | 0 | O | DIG | LATE<4> data output. |
| | | 1 | I | ST | PORTE<4> data input. |
| | P3B | 0 | O | — | ECCP3 PWM Output B. May be configured for tri-state during Enhanced PWM shutdown events. |
| | CCP8 | 0 | O | DIG | CCP8 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP8 capture input. |
| | AD12(2) | x | O | DIG | External memory interface, Address/Data Bit 12 output. |
| | | x | I | TTL | External memory interface, Data Bit 12 input. |
| RE5/P1C/ CCP7/AD13 | RE5 | 0 | O | DIG | LATE<5> data output. |
| | | 1 | I | ST | PORTE<5> data input. |
| | P1C | 0 | O | — | ECCP1 PWM Output C. May be configured for tri-state during Enhanced PWM shutdown events. |
| | CCP7 | 0 | O | DIG | CCP7 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP7 capture input. |
| | AD13(2) | x | O | DIG | External memory interface, Address/Data Bit 13 output. |
| | | x | I | TTL | External memory interface, Data Bit 13 input. |
| RE6/P1B/ CCP6/AD14 | RE6 | 0 | O | DIG | LATE<6> data output. |
| | | 1 | I | ST | PORTE<6> data input. |
| | P1B | 0 | O | — | ECCP1 PWM Output B. May be configured for tri-state during Enhanced PWM shutdown events. |
| | CCP6 | 0 | O | DIG | CCP6 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP9 capture input. |
| | AD14(2) | x | O | DIG | External memory interface, Address/Data Bit 14 output. |
| | | x | I | TTL | External memory interface, Data Bit 14 input. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger 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 when the CCP2MX Configuration bit is cleared and in Microcontroller mode.

**2:** This feature is only available on PIC18F8XKXX devices.


© 2009-2018 Microchip Technology Inc.

**TABLE 12-9: PORTE FUNCTIONS (CONTINUED)**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RE7/ECCP2/ P2A/AD15 | RE7 | 0 | O | DIG | LATE<7> data output. |
| | | 1 | I | ST | PORTE<7> data input. |
| | ECCP2(1) | 0 | O | DIG | ECCP2 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | ECCP2 capture input. |
| | P2A | 0 | O | — | ECCP2 PWM Output A.<br>May be configured for tri-state during Enhanced PWM shutdown event. |
| | AD15(2) | x | O | DIG | External memory interface, Address/Data Bit 15 output. |
| | | x | I | TTL | External memory interface, Data Bit 15 input. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger 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 when the CCP2MX Configuration bit is cleared and in Microcontroller mode.

**2:** This feature is only available on PIC18F8XKXX devices.

**TABLE 12-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTE | RE7 | RE6 | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 |
| LATE | LATE7 | LATE6 | LATE5 | LATE4 | LATE3 | LATE2 | LATE1 | LATE0 |
| TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 |
| PADCFG1 | RDPU | REPU | RJPU(2) | — | — | RTSECSEL1 | RTSECSEL0 | — |
| ODCON1 | SSP1OD | CCP2OD | CCP1OD | — | — | — | — | SSP2OD |
| ODCON2 | CCP10OD(1) | CCP9OD(1) | CCP8OD | CCP7OD | CCP6OD | CCP5OD | CCP4OD | CCP3OD |

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

**Note 1:** Unimplemented on PIC18FX5K22 devices, read as '0'.

**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.

## 12.7 PORTF, LATF and TRISF Registers

PORTF is a 7-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISF and LATF. All pins on PORTF are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Pins, RF1 through RF6, may be used as comparator inputs or outputs by setting the appropriate bits in the CMCON register. To use RF<7:1> as digital inputs, it is also necessary to turn off the comparators.

**Note 1:** On device Resets, pins, RF<7:1>, are configured as analog inputs and are read as '0'.

**2:** To configure PORTF as a digital I/O, turn off the comparators and clear ANCON1 and ANCON2 to digital.

**EXAMPLE 12-6: INITIALIZING PORTF**

```
CLRF     PORTF    ; Initialize PORTF by
                  ; clearing output
                  ; data latches
CLRF     LATF     ; Alternate method
                  ; to clear output
                  ; data latches
BANKSEL  ANCON1   ; Select bank with ANCON1 register
MOVLW    1Fh      ; Make AN6, AN7 and AN5 digital
MOVWF    ANCON1   ;
MOVLW    0Fh      ; Make AN8, AN9, AN10 and AN11
                    digital
MOVWF    ANCON    ; Set PORTF as digital I/O
BANKSEL  TRISF    ; Select bank with TRISF register
MOVLW    0CEh     ; Value used to
                  ; initialize data
                  ; direction
MOVWF    TRISF    ; Set RF3:RF1 as inputs
                  ; RF5:RF4 as outputs
                  ; RF7:RF6 as inputs
```

**TABLE 12-11: PORTF FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RF1/AN6/C2OUT/ CTDIN | 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 | 0 | O | DIG | Comparator 2 output; takes priority over port data. |
| | CTDIN | 1 | I | ST | CTMU pulse delay input. |
| RF2/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. |
| | AN7 | 1 | I | ANA | A/D Input Channel 7. Default configuration on POR. |
| | C1OUT | 0 | O | DIG | Comparator 1 output; takes priority over port data. |
| RF3/AN8/C2INB/ CTMUI | 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 and Comparator C2+ input. Default input configuration on POR; not affected by analog output. |
| | C2INB | 1 | I | ANA | Comparator 2 Input B. |
| | CTMUI | x | O | — | CTMU pulse generator charger for the C2INB comparator input. |
| 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 and Comparator C2- input. Default input configuration on POR; does not affect digital output. |
| | C2INA | 1 | I | ANA | Comparator 2 Input A. |
| RF5/AN10/CVREF/ C1INB | 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. |
| | AN10 | 1 | I | ANA | A/D Input Channel 10 and Comparator C1+ input. Default input configuration on POR. |
| | CVREF | x | O | ANA | Comparator voltage reference output. Enabling this feature disables digital I/O. |
| | C1INB | 1 | I | ANA | Comparator 1 Input B. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).


© 2009-2018 Microchip Technology Inc.

**TABLE 12-11: PORTF FUNCTIONS (CONTINUED)**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RF6/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. |
| | AN11 | 1 | I | ANA | A/D Input Channel 11 and Comparator C1- input. Default input configuration on POR; does not affect digital output. |
| | C1INA | 1 | I | ANA | Comparator 1 Input A. |
| RF7/AN5/$\overline{SS1}$ | RF7 | 0 | O | DIG | LATF<7> data output; not affected by analog input. |
| | | 1 | I | ST | PORTF<7> data input; disabled when analog input is enabled. |
| | AN5 | 1 | I | ANA | A/D Input Channel 5. Default configuration on POR. |
| | $\overline{SS1}$ | 1 | I | TTL | Slave select input for MSSP module. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, TTL = TTL Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**TABLE 12-12: SUMMARY OF REGISTERS ASSOCIATED WITH PORTF**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | — |
| LATF | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | — |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — |
| ANCON0 | ANSEL7 | ANSEL6 | ANSEL5 | ANSEL4 | ANSEL3 | ANSEL2 | ANSEL1 | ANSEL0 |
| ANCON1 | ANSEL15 | ANSEL14 | ANSEL13 | ANSEL12 | ANSEL11 | ANSEL10 | ANSEL9 | ANSEL8 |

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

## 12.8 PORTG, TRISG and LATG Registers

PORTG is a 5-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISG and LATG.

PORTG is multiplexed with the AUSART and CCP, ECCP, Analog, Comparator, RTCC and Timer input functions (Table 12-13). When operating as I/O, all PORTG pins have Schmitt Trigger input buffers. The open-drain functionality for the CCPx and UART can be configured using ODCONx.

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.

**EXAMPLE 12-7: INITIALIZING PORTG**

```
CLRF    PORTG       ;  Initialize PORTG by
                    ; clearing output
                    ; data latches
BCF     CM1CON, CON ; disable
                    ; comparator 1
CLRF    LATG        ; Alternate method
                    ; to clear output
                    ; data latches
BANKSEL ANCON2      ; Select bank with ACON2 register
MOVLW   0F0h        ; make AN16 to AN19
                    ; digital
MOVWF   ANCON2
BANKSEL TRISG       ; Select bank with TRISG register
MOVLW   04h         ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISG       ; Set RG1:RG0 as
                    ; outputs
                    ; RG2 as input
                    ; RG4:RG3 as inputs
```

**TABLE 12-13: PORTG FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RG0/ECCP3/ P3A | RG0 | 0 | O | DIG | LATG<0> data output. |
| | | 1 | I | ST | PORTG<0> data input. |
| | ECCP3 | 0 | O | DIG | ECCP3 compare output and ECCP3 PWM output; takes priority over port data. |
| | | 1 | I | ST | ECCP3 capture input. |
| | P3A | 0 | O | — | ECCP3 PWM Output A. May be configured for tri-state during Enhanced PWM shutdown events. |
| RG1/TX2/CK2/ AN19/C3OUT | RG1 | 0 | O | DIG | LATG<1> data output. |
| | | 1 | I | ST | PORTG<1> data input. |
| | TX2 | 1 | O | DIG | Synchronous serial data output (EUSART module); takes priority over port data. |
| | CK2 | 1 | O | DIG | Synchronous serial data input (EUSART module); user must configure as an input. |
| | | 1 | I | ST | Synchronous serial clock input (EUSART module). |
| | AN19 | 1 | I | ANA | A/D Input Channel 19. Default input configuration on POR. Does not affect digital output. |
| | C3OUT | x | O | DIG | Comparator 3 output. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).


© 2009-2018 Microchip Technology Inc.

**TABLE 12-13: PORTG FUNCTIONS (CONTINUED)**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RG2/RX2/DT2/ AN18/C3INA | RG2 | 0 | O | DIG | LATG<2> data output. |
| | | 1 | I | ST | PORTG<2> data input. |
| | RX2 | 1 | I | ST | Asynchronous serial receive data input (EUSART module). |
| | DT2 | 1 | O | DIG | Synchronous serial data output (EUSART module); takes priority over port data. |
| | | 1 | I | ST | Synchronous serial data input (EUSART module); user must configure as an input. |
| | AN18 | 1 | I | ANA | A/D Input Channel 18. Default input configuration on POR; does not affect digital output. |
| | C3INA | x | I | ANA | Comparator 3 Input A. |
| RG3/CCP4/AN17/ P3D/C3INB | RG3 | 0 | O | DIG | LATG<3> data output. |
| | | 1 | I | ST | PORTG<3> data input. |
| | CCP4 | 0 | O | DIG | CCP4 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP4 capture input. |
| | AN17 | 1 | I | ANA | A/D Input Channel 17. Default input configuration on POR; does not affect digital output. |
| | P3D | 0 | O | — | ECCP3 PWM Output D. May be configured for tri-state during Enhanced PWM. |
| | C3INB | x | I | ANA | Comparator 3 Input B. |
| RG4/RTCC/ T7CKI/T5G/ CCP5/AN16/ P1D/C3INC | RG4 | 0 | O | DIG | LATG<4> data output. |
| | | 1 | I | ST | PORTG<4> data input. |
| | RTCC | x | O | DIG | RTCC output. |
| | T7CKI | x | I | ST | Timer7 clock input. |
| | T5G | x | I | ST | Timer5 external clock gate input. |
| | CCP5 | 0 | O | DIG | CCP5 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP5 capture input. |
| | AN16 | 1 | I | ANA | A/D Input Channel 17. Default input configuration on POR; does not affect digital output. |
| | P1D | 0 | O | — | ECCP1 PWM Output D. May be configured for tri-state during Enhanced PWM. |
| | C3INC | x | I | ANA | Comparator 3 Input C. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**TABLE 12-14: SUMMARY OF REGISTERS ASSOCIATED WITH PORTG**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTG | — | — | RG5(2) | RG4 | RG3 | RG2 | RG1 | RG0 |
| TRISG | — | — | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 |
| ANCON2 | ANSEL23 | ANSEL22 | ANSEL21 | ANSEL20 | ANSEL19 | ANSEL18 | ANSEL17 | ANSEL16 |
| ODCON1 | SSP1OD | CCP2OD | CCP1OD | — | — | — | — | SSP2OD |
| ODCON2 | CCP10OD(1) | CCP9OD(1) | CCP8OD | CCP7OD | CCP6OD | CCP5OD | CCP4OD | CCP3OD |
| ODCON3 | U2OD | U1OD | — | — | — | — | — | CTMUDS |

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

**Note 1:** Unimplemented on PIC18FX5K22 devices, read as '0'.

**2:** This bit is available when Master Clear is disabled (MCLRE = 0). When MCLRE is set, the bit is unimplemented.

## 12.9 PORTH, LATH and TRISH Registers

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

PORTH is an 8-bit wide, bidirectional I/O port. The corresponding Data Direction and Output Latch registers are TRISH and LATH.

All pins on PORTH are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

**EXAMPLE 12-8: INITIALIZING PORTH**

```
CLRF     PORTH    ; Initialize PORTH by
                  ; clearing output
                  ; data latches
CLRF     LATH     ; Alternate method
                  ; to clear output
                  ; data latches
BANKSEL  ANCON2   ; Select bank with ANCON2 register
MOVLW    0Fh      ; Configure PORTH as
MOVWF    ANCON2   ; digital I/O
MOVLW    0Fh      ; Configure PORTH as
MOVWF    ANCON1   ; digital I/O
BANKSEL  TRISH    ; Select bank with TRISH register
MOVLW    0CFh     ; Value used to
                  ; initialize data
                  ; direction
MOVWF    TRISH    ; Set RH3:RH0 as inputs
                  ; RH5:RH4 as outputs
                  ; RH7:RH6 as inputs
```

**TABLE 12-15: PORTH FUNCTIONS**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RH0/AN23/A16 | RH0 | 0 | O | DIG | LATH<0> data output. |
| | | 1 | I | ST | PORTH<0> data input. |
| | AN23 | 1 | I | ANA | A/D Input Channel 23. Default input configuration on POR; does not affect digital input. |
| | A16 | x | O | DIG | External memory interface, Address Line 16; takes priority over port data. |
| RH1/AN22/A17 | RH1 | 0 | O | DIG | LATH<1> data output. |
| | | 1 | I | ST | PORTH<1> data input. |
| | AN22 | 1 | I | ANA | A/D Input Channel 22. Default input configuration on POR; does not affect digital input. |
| | A17 | x | O | DIG | External memory interface, Address Line 17; takes priority over port data. |
| RH2/AN21/A18 | RH2 | 0 | O | DIG | LATH<2> data output. |
| | | 1 | I | ST | PORTH<2> data input. |
| | AN21 | 1 | I | ANA | A/D Input Channel 21. Default input configuration on POR; does not affect digital input. |
| | A18 | x | O | DIG | External memory interface, Address Line 18; takes priority over port data. |
| RH3/AN20/A19 | RH3 | 0 | O | DIG | LATH<3> data output. |
| | | 1 | I | ST | PORTH<3> data input. |
| | AN20 | 1 | I | ANA | A/D Input Channel 20. Default input configuration on POR; does not affect digital input. |
| | A19 | x | O | DIG | External memory interface, Address Line 19; takes priority over port data. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).


© 2009-2018 Microchip Technology Inc.

**TABLE 12-15: PORTH FUNCTIONS (CONTINUED)**

| Pin Name | Function | TRIS Setting | I/O | I/O Type | Description |
|---|---|---|---|---|---|
| RH4/CCP9/ P3C/AN12/ C2INC | RH4 | 0 | O | DIG | LATH<4> data output. |
| | | 1 | I | ST | PORTH<4> data input. |
| | CCP9 | 0 | O | DIG | CCP9 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP9 capture input. |
| | P3C | 0 | O | — | ECCP3 PWM Output C.<br>May be configured for tri-state during Enhanced PWM. |
| | AN12 | 1 | I | ANA | A/D Input Channel 12.<br>Default input configuration on POR; does not affect digital input. |
| | C2INC | x | I | ANA | Comparator 2 Input C. |
| RH5/CCP8/ P3B/AN13/ C2IND | RH5 | 0 | O | DIG | LATH<5> data output. |
| | | 1 | I | ST | PORTH<5> data input. |
| | CCP8 | 0 | O | DIG | CCP8 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP8 capture input. |
| | P3B | 0 | O | — | ECCP3 PWM Output B.<br>May be configured for tri-state during Enhanced PWM. |
| | AN13 | 1 | I | ANA | A/D Input Channel 13.<br>Default input configuration on POR; does not affect digital input. |
| | C2IND | x | I | ANA | Comparator 2 Input D. |
| RH6/CCP7/ P1C/AN14/ C1INC | RH6 | 0 | O | DIG | LATH<6> data output. |
| | | 1 | I | ST | PORTH<6> data input. |
| | CCP7 | 0 | O | DIG | CCP7 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP7 capture input. |
| | P1C | 0 | O | — | ECCP1 PWM Output C.<br>May be configured for tri-state during Enhanced PWM. |
| | AN14 | 1 | I | ANA | A/D Input Channel 14.<br>Default input configuration on POR; does not affect digital input. |
| | C1INC | x | I | ANA | Comparator 1 Input C. |
| RH7/CCP6/ P1B/AN15 | RH7 | 0 | O | DIG | LATH<7> data output. |
| | | 1 | I | ST | PORTH<7> data input. |
| | CCP6 | 0 | O | DIG | CCP6 compare/PWM output; takes priority over port data. |
| | | 1 | I | ST | CCP6 capture input. |
| | P1B | 0 | O | — | ECCP1 PWM Output B.<br>May be configured for tri-state during Enhanced PWM. |
| | AN15 | 1 | I | ANA | A/D Input Channel 15.<br>Default input configuration on POR; does not affect digital input. |

**Legend:** O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input,
x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**TABLE 12-16: SUMMARY OF REGISTERS ASSOCIATED WITH PORTH**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTH(1) | RH7 | RH6 | RH5 | RH4 | RH3 | RH2 | RH1 | RH0 |
| LATH(1) | LATH7 | LATH6 | LATH5 | LATH4 | LATH3 | LATH2 | LATH1 | LATH0 |
| TRISH(1) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 |
| ANCON1 | ANSEL15 | ANSEL14 | ANSEL13 | ANSEL12 | ANSEL11 | ANSEL10 | ANSEL9 | ANSEL8 |
| ANCON2 | ANSEL23 | ANSEL22 | ANSEL21 | ANSEL20 | ANSEL19 | ANSEL18 | ANSEL17 | ANSEL16 |
| ODCON2 | CCP10OD(2) | CCP9OD(2) | CCP8OD | CCP7OD | CCP6OD | CCP5OD | CCP4OD | CCP3OD |

**Note 1:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.
**2:** Unimplemented on PIC18FX5K22 devices, read as '0'.

## 12.10 PORTJ, TRISJ and LATJ Registers

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

PORTJ is an 8-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISJ and LATJ.

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. The pull-ups are provided to keep the inputs at a known state for the external memory interface while powering up. A single control bit can turn off all the pull-ups. This is performed by clearing bit, RJPU (PADCFG1<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 12-9: INITIALIZING PORTJ**

```
CLRF    PORTJ   ; Initialize PORTJ by
                ; clearing output latches
CLRF    LATJ    ; Alternate method
                ; to clear output latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISJ   ; Set RJ3:RJ0 as inputs
                ; RJ5:RJ4 as output
                ; RJ7:RJ6 as inputs
```


© 2009-2018 Microchip Technology Inc.

**TABLE 12-17: 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; 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, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input,
x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

**TABLE 12-18: SUMMARY OF REGISTERS ASSOCIATED WITH PORTJ**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTJ(1) | RJ7 | RJ6 | RJ5 | RJ4 | RJ3 | RJ2 | RJ1 | RJ0 |
| LATJ(1) | LATJ7 | LATJ6 | LATJ5 | LATJ4 | LATJ3 | LATJ2 | LATJ1 | LATJ0 |
| TRISJ(1) | TRISJ7 | TRISJ6 | TRISJ5 | TRISJ4 | TRISJ3 | TRISJ2 | TRISJ1 | TRISJ0 |
| PADCFG1 | RDPU | REPU | RJPU(1) | — | — | RTSECSEL1 | RTSECSEL0 | — |

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

**Note 1:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.

## 12.11 Parallel Slave Port

PORTD can function as an 8-bit-wide Parallel Slave Port (PSP), or microprocessor port, when control bit, PSPMODE (PSPCON<4>), is set. The port is asynchronously readable and writable by the external world through the $\overline{RD}$ control input pin (RE0/P2D/$\overline{RD}$/AD8) and $\overline{WR}$ control input pin (RE1/P2C/$\overline{WR}$/AD9).

| **Note:** | The Parallel Slave Port is available only in Microcontroller mode. |
|---|---|

The PSP can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an eight-bit latch.

Setting bit, PSPMODE, enables port pin, RE0/P2D/$\overline{RD}$/AD8, to be the $\overline{RD}$ input, RE1/P2C/$\overline{WR}$/AD9 to be the $\overline{WR}$ input and RE2/P2B/CCP10/$\overline{CS}$/AD10 to be the $\overline{CS}$ (Chip Select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (= 111).

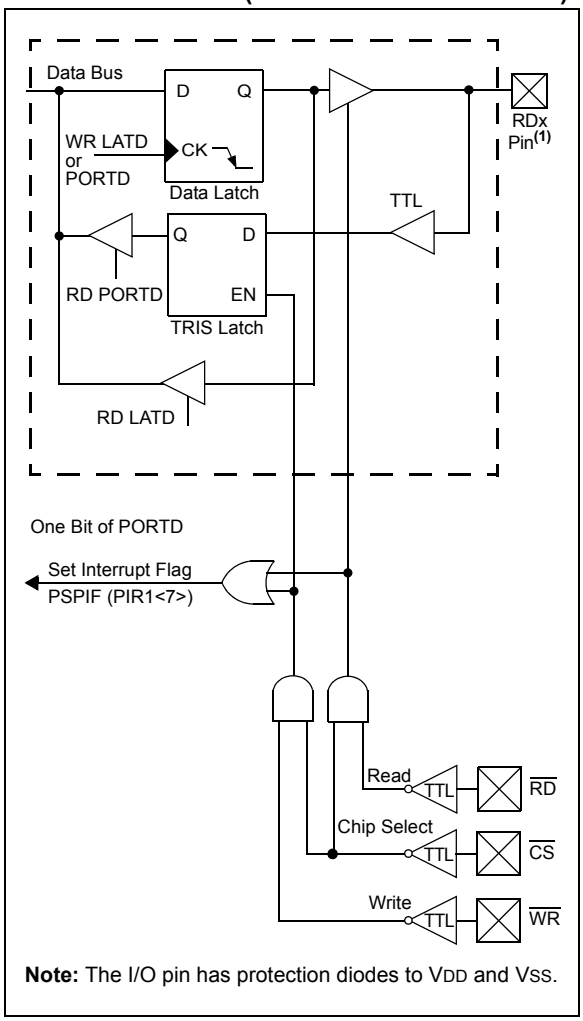
A write to the PSP occurs when both the $\overline{CS}$ and $\overline{WR}$ lines are first detected low and ends when either are detected high. The PSPIF and IBF flag bits (PIR1<7> and PSPCON<7>, respectively) are set when the write ends.

A read from the PSP occurs when both the $\overline{CS}$ and $\overline{RD}$ lines are first detected low. The data in PORTD is read out and the OBF bit (PSPCON<6>) is set. If the user writes new data to PORTD to set OBF, the data is immediately read out, but the OBF bit is not set.

When either the $\overline{CS}$ or $\overline{RD}$ line is detected high, the PORTD pins return to the input state and the PSPIF bit is set. User applications should wait for PSPIF to be set before servicing the PSP. When this happens, the IBF and OBF bits can be polled and the appropriate action taken.

The timing for the control signals in Write and Read modes is shown in Figure 12-4 and Figure 12-5, respectively.

**FIGURE 12-3: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)**




**Note:** The I/O pin has protection diodes to VDD and VSS.


© 2009-2018 Microchip Technology Inc.

**REGISTER 12-5: PSPCON: PARALLEL SLAVE PORT CONTROL REGISTER**

| R-0 | R-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| IBF | OBF | IBOV | PSPMODE | — | — | — | — |
| 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 = A word has been received and is waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF:** Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV:** Input Buffer Overflow Detect bit
1 = A write occurred when a previously input word had not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE:** Parallel Slave Port Mode Select bit
1 = Parallel Slave Port mode
0 = General Purpose I/O mode

bit 3-0 **Unimplemented:** Read as '0'

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

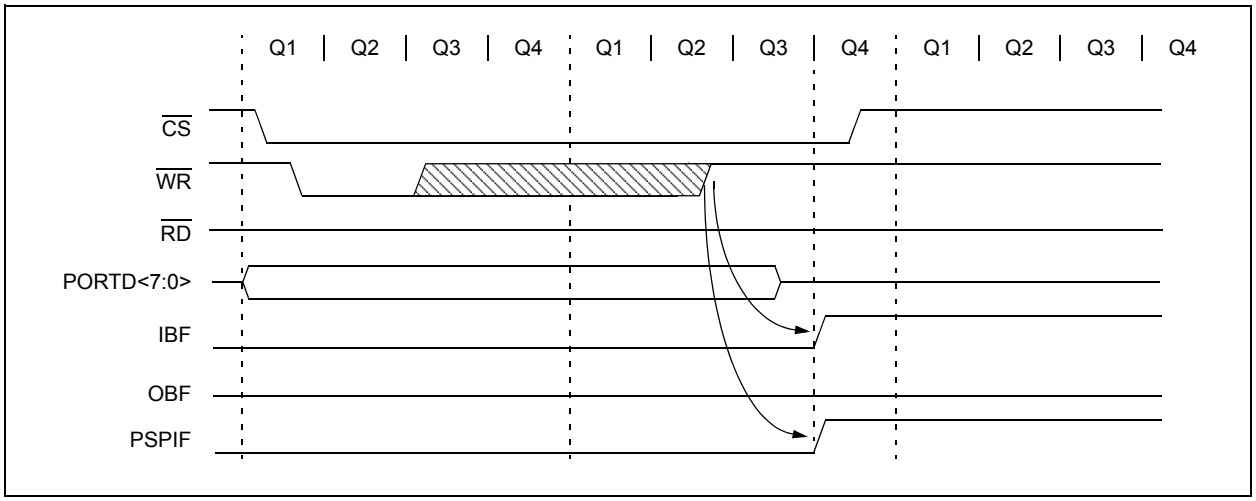


**FIGURE 12-5: PARALLEL SLAVE PORT READ WAVEFORMS**

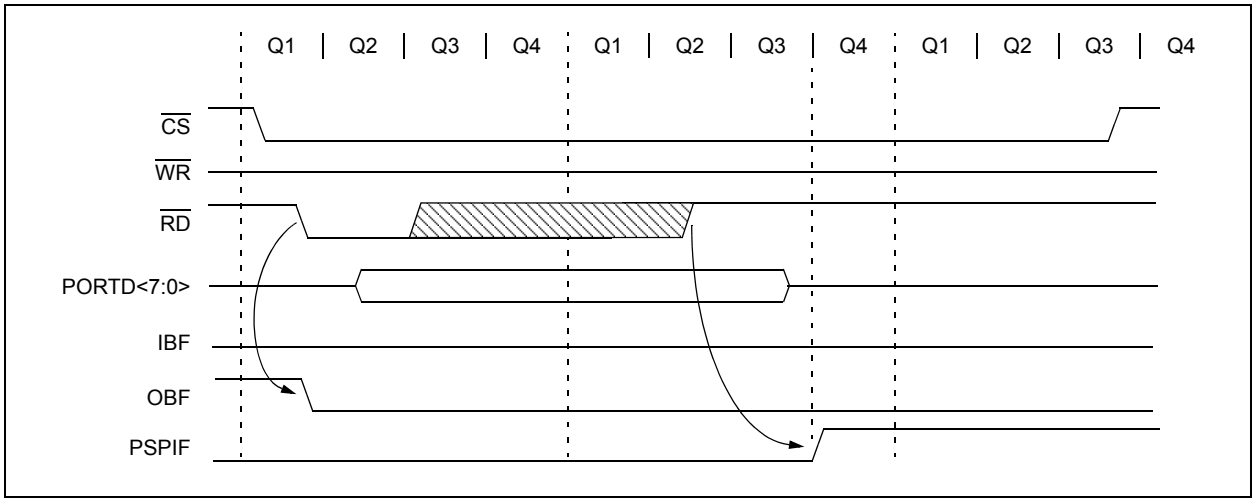



**TABLE 12-19: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 |
| LATD | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 |
| PORTE | RE7 | RE6 | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 |
| LATE | LATE7 | LATE6 | LATE5 | LATE4 | LATE3 | LATE2 | LATE1 | LATE0 |
| TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 |
| PSPCON | IBF | OBF | IBOV | PSPMODE | — | — | — | — |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| PMD1 | PSPMD | CTMUMD | RTCCMD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | EMBDM |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Parallel Slave Port.


© 2009-2018 Microchip Technology Inc.

## 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.

Figure 13-1 provides a simplified block diagram of the Timer0 module in 8-bit mode. Figure 13-2 provides 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 = Increment on high-to-low transition on T0CKI pin
0 = Increment 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

© 2009-2018 Microchip Technology Inc.

## 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 edge or falling edge of the T0CKI pin. 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.

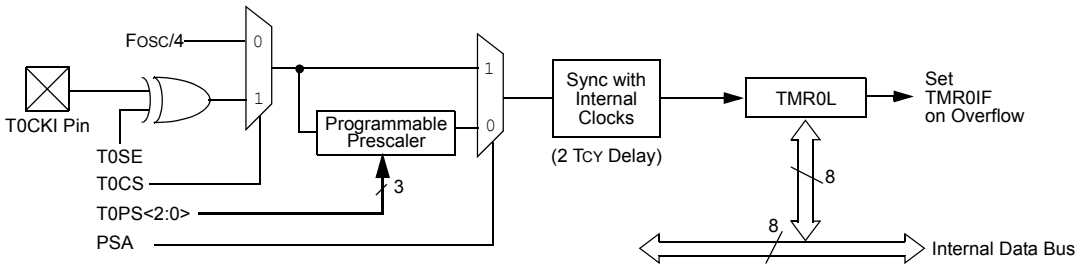
An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (TOSC). There is a delay between synchronization and the onset of incrementing the timer/counter.

## 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 (see 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 was valid, due to a rollover between successive reads of the high and low byte.

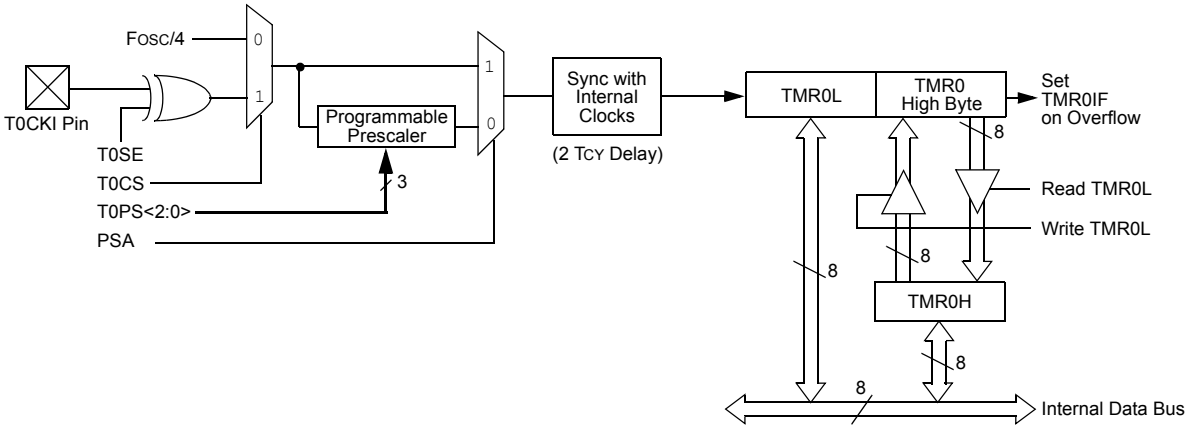
Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

**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.


© 2009-2018 Microchip Technology Inc.

## 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-two increments, are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (for example, `CLRF TMR0`, `MOVWF TMR0`, `BSF TMR0`) 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 (ISR).

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 |
|---|---|---|---|---|---|---|---|---|
| TMR0L | Timer0 Register Low Byte | | | | | | | |
| TMR0H | Timer0 Register High Byte | | | | | | | |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| T0CON | TMR0ON | T08BIT | T0CS | T0SE | PSA | T0PS2 | T0PS1 | T0PS0 |

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used by Timer0.

## 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 eight-bit registers (TMR1H and TMR1L)
- Selectable clock source (internal or external) with device clock or SOSC oscillator internal options
- Interrupt-on-overflow
- Reset on ECCP Special Event Trigger
- Timer with gated control

Figure 14-1 displays a simplified block diagram of the Timer1 module.

The Timer1 oscillator can also be used as a low-power clock source for the microcontroller in power-managed operation. The Timer1 can also work on the SOSC oscillator.

Timer1 is controlled through the T1CON Control register (Register 14-1). It also contains the Secondary Oscillator Enable bit (SOSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

The FOSC clock source should not be used with the ECCP capture/compare features. If the timer will be used with the capture or compare features, always select one of the other timer clocking options.

**REGISTER 14-1: T1CON: TIMER1 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 |
|---|---|---|---|---|---|---|---|
| TMR1CS1 | TMR1CS0 | T1CKPS1 | T1CKPS0 | SOSCEN | $\overline{\text{T1SYNC}}$ | RD16 | 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 7-6 **TMR1CS<1:0>:** Timer1 Clock Source Select bits
10 = Timer1 clock source is either from a pin or oscillator, depending on the SOSCEN bit:
SOSCEN = 0:
External clock from the T1CKI pin (on the rising edge).
SOSCEN = 1:
Depending on the SOSCSEL Configuration bit, the clock source is either a crystal oscillator on the SOSCI/SOSCO pins or an internal clock from the SCLKI pin.
01 = Timer1 clock source is the system clock (FOSC)[1]
00 = Timer1 clock source is the instruction clock (FOSC/4)

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

bit 3 **SOSCEN:** SOSC Oscillator Enable bit
1 = SOSC is enabled and available for Timer1
0 = SOSC is disabled for Timer1
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
TMR1CS<1:0> = 10:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
TMR1CS<1:0> = 0x:
This bit is ignored. Timer1 uses the internal clock when TMR1CS<1:0> = 1x.

bit 1 **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

**Note 1:** The FOSC clock source should not be selected if the timer will be used with the ECCP capture/compare features.


© 2009-2018 Microchip Technology Inc.

**REGISTER 14-1: T1CON: TIMER1 CONTROL REGISTER (CONTINUED)**

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

**Note 1:** The FOSC clock source should not be selected if the timer will be used with the ECCP capture/compare features.

## 14.1 Timer1 Gate Control Register

The Timer1 Gate Control register (T1GCON), displayed in Register 14-2, is used to control the Timer1 gate.

**REGISTER 14-2: T1GCON: TIMER1 GATE CONTROL REGISTER(1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-x | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GGO/$\overline{\text{T1DONE}}$ | T1GVAL | T1GSS1 | T1GSS0 |
| 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 **TMR1GE:** Timer1 Gate Enable bit

If TMR1ON = 0:
This bit is ignored.

If TMR1ON = 1:
1 = Timer1 counting is controlled by the Timer1 gate function
0 = Timer1 counts regardless of Timer1 gate function

bit 6 **T1GPOL:** Timer1 Gate Polarity bit

1 = Timer1 gate is active-high (Timer1 counts when gate is high)
0 = Timer1 gate is active-low (Timer1 counts when gate is low)

bit 5 **T1GTM:** Timer1 Gate Toggle Mode bit

1 = Timer1 Gate Toggle mode is enabled
0 = Timer1 Gate Toggle mode is disabled and toggle flip-flop is cleared
Timer1 gate flip-flop toggles on every rising edge.

bit 4 **T1GSPM:** Timer1 Gate Single Pulse Mode bit

1 = Timer1 Gate Single Pulse mode is enabled and is controlling Timer1 gate
0 = Timer1 Gate Single Pulse mode is disabled

bit 3 **T1GGO/$\overline{\text{T1DONE}}$:** Timer1 Gate Single Pulse Acquisition Status bit

1 = Timer1 gate single pulse acquisition is ready, waiting for an edge
0 = Timer1 gate single pulse acquisition has completed or has not been started
This bit is automatically cleared when T1GSPM is cleared.

bit 2 **T1GVAL:** Timer1 Gate Current State bit

Indicates the current state of the Timer1 gate that could be provided to TMR1H:TMR1L; unaffected by Timer1 Gate Enable (TMR1GE) bit.

bit 1-0 **T1GSS<1:0>:** Timer1 Gate Source Select bits

11 = Comparator 2 output
10 = Comparator 1 output
01 = TMR2 to match PR2 output
00 = Timer1 gate pin

**Note 1:** Programming the T1GCON prior to T1CON is recommended.


© 2009-2018 Microchip Technology Inc.

## 14.2 Timer1 Operation

The Timer1 module is an 8 or 16-bit incrementing counter that is accessed through the TMR1H:TMR1L register pair.

When used with an internal clock source, the module is a timer and increments on every instruction cycle. When used with an external clock source, the module can be used as either a timer or counter. It increments on every selected edge of the external source.

Timer1 is enabled by configuring the TMR1ON and TMR1GE bits in the T1CON and T1GCON registers, respectively.

When SOSC is selected as Crystal mode (by SOSCSEL), the RC1/SOSCI/ECCP2/P2A and RC0/SOSCO/SCLKI pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as '0'.

## 14.3 Clock Source Selection

The TMR1CS<1:0> and SOSCEN bits of the T1CON register are used to select the clock source for Timer1. Register 14-1 displays the clock source selections.

### 14.3.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected, the TMR1H:TMR1L register pair will increment on multiples of FOSC, as determined by the Timer1 prescaler.

### 14.3.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1 module may work as a timer or a counter.

When enabled to count, Timer1 is incremented on the rising edge of the external clock input, T1CKI. Either of these external clock sources can be synchronized to the microcontroller system clock or they can run asynchronously.

When used as a timer with a clock oscillator, an external, 32.768 kHz crystal can be used in conjunction with the dedicated internal oscillator circuit.

**Note:** In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:

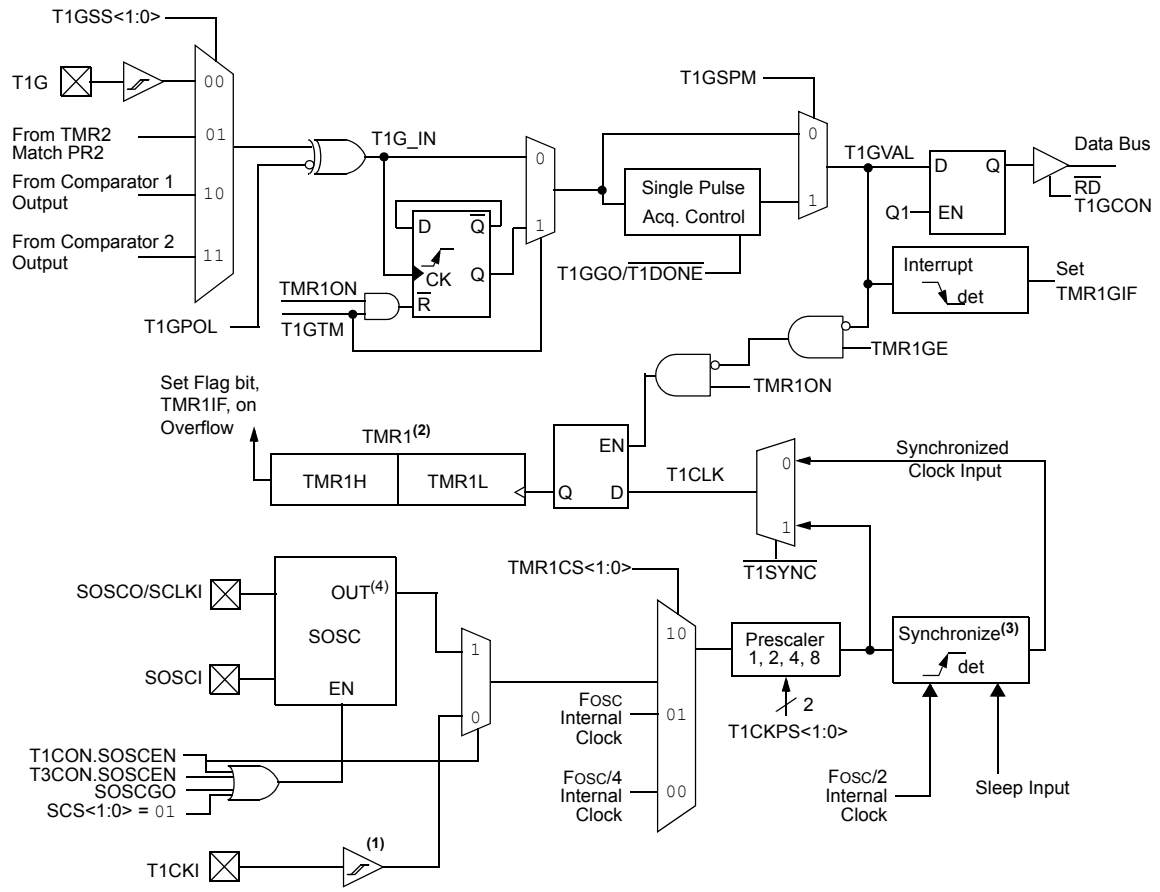
- Timer1 enabled after POR Reset
- Write to TMR1H or TMR1L
- Timer1 is disabled
- Timer1 is disabled (TMR1ON = 0)

When T1CKI is high, Timer1 is enabled (TMR1ON = 1) when T1CKI is low.

**TABLE 14-1: TIMER1 CLOCK SOURCE SELECTION**

| TMR1CS1 | TMR1CS0 | SOSCEN | Clock Source |
|---|---|---|---|
| 0 | 1 | x | Clock Source (FOSC) |
| 0 | 0 | x | Instruction Clock (FOSC/4) |
| 1 | 0 | 0 | External Clock on T1CKI Pin |
| 1 | 0 | 1 | Oscillator Circuit on SOSCI/SOSCO Pins |

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




**Note 1:** ST Buffer is a high-speed type when using T1CKI.
**2:** Timer1 register increments on rising edge.
**3:** Synchronization does not operate while in Sleep.
**4:** The output of SOSC is determined by the SOSCSEL<1:0> Configuration bits.


© 2009-2018 Microchip Technology Inc.

## 14.4 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes. When the RD16 control bit (T1CON<1>) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L loads 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 at once to both the high and low bytes of Timer1.

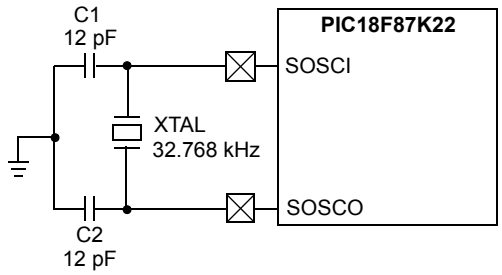
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler; the prescaler is only cleared on writes to TMR1L.

## 14.5 SOSC Oscillator

An on-chip crystal oscillator circuit is incorporated between pins, SOSCI (input) and SOSCO (amplifier output). It is enabled by any peripheral that requests it. There are eight ways the SOSC can be enabled: if the SOSC is selected as the source by any of the odd timers, which is done by each respective SOSCEN bit (TxCON<3>), if the SOSC is selected as the RTCC source by the RTCOSC Configuration bit (CONFIG3L<1>), if the SOSC is selected as the CPU clock source by the SCS bits (OSCCON<1:0>) or if the SOSCGO bit is set (OSCCON2<3>). The SOSCGO bit is used to warm up the SOSC so that it is ready before any peripheral requests it. 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 low-power oscillator is depicted in Figure 14-2. Table 14-2 provides the capacitor selection for the SOSC oscillator.

The user must provide a software time delay to ensure proper start-up of the SOSC oscillator.

**FIGURE 14-2: EXTERNAL COMPONENTS FOR THE SOSC LOW-POWER OSCILLATOR**




**Note:** See the Notes with Table 14-2 for additional information about capacitor selection.

**TABLE 14-2: CAPACITOR SELECTION FOR THE TIMER OSCILLATOR(2,3,4,5)**

| Oscillator Type | Freq. | C1 | C2 |
|---|---|---|---|
| LP | 32 kHz | 12 pF(1) | 12 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. Values listed would be typical of a CL = 10 pF rated crystal when SOSCSEL<1:0> = 11.

**5:** Incorrect capacitance value may result in a frequency not meeting the crystal manufacturer's tolerance specification.

The SOSC crystal oscillator drive level is determined based on the SOSCSEL<1:0> (CONFIG1L<4:3>) Configuration bits. The Higher Drive Level mode, SOSCSEL<1:0> = 11, is intended to drive a wide variety of 32.768 kHz crystals with a variety of Capacitance Load (CL) ratings.

The Lower Drive Level mode is highly optimized for extremely low-power consumption. It is not intended to drive all types of 32.768 kHz crystals. In the Low Drive Level mode, the crystal oscillator circuit may not work correctly if excessively large discrete capacitors are placed on the SOSCO and SOSCI pins. This mode is designed to work only with discrete capacitances of approximately 3 pF-10 pF on each pin.

Crystal manufacturers usually specify a CL (Capacitance Load) rating for their crystals. This value is related to, but not necessarily the same as, the values that should be used for C1 and C2 in Figure 14-2.

For more details on selecting the optimum C1 and C2 for a given crystal, see the crystal manufacturer's applications information. The optimum value depends, in part, on the amount of parasitic capacitance in the circuit, which is often unknown. For that reason, it is highly recommended that thorough testing and validation of the oscillator be performed after values have been selected.

### 14.5.1 USING SOSC AS A CLOCK SOURCE

The SOSC oscillator is also available as a clock source in power-managed modes. By setting the System Clock Select bits, SCS<1:0> (OSCCON<1:0>), to '01', the device switches to SEC_RUN mode, and both the CPU and peripherals are clocked from the SOSC 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 SOSC oscillator is providing the clock source, the SOSC System Clock Status flag, SOSCRUN (OSCCON2<6>), is set. This can be used to determine the controller's current clocking mode. It can also indicate the clock source currently being used by the Fail-Safe Clock Monitor (FSCM).

If the Clock Monitor is enabled and the SOSC oscillator fails while providing the clock, polling the SOCSRUN bit will indicate whether the clock is being provided by the SOSC oscillator or another source.

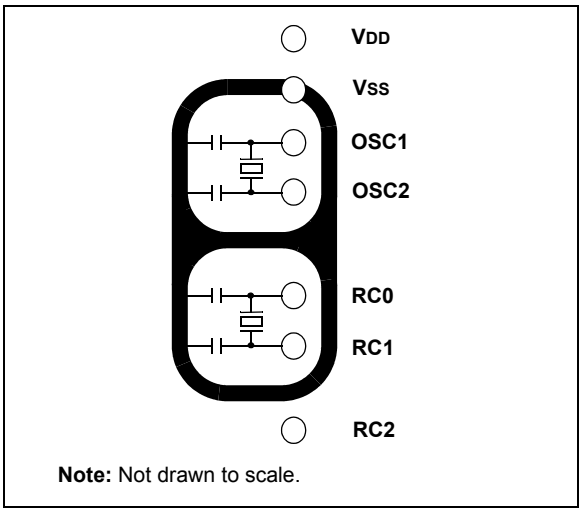
### 14.5.2 SOSC OSCILLATOR LAYOUT CONSIDERATIONS

The SOSC 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. This is especially true when the oscillator is configured for extremely Low-Power mode, SOSCSEL<1:0> (CONFIG1L<4:3>) = 01.

The oscillator circuit, displayed in Figure 14-2, 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, it may help to have a grounded guard ring around the oscillator circuit. The guard, as displayed in Figure 14-3, could be used on a single-sided PCB or in addition to a ground plane. (Examples of a high-speed circuit include the ECCP1 pin, in Output Compare or PWM mode, or the primary oscillator using the OSC2 pin.)

**FIGURE 14-3: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING**




**Note:** Not drawn to scale.

In the Low Drive Level mode, SOSCSEL<1:0> = 01, it is critical that RC2 I/O pin signals be kept away from the oscillator circuit. Configuring RC2 as a digital output, and toggling it, can potentially disturb the oscillator circuit, even with a relatively good PCB layout. If possible, either leave RC2 unused or use it as an input pin with a slew rate limited signal source. If RC2 must be used as a digital output, it may be necessary to use the Higher Drive Level Oscillator mode (SOSCSEL<1:0> = 11) with many PCB layouts.

Even in the Higher Drive Level mode, careful layout procedures should still be followed when designing the oscillator circuit.

In addition to dV/dt induced noise considerations, it is important to ensure that the circuit board is clean. Even a very small amount of conductive soldering flux residue can cause PCB leakage currents that can overwhelm the oscillator circuit.

## 14.6 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>).


© 2009-2018 Microchip Technology Inc.

## 14.7 Resetting Timer1 Using the ECCP Special Event Trigger

If ECCP modules are configured to use Timer1 and to generate a Special Event Trigger in Compare mode (CCPxM<3:0> = 1011), this signal will reset Timer1. The trigger from ECCP2 will also start an A/D conversion, if the A/D module is enabled. (For more information, see **Section 20.3.4 "Special Event Trigger"**.)

To take advantage of this feature, the module must be configured as either a timer or a synchronous counter. 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 Trigger from the ECCPx module will only clear the TMR1 register's content, but not set the TMR1IF interrupt flag bit (PIR1<0>). |
|---|---|

## 14.8 Timer1 Gate

Timer1 can be configured to count freely or the count can be enabled and disabled using the Timer1 gate circuitry. This is also referred to as Timer1 gate count enable.

Timer1 gate can also be driven by multiple selectable sources.

### 14.8.1 TIMER1 GATE COUNT ENABLE

The Timer1 Gate Enable mode is enabled by setting the TMR1GE bit of the T1GCON register. The polarity of the Timer1 Gate Enable mode is configured using the T1GPOL bit (T1GCON<6>).

When Timer1 Gate Enable mode is enabled, Timer1 will increment on the rising edge of the Timer1 clock source. When Timer1 Gate Enable mode is disabled, no incrementing will occur and Timer1 will hold the current count. See Figure 14-4 for timing details.

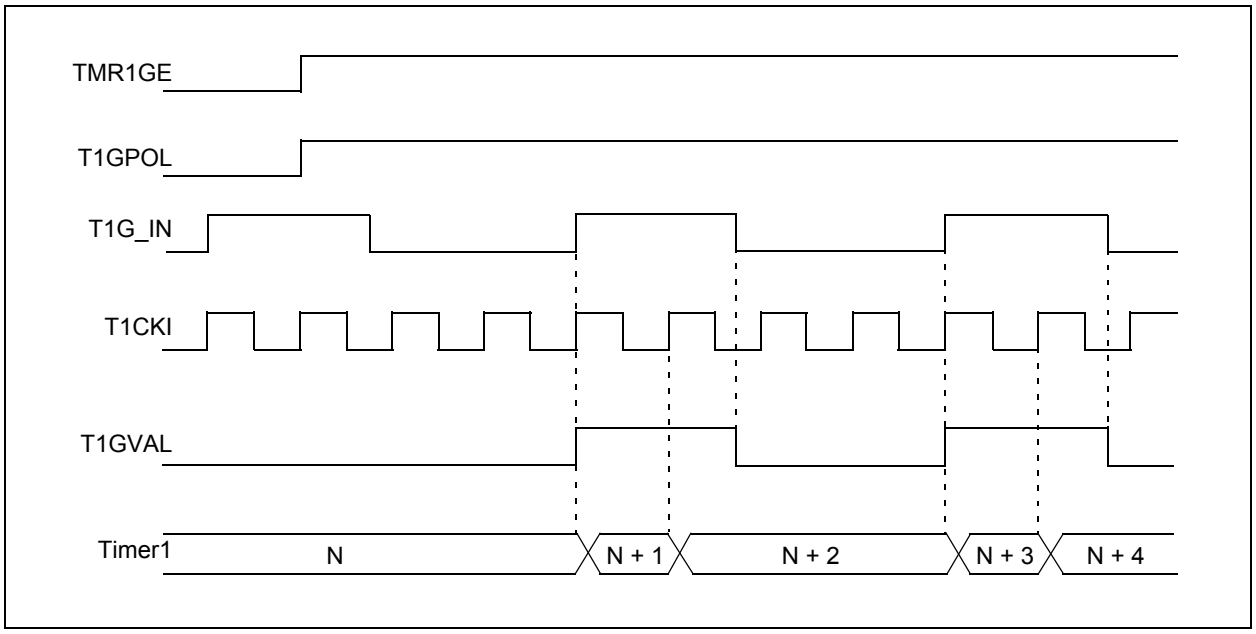
**TABLE 14-3: TIMER1 GATE ENABLE SELECTIONS**

| T1CLK(†) | T1GPOL (T1GCON<6>) | T1G Pin | Timer1 Operation |
|---|---|---|---|
| ↑ | 0 | 0 | Counts |
| ↑ | 0 | 1 | Holds Count |
| ↑ | 1 | 0 | Holds Count |
| ↑ | 1 | 1 | Counts |

† The clock on which TMR1 is running. For more information, see Figure 14-1.

| **Note:** | The CCP and ECCP modules use Timers, 1 through 8, for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRSx registers. For more details, see Register 19-2, Register 19-3 and Register 20-2. |
|---|---|

**FIGURE 14-4: TIMER1 GATE COUNT ENABLE MODE**




### 14.8.2 TIMER1 GATE SOURCE SELECTION

The Timer1 gate source can be selected from one of four sources. Source selection is controlled by the T1GSSx (T1GCON<1:0>) bits (see Table 14-4).

**TABLE 14-4: TIMER1 GATE SOURCES**

| T1GSS<1:0> | Timer1 Gate Source |
|---|---|
| 00 | Timer1 Gate Pin |
| 01 | TMR2 to Match PR2 (TMR2 increments to match PR2) |
| 10 | Comparator 1 Output (comparator logic high output) |
| 11 | Comparator 2 Output (comparator logic high output) |

The polarity for each available source is also selectable, controlled by the T1GPOL bit (T1GCON<6>).

#### 14.8.2.1 T1G Pin Gate Operation

The T1G pin is one source for Timer1 gate control. It can be used to supply an external source to the Timer1 gate circuitry.

#### 14.8.2.2 Timer2 Match Gate Operation

The TMR2 register will increment until it matches the value in the PR2 register. On the very next increment cycle, TMR2 will be reset to 00h. When this Reset occurs, a low-to-high pulse will automatically be generated and internally supplied to the Timer1 gate circuitry. The pulse will remain high for one instruction cycle and will return back to a low state until the next match.

The T1GPOL bit determines when the Timer1 counter increments based on this pulse. When T1GPOL = 1, Timer1 increments for a single instruction cycle following a TMR2 match with PR2. When T1GPOL = 0, Timer1 increments continuously, except for the cycle following the match, when the gate signal goes from low-to-high.

#### 14.8.2.3 Comparator 1 Output Gate Operation

The output of Comparator 1 can be internally supplied to the Timer1 gate circuitry. After setting up Comparator 1 with the CM1CON register, Timer1 will increment depending on the transitions of the CMP1OUT (CMSTAT<5>) bit.

#### 14.8.2.4 Comparator 2 Output Gate Operation

The output of Comparator 2 can be internally supplied to the Timer1 gate circuitry. After setting up Comparator 2 with the CM2CON register, Timer1 will increment depending on the transitions of the CMP2OUT (CMSTAT<6>) bit.


© 2009-2018 Microchip Technology Inc.

### 14.8.3 TIMER1 GATE TOGGLE MODE

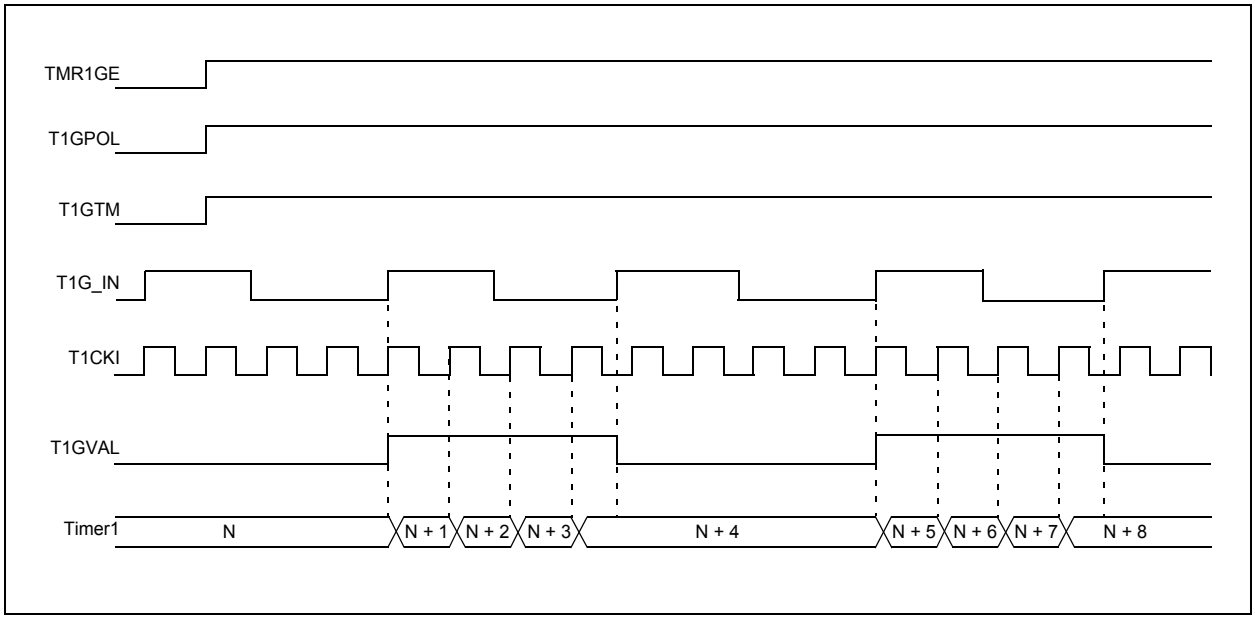
When Timer1 Gate Toggle mode is enabled, it is possible to measure the full cycle length of a Timer1 gate signal, as opposed to the duration of a single level pulse.

The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. (For timing details, see Figure 14-5.)

The T1GVAL bit (T1GCON<2>) indicates when the Toggled mode is active and the timer is counting.

The Timer1 Gate Toggle mode is enabled by setting the T1GTM bit (T1GCON<5>). When T1GTM is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

**FIGURE 14-5: TIMER1 GATE TOGGLE MODE**



### 14.8.4 TIMER1 GATE SINGLE PULSE MODE

When Timer1 Gate Single Pulse mode is enabled, it is possible to capture a single pulse gate event. Timer1 Gate Single Pulse mode is enabled by setting the T1GSPM bit (T1GCON<4>) and the T1GGO/$\overline{\text{T1DONE}}$ bit (T1GCON<3>). The Timer1 will be fully enabled on the next incrementing edge.

On the next trailing edge of the pulse, the T1GGO/$\overline{\text{T1DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the T1GGO/$\overline{\text{T1DONE}}$ bit is once again set in software.

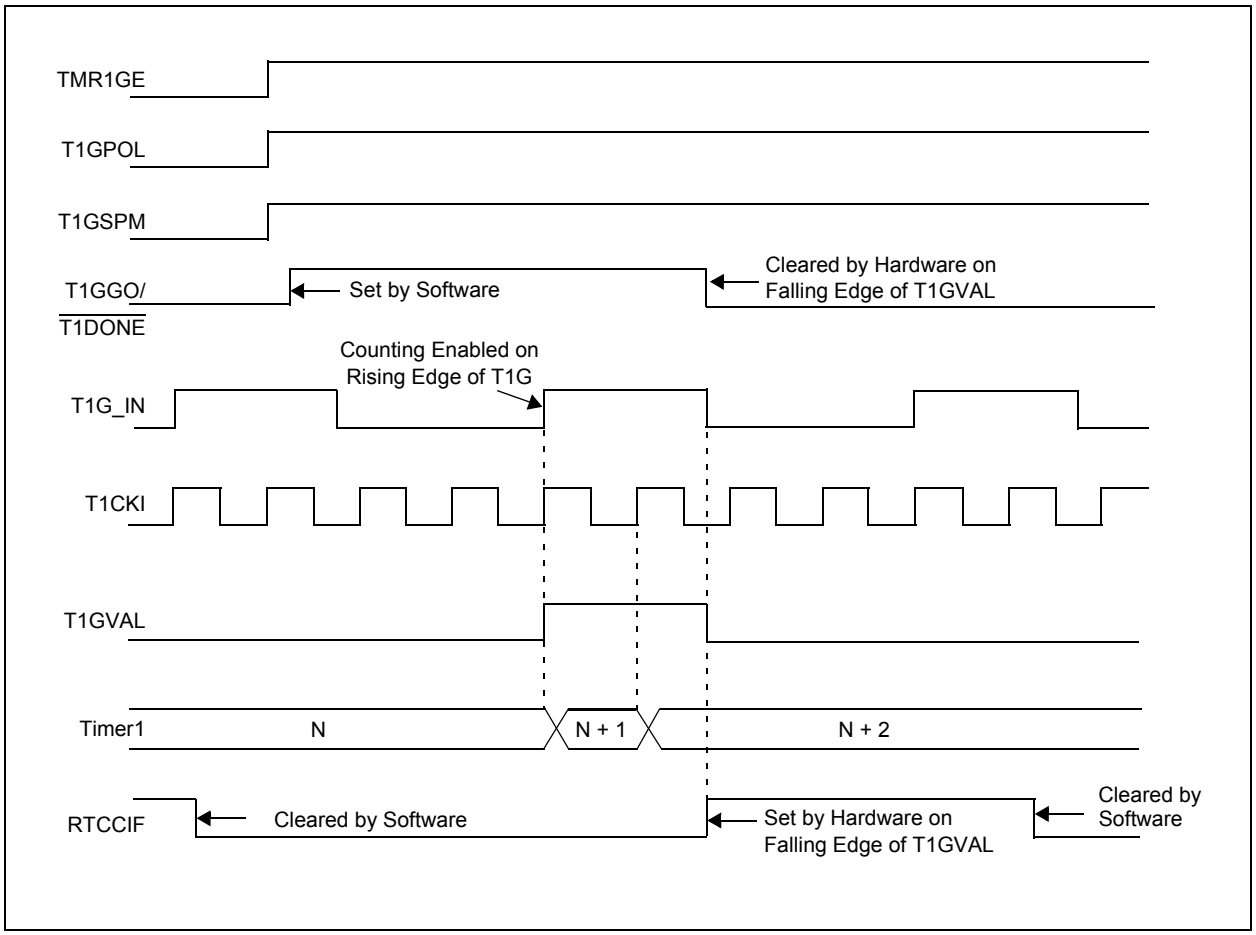
Clearing the T1GSPM bit of the T1GCON register will also clear the T1GGO/$\overline{\text{T1DONE}}$ bit. (For timing details, see Figure 14-6.)

Simultaneously enabling the Toggle and Single Pulse modes will permit both sections to work together. This allows the cycle times on the Timer1 gate source to be measured. (For timing details, see Figure 14-7.)

### 14.8.5 TIMER1 GATE VALUE STATUS

When the Timer1 gate value status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the T1GVAL bit (T1GCON<2>). This bit is valid even when the Timer1 gate is not enabled (TMR1GE bit is cleared).

**FIGURE 14-6: TIMER1 GATE SINGLE PULSE MODE**





© 2009-2018 Microchip Technology Inc.

**FIGURE 14-7: TIMER1 GATE SINGLE PULSE AND TOGGLE COMBINED MODE**

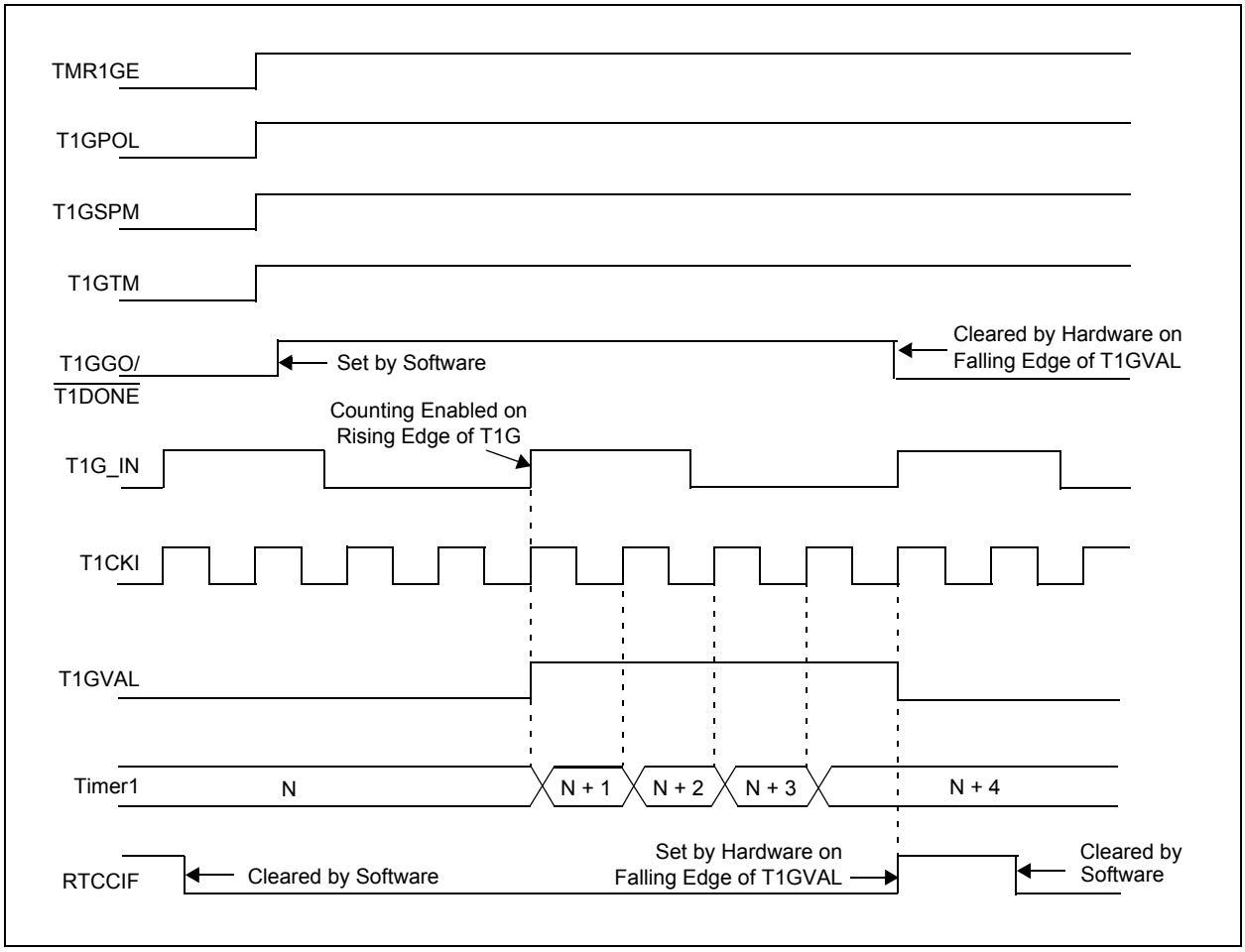



**TABLE 14-5: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| TMR1L | Timer1 Register Low Byte | | | | | | | |
| TMR1H | Timer1 Register High Byte | | | | | | | |
| T1CON | TMR1CS1 | TMR1CS0 | T1CKPS1 | T1CKPS0 | SOSCEN | $\overline{\text{T1SYNC}}$ | RD16 | TMR1ON |
| T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | $\overline{\text{T1GGO}}$/$\overline{\text{T1DONE}}$ | T1GVAL | T1GSS1 | T1GSS0 |
| OSCCON2 | — | SOSCRUN | — | — | SOSCGO | — | MFIOFS | MFIOSEL |
| PMD1 | PSPMD | CTMUMD | RTCCMD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | EMBDM |

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

**Note 1:** Unimplemented on 32-Kbyte devices (PIC18FX5K22).

## 15.0 TIMER2 MODULE

The Timer2 module incorporates the following features:

- Eight-bit Timer and Period registers (TMR2 and PR2, respectively)
- Both registers are readable and writable
- 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

This module is controlled through the T2CON register (Register 15-1) that 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 four-bit counter/prescaler on the clock input gives the prescale options of direct input, divide-by-4 or divide-by-16. 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 (POR), MCLR Reset, Watchdog Timer Reset (WDTR) or Brown-out Reset (BOR)

TMR2 is not cleared when T2CON is written.

| **Note:** | The CCP and ECCP modules use Timers, 1 through 8, for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRSx registers. For more details, see Register 20-2, Register 19-2 and Register 19-3. |
|---|---|

**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


© 2009-2018 Microchip Technology Inc.

## 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 postscaler 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 ECCP modules, where it is used as a time base for operations in PWM mode.

Timer2 can optionally be used as the shift clock source for the MSSP modules operating in SPI mode. Additional information is provided in **Section 21.0 "Master Synchronous Serial Port (MSSP) Module"**.

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

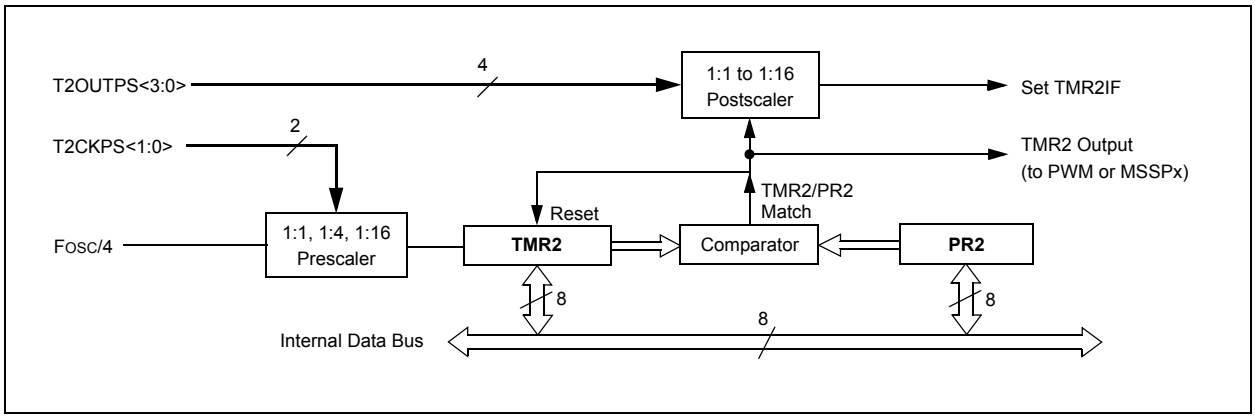



**TABLE 15-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| TMR2 | Timer2 Register | | | | | | | |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| PR2 | Timer2 Period Register | | | | | | | |
| PMD1 | PSPMD | CTMUMD | RTCCMD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | EMBDM |

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

## 16.0 TIMER3/5/7 MODULES

The Timer3/5/7 timer/counter modules incorporate these features:

- Software-selectable operation as a 16-bit timer or counter
- Readable and writable eight-bit registers (TMRxH and TMRxL)
- Selectable clock source (internal or external) with device clock or SOSC oscillator internal options
- Interrupt-on-overflow
- Module Reset on ECCP Special Event Trigger

Timer7 is unimplemented for devices with a program memory of 32 Kbytes (PIC18FX5K22).

**Note:** Throughout this section, generic references are used for register and bit names that are the same – except for an 'x' variable that indicates the item's association with the Timer3, Timer5 or Timer7 module. For example, the control register is named TxCON and refers to T3CON, T5CON and T7CON.

A simplified block diagram of the Timer3/5/7 module is shown in Figure 16-1.

The Timer3/5/7 module is controlled through the TxCON register (Register 16-1). It also selects the clock source options for the ECCP modules. (For more information, see **Section 20.1.1 "ECCP Module and Timer Resources"**.)

The FOSC clock source should not be used with the ECCP capture/compare features. If the timer will be used with the capture or compare features, always select one of the other timer clocking options.


© 2009-2018 Microchip Technology Inc.

**REGISTER 16-1: TxCON: TIMERx 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 |
|---|---|---|---|---|---|---|---|
| TMRxCS1 | TMRxCS0 | TxCKPS1 | TxCKPS0 | SOSCEN | $\overline{\text{TxSYNC}}$ | RD16 | TMRxON |
| 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 **TMRxCS<1:0>:** Timerx Clock Source Select bits

10 = Timer1 clock source depends on the SOSCEN bit:

SOSCEN = 0:

External clock from the T1CKI pin (on the rising edge).

SOSCEN = 1:

Depending on the SOSCSEL fuses, either a crystal oscillator on the SOSCI/SOSCO pins or an external clock from the SCLKI pin.

01 = Timerx clock source is the system clock (FOSC)[(1)]

00 = Timerx clock source is the instruction clock (FOSC/4)

bit 5-4 **TxCKPS<1:0>**: Timerx 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 **SOSCEN**: SOSC Oscillator Enable bit

1 = SOSC/SCLKI are enabled for Timerx (based on the SOSCSEL fuses)

0 = SOSC/SCLKI are disabled for Timerx and TxCKI is enabled

bit 2 **$\overline{\text{TxSYNC}}$:** Timerx External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)

When TMRxCS<1:0> = 10:

1 = Do not synchronize external clock input

0 = Synchronize external clock input

When TMRxCS<1:0> = 0x:

This bit is ignored; Timer3 uses the internal clock.

bit 1 **RD16:** 16-Bit Read/Write Mode Enable bit

1 = Enables register read/write of Timerx in one 16-bit operation

0 = Enables register read/write of Timerx in two eight-bit operations

bit 0 **TMRxON:** Timerx On bit

1 = Enables Timerx

0 = Stops Timerx

**Note 1:** The FOSC clock source should not be selected if the timer will be used with the ECCP capture/compare features.

## 16.1 Timer3/5/7 Gate Control Register

The Timer3/5/7 Gate Control register (TxGCON), provided in Register 14-2, is used to control the Timerx gate.

**REGISTER 16-2: TxGCON: TIMERx GATE CONTROL REGISTER(1)**

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R-x | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| TMRxGE | TxGPOL | TxGTM | TxGSPM | TxGGO/$\overline{\text{TxDONE}}$ | TxGVAL | TxGSS1 | TxGSS0 |
| 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 **TMRxGE:** Timerx Gate Enable bit

If TMRxON = 0:
This bit is ignored.

If TMRxON = 1:
1 = Timerx counting is controlled by the Timerx gate function
0 = Timerx counts regardless of Timerx gate function

bit 6 **TxGPOL:** Timerx Gate Polarity bit

1 = Timerx gate is active-high (Timerx counts when gate is high)
0 = Timerx gate is active-low (Timerx counts when gate is low)

bit 5 **TxGTM:** Timerx Gate Toggle Mode bit

1 = Timerx Gate Toggle mode is enabled.
0 = Timerx Gate Toggle mode is disabled and toggle flip-flop is cleared
Timerx gate flip-flop toggles on every rising edge.

bit 4 **TxGSPM:** Timerx Gate Single Pulse Mode bit

1 = Timerx Gate Single Pulse mode is enabled and is controlling Timerx gate
0 = Timerx Gate Single Pulse mode is disabled

bit 3 **TxGGO/$\overline{\text{TxDONE}}$:** Timerx Gate Single Pulse Acquisition Status bit

1 = Timerx gate single pulse acquisition is ready, waiting for an edge
0 = Timerx gate single pulse acquisition has completed or has not been started
This bit is automatically cleared when TxGSPM is cleared.

bit 2 **TxGVAL:** Timerx Gate Current State bit

Indicates the current state of the Timerx gate that could be provided to TMRxH:TMRxL. Unaffected by the Timerx Gate Enable (TMRxGE) bit.

bit 1-0 **TxGSS<1:0>:** Timerx Gate Source Select bits

11 = Comparator 2 output
10 = Comparator 1 output
01 = TMR(x+1) to match PR(x+1) output(2)
00 = Timer1 gate pin
The Watchdog Timer oscillator is turned on if TMRxGE = 1, regardless of the state of TMRxON.

**Note 1:** Programming the TxGCON prior to TxCON is recommended.
**2:** Timer(x+1) will be Timer4/6/8 for Timerx (Timer3/5/7), respectively.


© 2009-2018 Microchip Technology Inc.

**REGISTER 16-3: OSCCON2: OSCILLATOR CONTROL REGISTER 2**

| U-0 | R-0 | U-0 | U-0 | R/W-0 | U-0 | R-x | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | SOSCRUN | — | — | SOSCGO | — | MFIOFS | MFIOSEL |
| 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 **SOSCRUN:** SOSC Run Status bit
- 1 = System clock comes from a secondary SOSC
- 0 = System clock comes from an oscillator other than SOSC

bit 5-4 **Unimplemented:** Read as '0'

bit 3 **SOSCGO:** Oscillator Start Control bit
- 1 = Oscillator is running even if no other sources are requesting it
- 0 = Oscillator is shut off if no other sources are requesting it (When the SOSC is selected to run from a digital clock input, rather than an external crystal, this bit has no effect.)

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

bit 1 **MFIOFS:** MF-INTOSC Frequency Stable bit
- 1 = MF-INTOSC is stable
- 0 = MF-INTOSC is not stable

bit 0 **MFIOSEL:** MF-INTOSC Select bit
- 1 = MF-INTOSC is used in place of HF-INTOSC frequencies of 500 kHz, 250 kHz and 31.25 kHz
- 0 = MF-INTOSC is not used

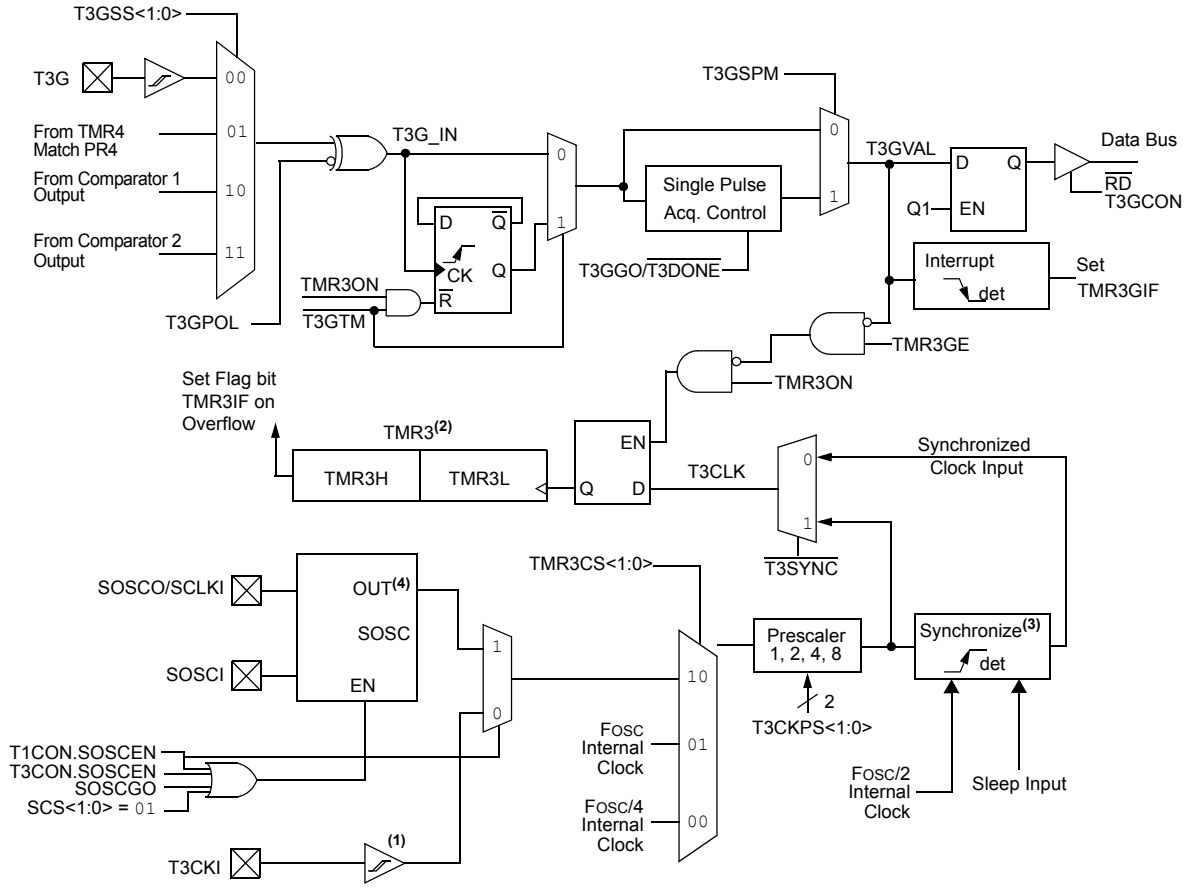
## 16.2 Timer3/5/7 Operation

Timer3, Timer5 and Timer7 can operate in these modes:

- Timer
- Synchronous Counter
- Asynchronous Counter
- Timer with Gated Control

The operating mode is determined by the clock select bits, TMRxCSx (TxCON<7:6>). When the TMRxCSx bits are cleared (= 00), Timer3/5/7 increments on every internal instruction cycle (FOSC/4). When TMRxCSx = 01, the Timer3/5/7 clock source is the system clock (FOSC), and when it is '10', Timer3/5/7 works as a counter from the external clock from the TxCKI pin (on the rising edge after the first falling edge) or the SOSC oscillator.

**FIGURE 16-1: TIMER3/5/7 BLOCK DIAGRAM**




**Note 1:** ST Buffer is high-speed type when using T3CKI.
**2:** Timer3 registers increment on rising edge.
**3:** Synchronization does not operate while in Sleep.
**4:** The output of SOSC is determined by the SOSCSEL<1:0> Configuration bits.


© 2009-2018 Microchip Technology Inc.

## 16.3 Timer3/5/7 16-Bit Read/Write Mode

Timer3/5/7 can be configured for 16-bit reads and writes (see Figure 16.3). When the RD16 control bit (TxCON<1>) is set, the address for TMRxH is mapped to a buffer register for the high byte of Timer3/5/7. A read from TMRxL will load the contents of the high byte of Timer3/5/7 into the Timerx High Byte Buffer register. This provides users with the ability to accurately read all 16 bits of Timer3/5/7 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 Timer3/5/7 must also take place through the TMRxH Buffer register. The Timer3/5/7 high byte is updated with the contents of TMRxH when a write occurs to TMRxL. This allows users to write all 16 bits to both the high and low bytes of Timer3/5/7 at once.

The high byte of Timer3/5/7 is not directly readable or writable in this mode. All reads and writes must take place through the Timerx High Byte Buffer register.

Writes to TMRxH do not clear the Timer3/5/7 prescaler. The prescaler is only cleared on writes to TMRxL.

## 16.4 Using the SOSC Oscillator as the Timer3/5/7 Clock Source

The SOSC internal oscillator may be used as the clock source for Timer3/5/7. The SOSC oscillator is enabled by any peripheral that requests it. There are eight ways the SOSC can be enabled: if the SOSC is selected as the source by any of the odd timers, which is done by each respective SOSCEN bit (TxCON<3>), if the SOSC is selected as the RTCC source by the RTCOSC Configuration bit (CONFIG3L<1>), if the SOSC is selected as the CPU clock source by the SCS bits (OSCCON<1:0>) or if the SOSCGO bit is set (OSCCON2<3>). The SOSCGO bit is used to warm up the SOSC so that it is ready before any peripheral requests it. To use it as the Timer3/5/7 clock source, the TMRxCS bit must also be set. As previously noted, this also configures Timer3/5/7 to increment on every rising edge of the oscillator source.

The SOSC oscillator is described in **Section 14.5 "SOSC Oscillator"**.

## 16.5 Timer3/5/7 Gates

Timer3/5/7 can be configured to count freely or the count can be enabled and disabled using the Timer3/5/7 gate circuitry. This is also referred to as the Timer3/5/7 gate count enable.

The Timer3/5/7 gate can also be driven by multiple selectable sources.

### 16.5.1 TIMER3/5/7 GATE COUNT ENABLE

The Timerx Gate Enable mode is enabled by setting the TMRxGE bit (TxGCON<7>). The polarity of the Timerx Gate Enable mode is configured using the TxGPOL bit (TxGCON<6>).

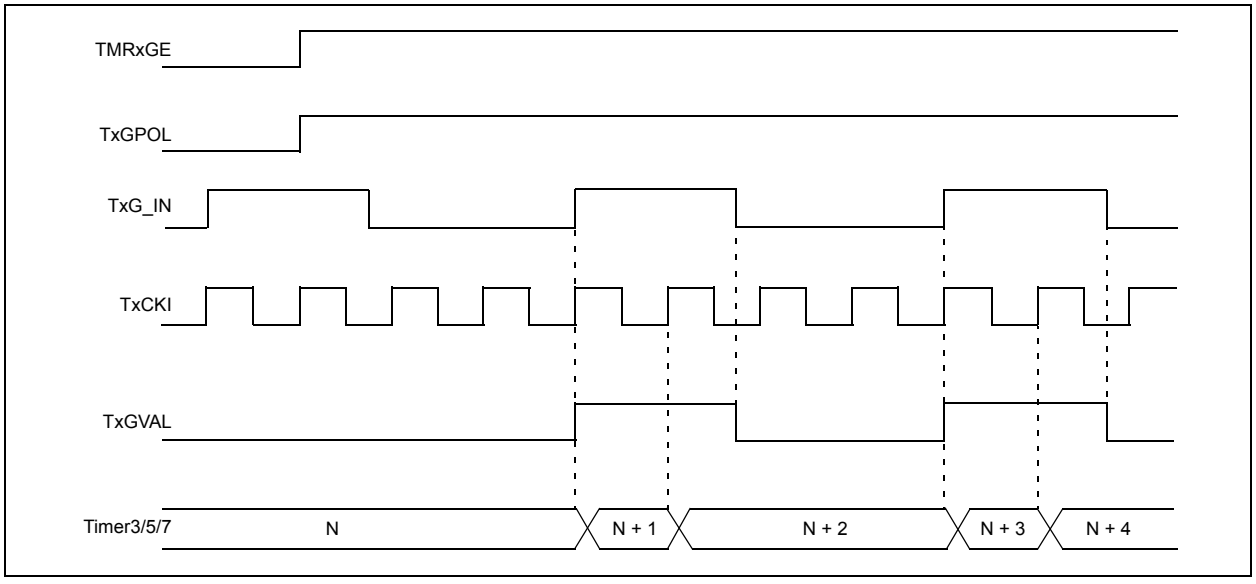
When Timerx Gate Enable mode is enabled, Timer3/5/7 will increment on the rising edge of the Timer3/5/7 clock source. When Timerx Gate Enable mode is disabled, no incrementing will occur and Timer3/5/7 will hold the current count. See Figure 16-2 for timing details.

**TABLE 16-1: TIMER3/5/7 GATE ENABLE SELECTIONS**

| TxCLK(†) | TxGPOL (TxGCON<6>) | TxG Pin | Timerx Operation |
|---|---|---|---|
| ↑ | 0 | 0 | Counts |
| ↑ | 0 | 1 | Holds Count |
| ↑ | 1 | 0 | Holds Count |
| ↑ | 1 | 1 | Counts |

† The clock on which TMR3/5/7 is running. For more information, see TxCLK in Figure 16-1.

**FIGURE 16-2: TIMER3/5/7 GATE COUNT ENABLE MODE**





© 2009-2018 Microchip Technology Inc.

### 16.5.2 TIMER3/5/7 GATE SOURCE SELECTION

The Timer3/5/7 gate source can be selected from one of four different sources. Source selection is controlled by the TxGSS<1:0> bits (TxGCON<1:0>). The polarity for each available source is also selectable and is controlled by the TxGPOL bit (TxGCON <6>).

**TABLE 16-2: TIMER3/5/7 GATE SOURCES**

| TxGSS<1:0> | Timerx Gate Source |
|---|---|
| 00 | Timerx Gate Pin |
| 01 | TMR(x+1) to Match PR(x+1) (TMR(x+1) increments to match PR(x+1)) |
| 10 | Comparator 1 Output (comparator logic high output) |
| 11 | Comparator 2 Output (comparator logic high output) |

#### 16.5.2.1 TxG Pin Gate Operation

The TxG pin is one source for Timer3/5/7 gate control. It can be used to supply an external source to the Timerx gate circuitry.

#### 16.5.2.2 Timer4/6/8 Match Gate Operation

The TMR(x+1) register will increment until it matches the value in the PR(x+1) register. On the very next increment cycle, TMR2 will be reset to 00h. When this Reset occurs, a low-to-high pulse will automatically be generated and internally supplied to the Timerx gate circuitry. The pulse will remain high for one instruction cycle and will return back to a low state until the next match.

Depending on TxGPOL, Timerx increments differently when TMR(x+1) matches PR(x+1). When TxGPOL = 1, Timerx increments for a single instruction cycle following a TMR(x+1) match with PR(x+1). When TxGPOL = 0, Timerx increments continuously, except for the cycle following the match, when the gate signal goes from low-to-high.

#### 16.5.2.3 Comparator 1 Output Gate Operation

The output of Comparator 1 can be internally supplied to the Timerx gate circuitry. After setting up Comparator 1 with the CM1CON register, Timerx will increment depending on the transitions of the CMP1OUT (CMSTAT<5>) bit.

#### 16.5.2.4 Comparator 2 Output Gate Operation

The output of Comparator 2 can be internally supplied to the Timerx gate circuitry. After setting up Comparator 2 with the CM2CON register, Timerx will increment depending on the transitions of the CMP2OUT (CMSTAT<6>) bit.

### 16.5.3 TIMER3/5/7 GATE TOGGLE MODE

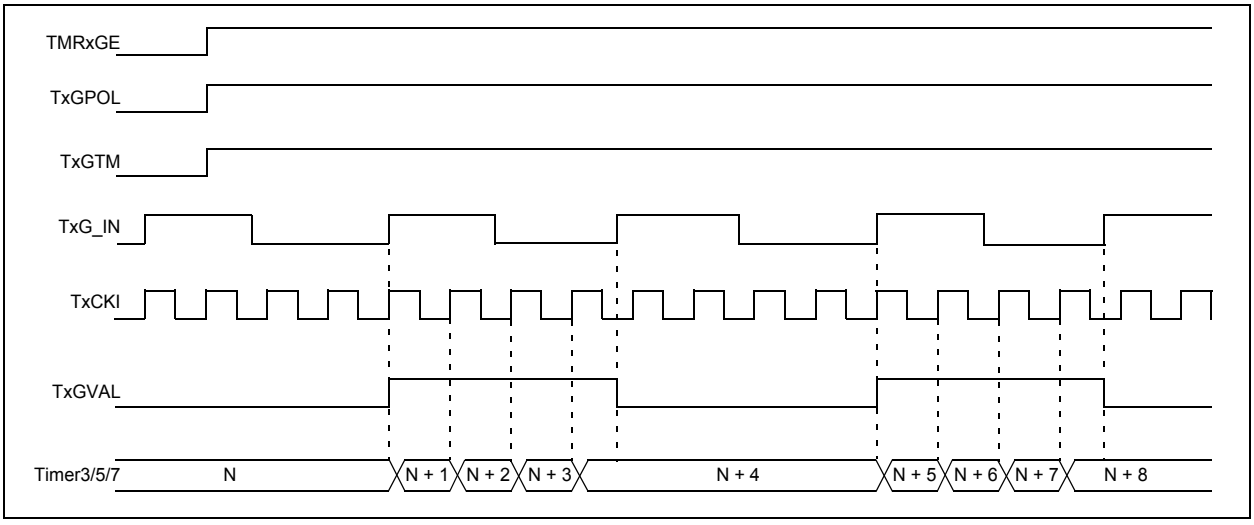
When Timer3/5/7 Gate Toggle mode is enabled, it is possible to measure the full cycle length of a Timer3/5/7 gate signal, as opposed to the duration of a single level pulse.

The Timerx gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. (For timing details, see Figure 16-3.)

The TxGVAL bit will indicate when the Toggled mode is active and the timer is counting.

Timer3/5/7 Gate Toggle mode is enabled by setting the TxGTM bit (TxGCON<5>). When the TxGTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

**FIGURE 16-3: TIMER3/5/7 GATE TOGGLE MODE**



### 16.5.4 TIMER3/5/7 GATE SINGLE PULSE MODE

When Timer3/5/7 Gate Single Pulse mode is enabled, it is possible to capture a single pulse gate event. Timer3/5/7 Gate Single Pulse mode is first enabled by setting the TxGSPM bit (TxGCON<4>). Next, the TxGGO/$\overline{\text{TxDONE}}$ bit (TxGCON<3>) must be set.

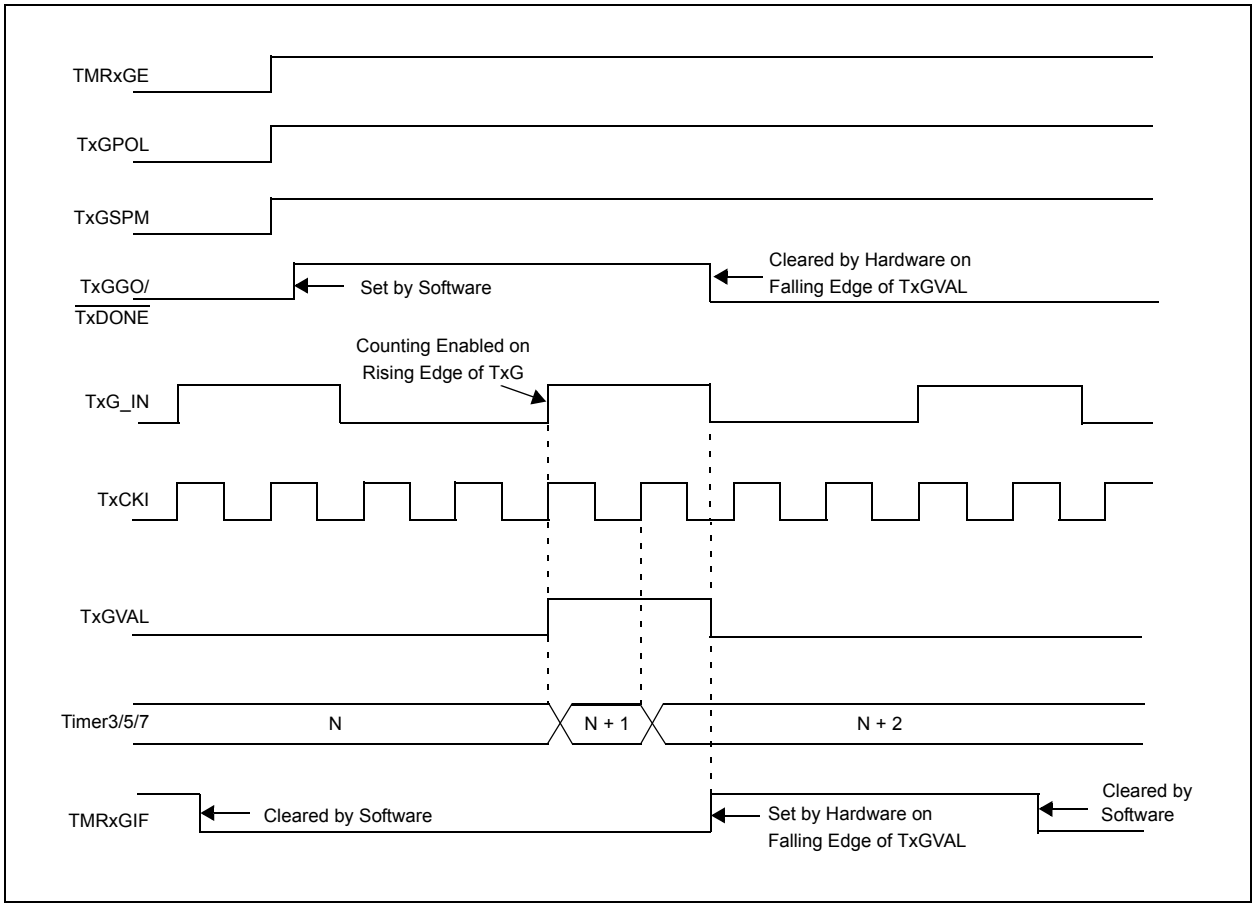
The Timer3/5/7 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the TxGGO/$\overline{\text{TxDONE}}$ bit will automatically be cleared.

No other gate events will be allowed to increment Timer3/5/7 until the TxGGO/$\overline{\text{TxDONE}}$ bit is once again set in software.

Clearing the TxGSPM bit also will clear the TxGGO/$\overline{\text{TxDONE}}$ bit. (For timing details, see Figure 16-4.)

Simultaneously enabling the Toggle mode and the Single Pulse mode will permit both sections to work together. This allows the cycle times on the Timer3/5/7 gate source to be measured. (For timing details, see Figure 16-5.)

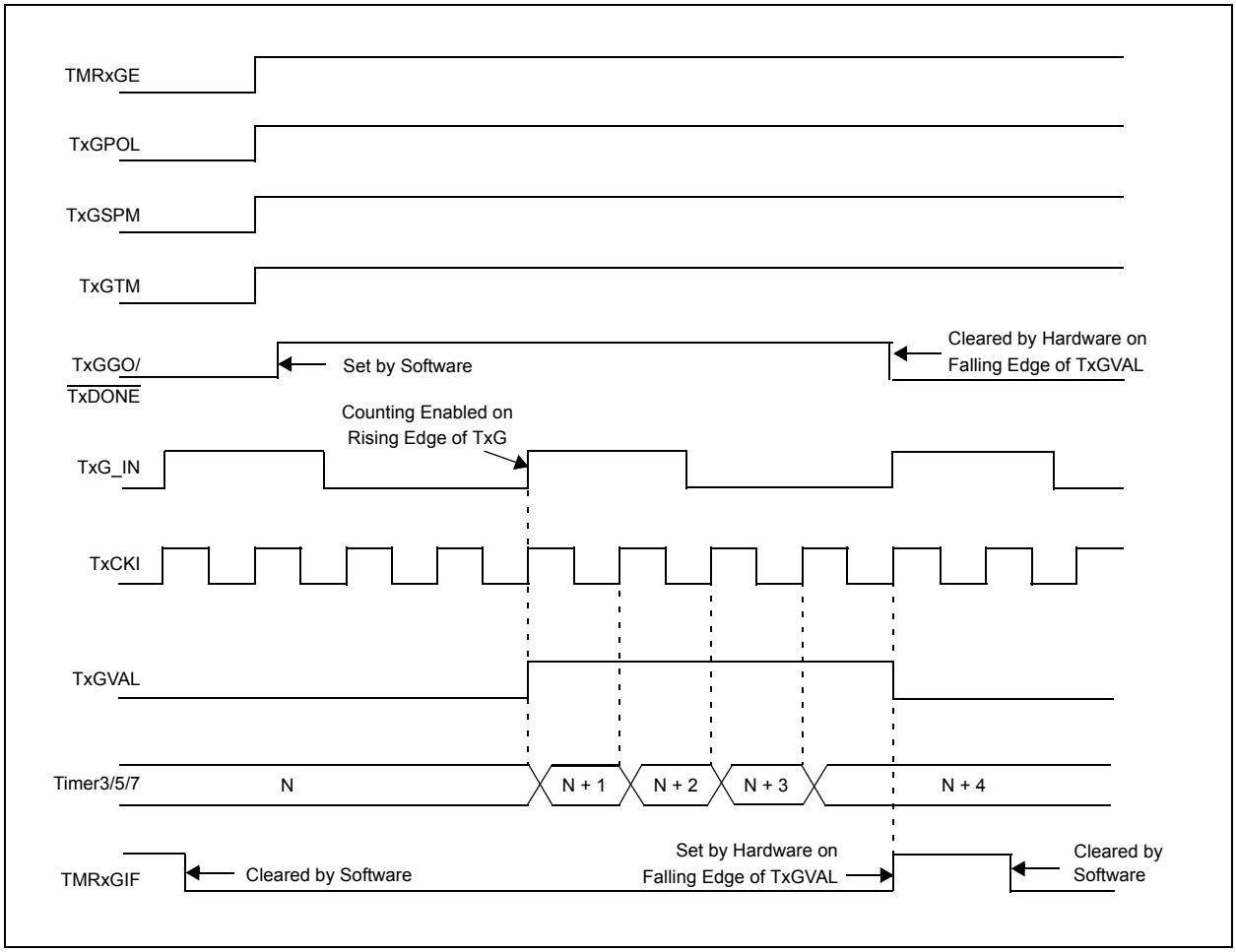
**FIGURE 16-4: TIMER3/5/7 GATE SINGLE PULSE MODE**





© 2009-2018 Microchip Technology Inc.

**FIGURE 16-5: TIMER3/5/7 GATE SINGLE PULSE AND TOGGLE COMBINED MODE**




### 16.5.5 TIMER3/5/7 GATE VALUE STATUS

When Timer3/5/7 gate value status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the TxGVAL bit (TxGCON<2>). The TxGVAL bit is valid even when the Timer3/5/7 gate is not enabled (TMRxGE bit is cleared).

### 16.5.6 TIMER3/5/7 GATE EVENT INTERRUPT

When the Timer3/5/7 gate event interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of TxGVAL occurs, the TMRxGIF flag bit in the PIRx register will be set. If the TMRxGIE bit in the PIEx register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer3/5/7 gate is not enabled (TMRxGE bit is cleared).

## 16.6 Timer3/5/7 Interrupt

The TMRx register pair (TMRxH:TMRxL) increments from 0000h to FFFFh and overflows to 0000h. The Timerx interrupt, if enabled, is generated on overflow and is latched in the interrupt flag bit, TMRxIF. Table 16-3 gives each module's flag bit.

**TABLE 16-3: TIMER3/5/7 INTERRUPT FLAG BITS**

| Timer Module | Flag Bit |
|---|---|
| 3 | PIR2<1> |
| 5 | PIR5<1> |
| 7 | PIR5<3> |

This interrupt can be enabled or disabled by setting or clearing the TMRxIE bit, respectively. Table 16-4 gives each module's enable bit.

**TABLE 16-4: TIMER3/5/7 INTERRUPT ENABLE BITS**

| Timer Module | Flag Bit |
|---|---|
| 3 | PIE2<1> |
| 5 | PIE5<1> |
| 7 | PIE5<3> |

## 16.7 Resetting Timer3/5/7 Using the ECCP Special Event Trigger

If the ECCP modules are configured to use Timerx and to generate a Special Event Trigger in Compare mode (CCPxM<3:0> = 1011), this signal will reset Timerx. The trigger from ECCP2 will also start an A/D conversion if the A/D module is enabled. (For more information, see **Section 20.3.4 "Special Event Trigger"**.)

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 Timerx.

If Timerx is running in Asynchronous Counter mode, the Reset operation may not work.

In the event that a write to Timerx coincides with a Special Event Trigger from an ECCP module, the write will take precedence.

| | |
|---|---|
| **Note:** | The Special Event Triggers from the ECCPx module will only clear the TMR3 register's content, but not set the TMR3IF interrupt flag bit (PIR1<0>). |

| | |
|---|---|
| **Note:** | The CCP and ECCP modules use Timers, 1 through 8, for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRSx registers. For more details, see Register 19-2, Register 19-3 and Register 20-2 |


© 2009-2018 Microchip Technology Inc.

**TABLE 16-5: REGISTERS ASSOCIATED WITH TIMER3/5/7 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR5 | TMR7GIF(1) | TMR12IF(1) | TMR10IF(1) | TMR8IF | TMR7IF(1) | TMR6IF | TMR5IF | TMR4IF |
| IPR5 | TMR7GIP(1) | TMR12IP(1) | TMR10IP(1) | TMR8IP | TMR7IP(1) | TMR6IP | TMR5IP | TMR4IP |
| PIE5 | TMR7GIE(1) | TMR12IE(1) | TMR10IE(1) | TMR8IE | TMR7IE(1) | TMR6IE | TMR5IE | TMR4IE |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE2 | OSCFIE | — | SSP2IE | BCL2IE | BCL1IE | HLVDIE | TMR3IE | TMR3GIE |
| PIR2 | OSCFIF | — | SSP2IF | BCL2IF | BCL1IF | HLVDIF | TMR3IF | TMR3GIF |
| IPR2 | OSCFIP | — | SSP2IP | BCL2IP | BCL1IP | HLVDIP | TMR3IP | TMR3GIP |
| TMR3H | Timer3 Register High Byte | | | | | | | |
| TMR3L | Timer3 Register Low Byte | | | | | | | |
| T3GCON | TMR3GE | T3GPOL | T3GTM | T3GSPM | T3GGO/$\overline{\text{T3DONE}}$ | T3GVAL | T3GSS1 | T3GSS0 |
| T3CON | TMR3CS1 | TMR3CS0 | T3CKPS1 | T3CKPS0 | SOSCEN | $\overline{\text{T3SYNC}}$ | RD16 | TMR3ON |
| TMR5H | Timer5 Register High Byte | | | | | | | |
| TMR5L | Timer5 Register Low Byte | | | | | | | |
| T5GCON | TMR5GE | T5GPOL | T5GTM | T5GSPM | T5GGO/$\overline{\text{T5DONE}}$ | T5GVAL | T5GSS1 | T5GSS0 |
| T5CON | TMR5CS1 | TMR5CS0 | T5CKPS1 | T5CKPS0 | SOSCEN | $\overline{\text{T5SYNC}}$ | RD16 | TMR5ON |
| TMR7H(1) | Timer7 Register High Byte | | | | | | | |
| TMR7L(1) | Timer7 Register Low Byte | | | | | | | |
| T7GCON(1) | TMR7GE | T7GPOL | T7GTM | T7GSPM | T7GGO/$\overline{\text{T7DONE}}$ | T7GVAL | T7GSS1 | T7GSS0 |
| T7CON(1) | TMR7CS1 | TMR7CS0 | T7CKPS1 | T7CKPS0 | SOSCEN | $\overline{\text{T7SYNC}}$ | RD16 | TMR7ON |
| OSCCON2 | — | SOSCRUN | — | — | SOSCGO | — | MFIOFS | MFIOSEL |
| PMD1 | PSPMD | CTMUMD | RTCCMD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | EMBMD |
| PMD2 | TMR10MD(1) | TMR8MD | TMR7MD(1) | TMR6MD | TMR5MD | CMP3MD | CMP2MD | CMP2MD |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer3/5/7 module.

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).

## 17.0 TIMER4/6/8/10/12 MODULES

The Timer4/6/8/10/12 timer modules have the following features:

- Eight-bit Timer register (TMRx)
- Eight-bit Period register (PRx)
- Readable and writable (all registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMRx match of PRx

Timer10 and Timer12 are unimplemented for devices with a program memory of 32 Kbytes (PIC18FX5K22).

**Note:** Throughout this section, generic references are used for register and bit names that are the same, except for an 'x' variable that indicates the item's association with the Timer4, Timer6, Timer8, Timer10 or Timer12 module. For example, the control register is named TxCON and refers to T4CON, T6CON, T8CON, T10CON and T12CON.

The Timer4/6/8/10/12 modules have a control register, which is shown in Register 17-1. Timer4/6/8/10/12 can be shut off by clearing control bit, TMRxON (TxCON<2>), to minimize power consumption. The prescaler and post-scaler selection of Timer4/6/8/10/12 are also controlled by this register. Figure 17-1 is a simplified block diagram of the Timer4/6/8/10/12 modules.

### 17.1 Timer4/6/8/10/12 Operation

Timer4/6/8/10/12 can be used as the PWM time base for the PWM mode of the ECCP modules. The TMRx registers are readable and writable, and are 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, TxCKPS<1:0> (TxCON<1:0>). The match output of TMRx goes through a four-bit postscaler (that gives a 1:1 to 1:16 inclusive scaling) to generate a TMRx interrupt, latched in the flag bit, TMRxIF. Table 17-1 shows each module's flag bit.

**TABLE 17-1: TIMER4/6/8/10/12 FLAG BITS**

| Timer Module | Flag Bit PIR5<x> | Timer Module | Flag Bit PIR5<x> |
|---|---|---|---|
| 4 | 0 | 10 | 5 |
| 6 | 2 | 12 | 6 |
| 8 | 4 | | |

The interrupt can be enabled or disabled by setting or clearing the Timerx Interrupt Enable bit (TMRxIE), shown in Table 17-2.

**TABLE 17-2: TIMER4/6/8/10/12 INTERRUPT ENABLE BITS**

| Timer Module | Flag Bit PIE5<x> | Timer Module | Flag Bit PIE5<x> |
|---|---|---|---|
| 4 | 0 | 10 | 5 |
| 6 | 2 | 12 | 6 |
| 8 | 4 | | |

The prescaler and postscaler counters are cleared when any of the following occurs:

- A write to the TMRx register
- A write to the TxCON register
- Any device Reset – Power-on Reset (POR), MCLR Reset, Watchdog Timer Reset (WDTR) or Brown-out Reset (BOR)

A TMRx is not cleared when a TxCON is written.

**Note:** The CCP and ECCP modules use Timers, 1 through 8, for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRSx registers. For more details, see Register 19-2, Register 19-3 and Register 20-2.


© 2009-2018 Microchip Technology Inc.

**REGISTER 17-1: TxCON: TIMERx 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 |
|---|---|---|---|---|---|---|---|
| — | TxOUTPS3 | TxOUTPS2 | TxOUTPS1 | TxOUTPS0 | TMRxON | TxCKPS1 | TxCKPS0 |
| 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 **TxOUTPS<3:0>:** Timerx Output Postscale Select bits
0000 = 1:1 Postscale
0001 = 1:2 Postscale
•
•
•
1111 = 1:16 Postscale

bit 2 **TMRxON**: Timerx On bit
1 = Timerx is on
0 = Timerx is off

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

## 17.2 Timer4/6/8/10/12 Interrupt

The Timer4/6/8/10/12 modules have eight-bit Period registers, PRx, that are both readable and writable. Timer4/6/8/10/12 increment from 00h until they match PR4/6/8/10/12 and then reset to 00h on the next increment cycle. The PRx registers are initialized to FFh upon Reset.

## 17.3 Output of TMRx

The outputs of TMRx (before the postscaler) are used only as a PWM time base for the ECCP modules. They are not used as baud rate clocks for the MSSP modules as is the Timer2 output.

**FIGURE 17-1: TIMER4 BLOCK DIAGRAM**

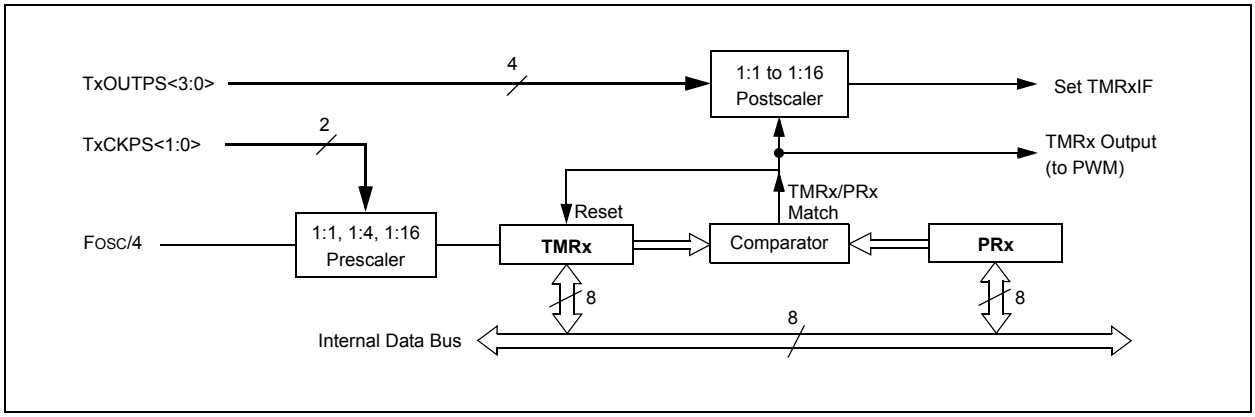


**TABLE 17-3: REGISTERS ASSOCIATED WITH TIMER4/6/8/10/12 AS A TIMER/COUNTER**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| IPR5 | TMR7GIP(1) | TMR12IP(1) | TMR10IP(1) | TMR8IP | TMR7IP(1) | TMR6IP | TMR5IP | TMR4IP |
| PIR5 | TMR7GIF(1) | TMR12IF(1) | TMR10IF(1) | TMR8IF | TMR7IF(1) | TMR6IF | TMR5IF | TMR4IF |
| PIE5 | TMR7GIE(1) | TMR12IE(1) | TMR10IE(1) | TMR8IE | TMR7IE(1) | TMR6IE | TMR5IE | TMR4IE |
| TMR4 | Timer4 Register | | | | | | | |
| T4CON | — | T4OUTPS3 | T4OUTPS2 | T4OUTPS1 | T4OUTPS0 | TMR4ON | T4CKPS1 | T4CKPS0 |
| PR4 | Timer4 Period Register | | | | | | | |
| TMR6 | Timer6 Register | | | | | | | |
| T6CON | — | T6OUTPS3 | T6OUTPS2 | T6OUTPS1 | T6OUTPS0 | TMR6ON | T6CKPS1 | T6CKPS0 |
| PR6 | Timer6 Period Register | | | | | | | |
| TMR8 | Timer8 Register | | | | | | | |
| T8CON | — | T8OUTPS3 | T8OUTPS2 | T8OUTPS1 | T8OUTPS0 | TMR8ON | T8CKPS1 | T8CKPS0 |
| PR8 | Timer8 Period Register | | | | | | | |
| TMR10(1) | Timer10 Register | | | | | | | |
| T10CON(1) | — | T10OUTPS3 | T10OUTPS2 | T10OUTPS1 | T10OUTPS0 | TMR10ON | T10CKPS1 | T10CKPS0 |
| PR10(1) | Timer10 Period Register | | | | | | | |
| TMR12(1) | Timer12 Register | | | | | | | |
| T12CON(1) | — | T12OUTPS3 | T12OUTPS2 | T12OUTPS1 | T12OUTPS0 | TMR12ON | T12CKPS1 | T12CKPS0 |
| PR12(1) | Timer12 Period Register | | | | | | | |
| PMD1 | PSPMD | CTMUMD | RTCCMD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | EMBMD |
| PMD2 | TMR10MD(1) | TMR8MD | TMR7MD(1) | TMR6MD | TMR5MD | CMP3MD | CMP2MD | CMP2MD |
| PMD3 | CCP10MD(1) | CCP9MD(1) | CCP8MD | CCP7MD | CCP6MD | CCP5MD | CCP4MD | TMR12MD(1) |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by the Timer4/6/8/10/12 module.

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18FX5K22).


© 2009-2018 Microchip Technology Inc.

## 18.0 REAL-TIME CLOCK AND CALENDAR (RTCC)

The key features of the Real-Time Clock and Calendar (RTCC) module are:

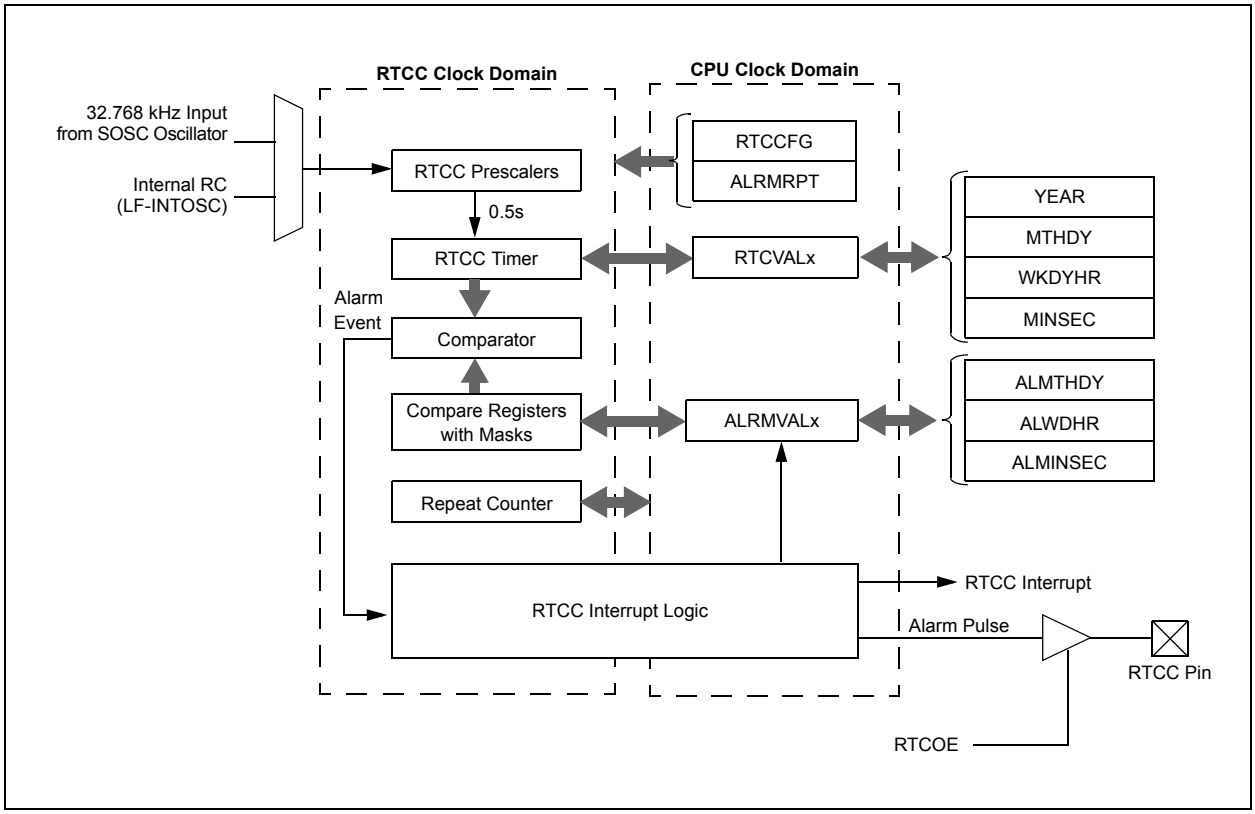
- Time: hours, minutes and seconds
- Twenty-four hour format (military time)
- Calendar: weekday, date, month and year
- Alarm configurable
- Year range: 2000 to 2099
- Leap year correction
- BCD format for compact firmware
- Optimized for low-power operation
- User calibration with auto-adjust
- Calibration range: ±2.64 seconds error per month
- Requirements: external 32.768 kHz clock crystal
- Alarm pulse or seconds clock output on RTCC pin

The RTCC module is intended for applications where accurate time must be maintained for extended period with minimum to no intervention from the CPU. The module is optimized for low-power usage in order to provide extended battery life while keeping track of time.

The module is a 100-year clock and calendar with automatic leap year detection. The range of the clock is from 00:00:00 (midnight) on January 1, 2000 to 23:59:59 on December 31, 2099.

Hours are measured in 24-hour (military time) format. The clock provides a granularity of one second with half-second visibility to the user.

**FIGURE 18-1: RTCC BLOCK DIAGRAM**



## 18.1 RTCC MODULE REGISTERS

The RTCC module registers are divided into the following categories:

### RTCC Control Registers

- RTCCFG
- RTCCAL
- PADCFG1
- ALRMCFG
- ALRMRPT

### RTCC Value Registers

- RTCVALH
- RTCVALL

  Both registers access the following registers:
  - YEAR
  - MONTH
  - DAY
  - WEEKDAY
  - HOUR
  - MINUTE
  - SECOND

### Alarm Value Registers

- ALRMVALH
- ALRMVALL

  Both registers access the following registers:
  - ALRMMNTH
  - ALRMDAY
  - ALRMWD
  - ALRMHR
  - ALRMMIN
  - ALRMSEC

| **Note:** | The RTCVALH and RTCVALL registers can be accessed through RTCRPT<1:0> (RTCCFG<1:0>). ALRMVALH and ALRMVALL can be accessed through ALRMPTR<1:0> (ALRMCFG<1:0>). |
|---|---|


© 2009-2018 Microchip Technology Inc.

### 18.1.1 RTCC CONTROL REGISTERS

**REGISTER 18-1: RTCCFG: RTCC CONFIGURATION REGISTER(1)**

| R/W-0 | U-0 | R/W-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| RTCEN(2) | — | RTCWREN(4) | RTCSYNC | HALFSEC(3) | RTCOE | RTCPTR1 | RTCPTR0 |
| 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 **RTCEN:** RTCC Enable bit(2)
- 1 = RTCC module is enabled
- 0 = RTCC module is disabled

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

bit 5 **RTCWREN:** RTCC Value Registers Write Enable bit(4)
- 1 = RTCVALH and RTCVALL registers can be written to by the user
- 0 = RTCVALH and RTCVALL registers are locked out from being written to by the user

bit 4 **RTCSYNC:** RTCC Value Registers Read Synchronization bit
- 1 = RTCVALH, RTCVALL and ALRMRPT registers can change while reading if a rollover ripple results in an invalid data read. If the register is read twice and results in the same data, the data can be assumed to be valid.
- 0 = RTCVALH, RTCVALL and ALCFGRPT registers can be read without concern over a rollover ripple

bit 3 **HALFSEC:** Half-Second Status bit(3)
- 1 = Second half period of a second
- 0 = First half period of a second

bit 2 **RTCOE:** RTCC Output Enable bit
- 1 = RTCC clock output is enabled
- 0 = RTCC clock output is disabled

bit 1-0 **RTCPTR<1:0>:** RTCC Value Register Window Pointer bits

Points to the corresponding RTCC Value registers when reading the RTCVALH and RTCVALL registers. The RTCPTR<1:0> value decrements on every read or write of RTCVALH<15:8> until it reaches '00'.

RTCVALH:
- 00 = Minutes
- 01 = Weekday
- 10 = Month
- 11 = Reserved

RTCVALL:
- 00 = Seconds
- 01 = Hours
- 10 = Day
- 11 = Year

**Note 1:** The RTCCFG register is only affected by a POR.
**2:** A write to the RTCEN bit is only allowed when RTCWREN = 1.
**3:** This bit is read-only; it is cleared to '0' on a write to the lower half of the MINSEC register.
**4:** RTCWREN can only be written with the unlock sequence (see Example 18-1).

**REGISTER 18-2: RTCCAL: RTCC CALIBRATION 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 |
|---|---|---|---|---|---|---|---|
| CAL7 | CAL6 | CAL5 | CAL4 | CAL3 | CAL2 | CAL1 | CAL0 |
| 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 **CAL<7:0>:** RTC Drift Calibration bits

01111111 = Maximum positive adjustment. Adds 508 RTC clock pulses every minute.
.
.
.
00000001 = Minimum positive adjustment. Adds four RTC clock pulses every minute.
00000000 = No adjustment
11111111 = Minimum negative adjustment. Subtracts four RTC clock pulses every minute.
.
.
.
10000000 = Maximum negative adjustment. Subtracts 512 RTC clock pulses every minute.

**REGISTER 18-3: PADCFG1: PAD CONFIGURATION REGISTER**

| R/W-0 | R/W-0 | R/W-0 | U-0 | U-0 | R/W-0 | R/W-0 | U-0 |
|---|---|---|---|---|---|---|---|
| RDPU | REPU | RJPU(2) | — | — | RTSECSEL1(1) | RTSECSEL0(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 **RDPU:** PORTD Pull-up Enable bit
1 = PORTD pull-up resistors are enabled by individual port latch values
0 = All PORTD pull-up resistors are disabled

bit 6 **REPU:** PORTE Pull-up Enable bit
1 = PORTE pull-up resistors are enabled by individual port latch values
0 = All PORTE pull-up resistors are disabled

bit 5 **RJPU:** PORTJ Pull-up Enable bit(2)
1 = PORTJ pull-up resistors are enabled by individual port latch values
0 = All PORTJ pull-up resistors are disabled

bit 2-1 **RTSECSEL<1:0>:** RTCC Seconds Clock Output Select bits(1)
11 = Reserved; do not use
10 = RTCC source clock is selected for the RTCC pin (pin can be LF-INTOSC or SOSC, depending on the RTCOSC (CONFIG3L<1>) bit setting)
01 = RTCC seconds clock is selected for the RTCC pin
00 = RTCC alarm pulse is selected for the RTCC pin

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

**Note 1:** To enable the actual RTCC output, the RTCOE (RTCCFG<2>) bit must be set.
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.


© 2009-2018 Microchip Technology Inc.

**REGISTER 18-4: ALRMCFG: ALARM CONFIGURATION 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 |
|---|---|---|---|---|---|---|---|
| ALRMEN | CHIME | AMASK3 | AMASK2 | AMASK1 | AMASK0 | ALRMPTR1 | ALRMPTR0 |
| 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 **ALRMEN:** Alarm Enable bit

1 = Alarm is enabled (cleared automatically after an alarm event whenever ARPT<7:0> = 00 and CHIME = 0)
0 = Alarm is disabled

bit 6 **CHIME:** Chime Enable bit

1 = Chime is enabled; ALRMPTR<1:0> bits are allowed to roll over from 00h to FFh
0 = Chime is disabled; ALRMPTR<1:0> bits stop once they reach 00h

bit 5-2 **AMASK<3:0>:** Alarm Mask Configuration bits

0000 = Every half second
0001 = Every second
0010 = Every 10 seconds
0011 = Every minute
0100 = Every 10 minutes
0101 = Every hour
0110 = Once a day
0111 = Once a week
1000 = Once a month
1001 = Once a year (except when configured for February 29th, once every four years)
101x = Reserved – Do not use
11xx = Reserved – Do not use

bit 1-0 **ALRMPTR<1:0>:** Alarm Value Register Window Pointer bits

Points to the corresponding Alarm Value registers when reading the ALRMVALH and ALRMVALL registers. The ALRMPTR<1:0> value decrements on every read or write of ALRMVALH until it reaches '00'.

ALRMVALH:
00 = ALRMMIN
01 = ALRMWD
10 = ALRMMNTH
11 = Unimplemented

ALRMVALL:
00 = ALRMSEC
01 = ALRMHR
10 = ALRMDAY
11 = Unimplemented

**REGISTER 18-5: ALRMRPT: ALARM REPEAT 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 |
|---|---|---|---|---|---|---|---|
| ARPT7 | ARPT6 | ARPT5 | ARPT4 | ARPT3 | ARPT2 | ARPT1 | ARPT0 |
| 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 **ARPT<7:0>:** Alarm Repeat Counter Value bits
11111111 = Alarm will repeat 255 more times
.
.
.
00000000 = Alarm will not repeat
The counter decrements on any alarm event. The counter is prevented from rolling over from 00h to FFh unless CHIME = 1.

### 18.1.2 RTCVALH AND RTCVALL REGISTER MAPPINGS

**REGISTER 18-6: RESERVED REGISTER**

| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| 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 **Unimplemented:** Read as '0'

**REGISTER 18-7: YEAR: YEAR VALUE REGISTER(1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| YRTEN3 | YRTEN2 | YRTEN1 | YRTEN0 | YRONE3 | YRONE2 | YRONE1 | YRONE0 |
| 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 7-4 **YRTEN<3:0>:** Binary Coded Decimal Value of Year's Tens Digit bits
Contains a value from 0 to 9.

bit 3-0 **YRONE<3:0>:** Binary Coded Decimal Value of Year's Ones Digit bits
Contains a value from 0 to 9.

**Note 1:** A write to the YEAR register is only allowed when RTCWREN = 1.


© 2009-2018 Microchip Technology Inc.

**REGISTER 18-8: MONTH: MONTH VALUE REGISTER(1)**

| U-0 | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | — | MTHTEN0 | MTHONE3 | MTHONE2 | MTHONE1 | MTHONE0 |
| 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 **MTHTEN0:** Binary Coded Decimal Value of Month's Tens Digit bit
Contains a value of 0 or 1.

bit 3-0 **MTHONE<3:0>:** Binary Coded Decimal Value of Month's Ones Digit bits
Contains a value from 0 to 9.

**Note 1:** A write to this register is only allowed when RTCWREN = 1.

**REGISTER 18-9: DAY: DAY VALUE REGISTER(1)**

| U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | DAYTEN1 | DAYTEN0 | DAYONE3 | DAYONE2 | DAYONE1 | DAYONE0 |
| 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 **DAYTEN<1:0>:** Binary Coded Decimal value of Day's Tens Digit bits
Contains a value from 0 to 3.

bit 3-0 **DAYONE<3:0>:** Binary Coded Decimal Value of Day's Ones Digit bits
Contains a value from 0 to 9.

**Note 1:** A write to this register is only allowed when RTCWREN = 1.

**REGISTER 18-10: WEEKDAY: WEEKDAY VALUE REGISTER(1)**

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | WDAY2 | WDAY1 | WDAY0 |
| 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 7-3 **Unimplemented:** Read as '0'

bit 2-0 **WDAY<2:0>:** Binary Coded Decimal Value of Weekday Digit bits
Contains a value from 0 to 6.

**Note 1:** A write to this register is only allowed when RTCWREN = 1.

**REGISTER 18-11: HOUR: HOUR VALUE REGISTER(1)**

| U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | HRTEN1 | HRTEN0 | HRONE3 | HRONE2 | HRONE1 | HRONE0 |
| 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 **HRTEN<1:0>:** Binary Coded Decimal Value of Hour's Tens Digit bits
Contains a value from 0 to 2.

bit 3-0 **HRONE<3:0>:** Binary Coded Decimal Value of Hour's Ones Digit bits
Contains a value from 0 to 9.

**Note 1:** A write to this register is only allowed when RTCWREN = 1.

**REGISTER 18-12: MINUTE: MINUTE VALUE REGISTER**

| U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | MINTEN2 | MINTEN1 | MINTEN0 | MINONE3 | MINONE2 | MINONE1 | MINONE0 |
| 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 6-4 **MINTEN<2:0>:** Binary Coded Decimal Value of Minute's Tens Digit bits
Contains a value from 0 to 5.

bit 3-0 **MINONE<3:0>:** Binary Coded Decimal Value of Minute's Ones Digit bits
Contains a value from 0 to 9.

**REGISTER 18-13: SECOND: SECOND VALUE REGISTER**

| U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | SECTEN2 | SECTEN1 | SECTEN0 | SECONE3 | SECONE2 | SECONE1 | SECONE0 |
| 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 6-4 **SECTEN<2:0>:** Binary Coded Decimal Value of Second's Tens Digit bits
Contains a value from 0 to 5.

bit 3-0 **SECONE<3:0>:** Binary Coded Decimal Value of Second's Ones Digit bits
Contains a value from 0 to 9.


© 2009-2018 Microchip Technology Inc.

### 18.1.3 ALRMVALH AND ALRMVALL REGISTER MAPPINGS

**REGISTER 18-14: ALRMMNTH: ALARM MONTH VALUE REGISTER(1)**

| U-0 | U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | — | MTHTEN0 | MTHONE3 | MTHONE2 | MTHONE1 | MTHONE0 |
| 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 **MTHTEN0:** Binary Coded Decimal Value of Month's Tens Digit bits
Contains a value of 0 or 1.

bit 3-0 **MTHONE<3:0>:** Binary Coded Decimal Value of Month's Ones Digit bits
Contains a value from 0 to 9.

**Note 1:** A write to this register is only allowed when RTCWREN = 1.

**REGISTER 18-15: ALRMDAY: ALARM DAY VALUE REGISTER(1)**

| U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | DAYTEN1 | DAYTEN0 | DAYONE3 | DAYONE2 | DAYONE1 | DAYONE0 |
| 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 **DAYTEN<1:0>:** Binary Coded Decimal Value of Day's Tens Digit bits
Contains a value from 0 to 3.

bit 3-0 **DAYONE<3:0>:** Binary Coded Decimal Value of Day's Ones Digit bits
Contains a value from 0 to 9.

**Note 1:** A write to this register is only allowed when RTCWREN = 1.

**REGISTER 18-16: ALRMWD: ALARM WEEKDAY VALUE REGISTER(1)**

| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | WDAY2 | WDAY1 | WDAY0 |
| 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 7-3 **Unimplemented:** Read as '0'

bit 2-0 **WDAY<2:0>:** Binary Coded Decimal Value of Weekday Digit bits
Contains a value from 0 to 6.

**Note 1:** A write to this register is only allowed when RTCWREN = 1.

**REGISTER 18-17: ALRMHR: ALARM HOURS VALUE REGISTER(1)**

| U-0 | U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | — | HRTEN1 | HRTEN0 | HRONE3 | HRONE2 | HRONE1 | HRONE0 |
| 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 **HRTEN<1:0>:** Binary Coded Decimal Value of Hour's Tens Digit bits
Contains a value from 0 to 2.

bit 3-0 **HRONE<3:0>:** Binary Coded Decimal Value of Hour's Ones Digit bits
Contains a value from 0 to 9.

**Note 1:** A write to this register is only allowed when RTCWREN = 1.

**REGISTER 18-18: ALRMMIN: ALARM MINUTES VALUE REGISTER**

| U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | MINTEN2 | MINTEN1 | MINTEN0 | MINONE3 | MINONE2 | MINONE1 | MINONE0 |
| 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 6-4 **MINTEN<2:0>:** Binary Coded Decimal Value of Minute's Tens Digit bits
Contains a value from 0 to 5.

bit 3-0 **MINONE<3:0>:** Binary Coded Decimal Value of Minute's Ones Digit bits
Contains a value from 0 to 9.

**REGISTER 18-19: ALRMSEC: ALARM SECONDS VALUE REGISTER**

| U-0 | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| — | SECTEN2 | SECTEN1 | SECTEN0 | SECONE3 | SECONE2 | SECONE1 | SECONE0 |
| 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 6-4 **SECTEN<2:0>:** Binary Coded Decimal Value of Second's Tens Digit bits
Contains a value from 0 to 5.

bit 3-0 **SECONE<3:0>:** Binary Coded Decimal Value of Second's Ones Digit bits
Contains a value from 0 to 9.


© 2009-2018 Microchip Technology Inc.

### 18.1.4 RTCEN BIT WRITE

RTCWREN (RTCCFG<5>) must be set before a write to RTCEN can take place. Any write to the RTCEN bit, while RTCWREN = 0, will be ignored.

Like the RTCEN bit, the RTCVALH and RTCVALL registers can only be written to when RTCWREN = 1. A write to these registers, while RTCWREN = 0, will be ignored.

## 18.2 Operation

### 18.2.1 REGISTER INTERFACE

The register interface for the RTCC and alarm values is implemented using the Binary Coded Decimal (BCD) format. This simplifies the firmware when using the module, as each of the digits is contained within its own 4-bit value (see Figure 18-2 and Figure 18-3).

**FIGURE 18-2: TIMER DIGIT FORMAT**

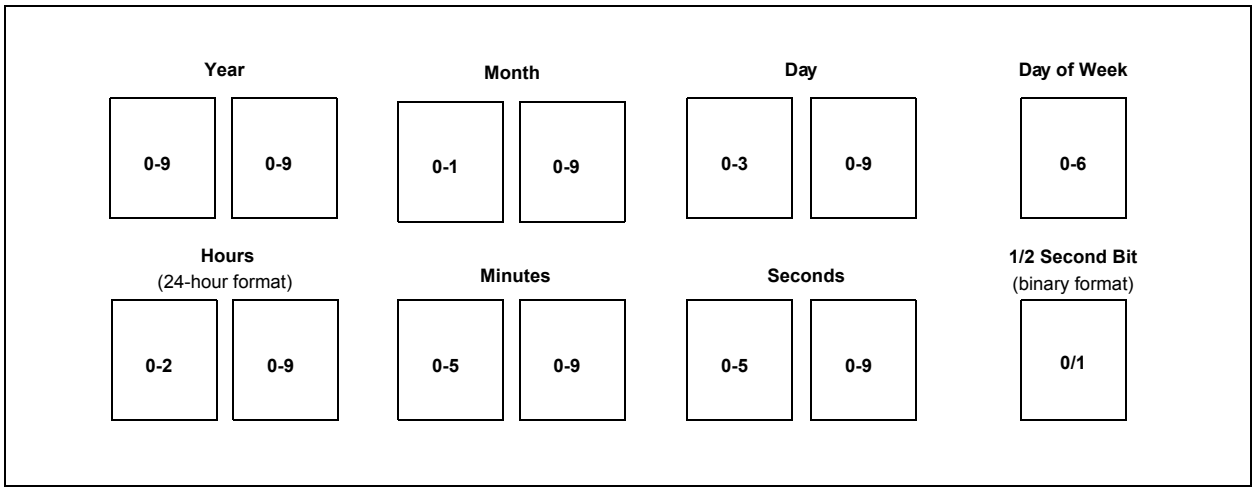
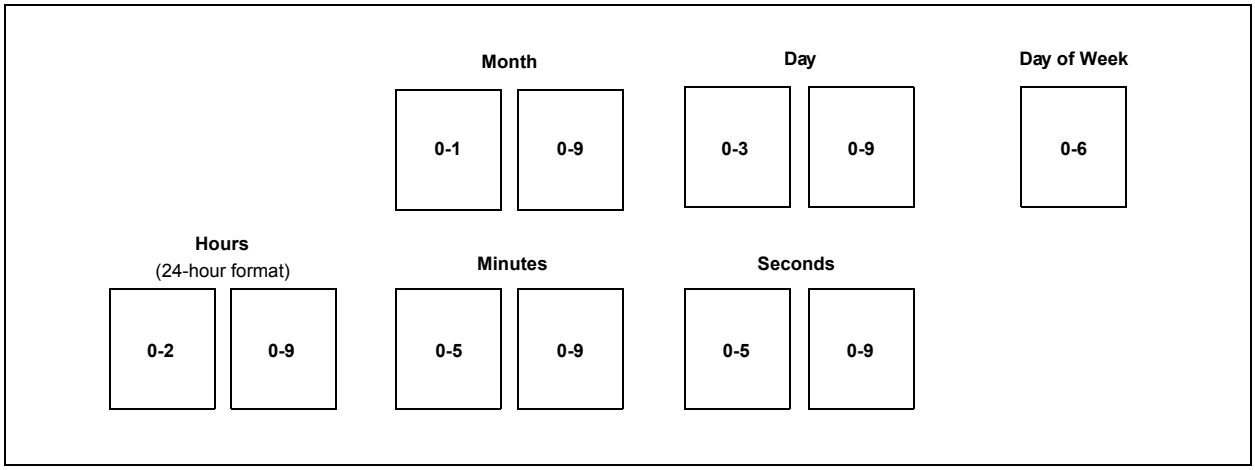



**FIGURE 18-3: ALARM DIGIT FORMAT**

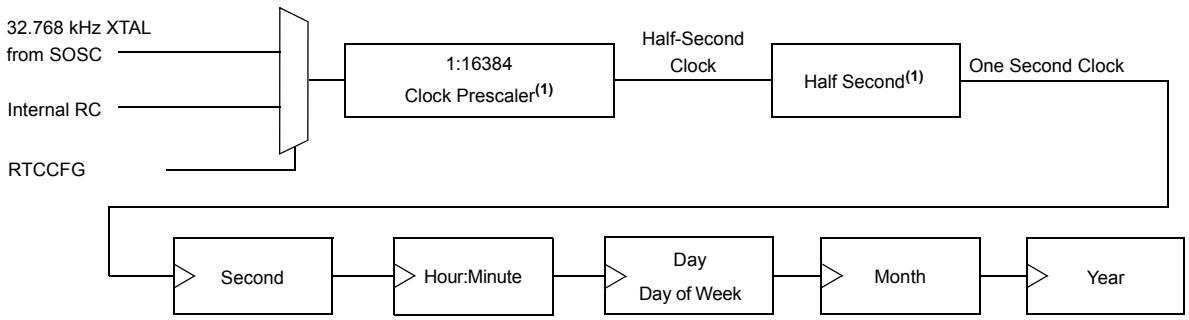


### 18.2.2 CLOCK SOURCE

As previously mentioned, the RTCC module is intended to be clocked by an external Real-Time Clock (RTC) crystal, oscillating at 32.768 kHz, but an internal oscillator can be used. The RTCC clock selection is decided by the RTCOSC bit (CONFIG3L<0>).

Calibration of the crystal can be done through this module to yield an error of 3 seconds or less per month. (For further details, see **Section 18.2.9 "Calibration"**.)

**FIGURE 18-4: CLOCK SOURCE MULTIPLEXING**




**Note 1:** Writing to the lower half of the MINSEC register resets all counters, allowing fraction of a second synchronization; clock prescaler is held in Reset when RTCEN = 0.

#### 18.2.2.1 Real-Time Clock Enable

The RTCC module can be clocked by an external, 32.768 kHz crystal (SOSC oscillator) or the LF-INTOSC oscillator, which can be selected in CONFIG3L<0>.

If the external clock is used, the SOSC oscillator should be enabled. If LF-INTOSC is providing the clock, the INTOSC clock can be brought out to the RTCC pin by the RTSECSEL<1:0> bits (PADCFG<2:1>).

### 18.2.3 DIGIT CARRY RULES

This section explains which timer values are affected when there is a rollover:

- Time of Day: From 23:59:59 to 00:00:00 with a carry to the Day field
- Month: From 12/31 to 01/01 with a carry to the Year field
- Day of Week: From 6 to 0 with no carry (see Table 18-1)
- Year Carry: From 99 to 00; this also surpasses the use of the RTCC

For the day-to-month rollover schedule, see Table 18-2.

Because the following values are in BCD format, the carry to the upper BCD digit occurs at the count of 10, not 16 (SECONDS, MINUTES, HOURS, WEEKDAY, DAYS and MONTHS).

**TABLE 18-1: DAY OF WEEK SCHEDULE**

| Day of Week | |
|---|---|
| Sunday | 0 |
| Monday | 1 |
| Tuesday | 2 |
| Wednesday | 3 |
| Thursday | 4 |
| Friday | 5 |
| Saturday | 6 |

**TABLE 18-2: DAY TO MONTH ROLLOVER SCHEDULE**

| Month | Maximum Day Field |
|---|---|
| 01 (January) | 31 |
| 02 (February) | 28 or 29(1) |
| 03 (March) | 31 |
| 04 (April) | 30 |
| 05 (May) | 31 |
| 06 (June) | 30 |
| 07 (July) | 31 |
| 08 (August) | 31 |
| 09 (September) | 30 |
| 10 (October) | 31 |
| 11 (November) | 30 |
| 12 (December) | 31 |

**Note 1:** See **Section 18.2.4 "Leap Year"**.


© 2009-2018 Microchip Technology Inc.

### 18.2.4 LEAP YEAR

Since the year range on the RTCC module is 2000 to 2099, the leap year calculation is determined by any year divisible by four in the above range. Only February is affected in a leap year.

February will have 29 days in a leap year and 28 days in any other year.

### 18.2.5 GENERAL FUNCTIONALITY

All Timer registers containing a time value of seconds or greater are writable. The user configures the time by writing the required year, month, day, hour, minutes and seconds to the Timer registers, via register pointers. (See **Section 18.2.8 "Register Mapping"**.)

The timer uses the newly written values and proceeds with the count from the required starting point.

The RTCC is enabled by setting the RTCEN bit (RTCCFG<7>). If enabled, while adjusting these registers, the timer still continues to increment. However, any time the MINSEC register is written to, both of the timer prescalers are reset to '0'. This allows fraction of a second synchronization.

The Timer registers are updated in the same cycle as the write instruction's execution by the CPU. The user must ensure that when RTCEN = 1, the updated registers will not be incremented at the same time. This can be accomplished in several ways:

- By checking the RTCSYNC bit (RTCCFG<4>)
- By checking the preceding digits from which a carry can occur
- By updating the registers immediately following the seconds pulse (or an alarm interrupt)

The user has visibility to the half-second field of the counter. This value is read-only and can be reset only by writing to the lower half of the SECONDS register.

### 18.2.6 SAFETY WINDOW FOR REGISTER READS AND WRITES

The RTCSYNC bit indicates a time window during which the RTCC Clock Domain registers can be safely read and written without concern about a rollover. When RTCSYNC = 0, the registers can be safely accessed by the CPU.

Whether RTCSYNC = 1 or 0, the user should employ a firmware solution to ensure that the data read did not fall on a rollover boundary, resulting in an invalid or partial read. This firmware solution would consist of reading each register twice and then comparing the two values. If the two values matched, then a rollover did not occur.

### 18.2.7 WRITE LOCK

In order to perform a write to any of the RTCC Timer registers, the RTCWREN bit (RTCCFG<5>) must be set.

To avoid accidental writes to the RTCC Timer register, it is recommended that the RTCWREN bit (RTCCFG<5>) be kept clear when not writing to the register. For the RTCWREN bit to be set, there is only one instruction cycle time window allowed between the 55h/AA sequence and the setting of RTCWREN. For that reason, it is recommended that users follow the code example in Example 18-1.

**EXAMPLE 18-1: SETTING THE RTCWREN BIT**

```
movlw   0x55
movwf   EECON2
movlw   0xAA
movwf   EECON2
bsf     RTCCFG,RTCWREN
```

### 18.2.8 REGISTER MAPPING

To limit the register interface, the RTCC Timer and Alarm Timer registers are accessed through corresponding register pointers. The RTCC Value register window (RTCVALH and RTCVALL) uses the RTCPTRx bits (RTCCFG<1:0>) to select the required Timer register pair.

By reading or writing to the RTCVALH register, the RTCC Pointer value (RTCPTR<1:0>) decrements by '1' until it reaches '00'. When '00' is reached, the MINUTES and SECONDS value is accessible through RTCVALH and RTCVALL until the pointer value is manually changed.

**TABLE 18-3: RTCVALH AND RTCVALL REGISTER MAPPING**

| RTCPTR<1:0> | RTCC Value Register Window | |
|---|---|---|
| | RTCVALH | RTCVALL |
| 00 | MINUTES | SECONDS |
| 01 | WEEKDAY | HOURS |
| 10 | MONTH | DAY |
| 11 | — | YEAR |

The Alarm Value register windows (ALRMVALH and ALRMVALL) use the ALRMPTR bits (ALRMCFG<1:0>) to select the desired alarm register pair.

By reading or writing to the ALRMVALH register, the Alarm Pointer value, ALRMPTR<1:0>, decrements by one until it reaches '00'. When it reaches '00', the ALRMMIN and ALRMSEC values are accessible through ALRMVALH and ALRMVALL until the pointer value is manually changed.

**TABLE 18-4: ALRMVAL REGISTER MAPPING**

| ALRMPTR<1:0> | Alarm Value Register Window | |
|---|---|---|
| | ALRMVALH | ALRMVALL |
| 00 | ALRMMIN | ALRMSEC |
| 01 | ALRMWD | ALRMHR |
| 10 | ALRMMNTH | ALRMDAY |
| 11 | — | — |

### 18.2.9 CALIBRATION

The real-time crystal input can be calibrated using the periodic auto-adjust feature. When properly calibrated, the RTCC can provide an error of less than three seconds per month.

To perform this calibration, find the number of error clock pulses and store the value into the lower half of the RTCCAL register. The eight-bit, signed value, loaded into RTCCAL, is multiplied by four and will be either added or subtracted from the RTCC timer, once every minute.

To calibrate the RTCC module:

1. Use another timer resource on the device to find the error of the 32.768 kHz crystal.
2. Convert the number of error clock pulses per minute (see Equation 18-1).

**EQUATION 18-1: CONVERTING ERROR CLOCK PULSES**

(Ideal Frequency (32,758) – Measured Frequency) * 60 = Error Clocks per Minute

   - If the oscillator is *faster* than ideal (negative result from Step 2), the RCFGCALL register value needs to be negative. This causes the specified number of clock pulses to be subtracted from the timer counter, once every minute.
   - If the oscillator is *slower* than ideal (positive result from Step 2), the RCFGCALL register value needs to be positive. This causes the specified number of clock pulses to be added to the timer counter, once every minute.
3. Load the RTCCAL register with the correct value.

Writes to the RTCCAL register should occur only when the timer is turned off or immediately after the rising edge of the seconds pulse.

**Note:** In determining the crystal's error value, it is the user's responsibility to include the crystal's initial error from drift due to temperature or crystal aging.

## 18.3 Alarm

The Alarm features and characteristics are:

- Configurable from half a second to one year
- Enabled using the ALRMEN bit (ALRMCFG<7>, Register 18-4)
- Offers one-time and repeat alarm options

### 18.3.1 CONFIGURING THE ALARM

The alarm feature is enabled using the ALRMEN bit. This bit is cleared when an alarm is issued. The bit will not be cleared if the CHIME bit = 1 or if ALRMRPT ≠ 0.

The interval selection of the alarm is configured through the ALRMCFG bits (AMASK<3:0>); see Figure 18-5. These bits determine which, and how many, digits of the alarm must match the clock value for the alarm to occur.

The alarm can also be configured to repeat based on a preconfigured interval. The number of times this occurs, after the alarm is enabled, is stored in the ALRMRPT register.

**Note:** While the alarm is enabled (ALRMEN = 1), changing any of the registers, other than the RTCCAL, ALRMCFG and ALRMRPT registers and the CHIME bit, can result in a false alarm event leading to a false alarm interrupt. To avoid this, only change the timer and alarm values while the alarm is disabled (ALRMEN = 0). It is recommended that the ALRMCFG and ALRMRPT registers and CHIME bit be changed when RTCSYNC = 0.


© 2009-2018 Microchip Technology Inc.

**FIGURE 18-5: ALARM MASK SETTINGS**

| Alarm Mask Setting AMASK<3:0> | Day of the Week | Month | Day | Hours | Minutes | Seconds |
|---|---|---|---|---|---|---|
| 0000 – Every half second<br>0001 – Every second | | | | | | |
| 0010 – Every 10 seconds | | | | | | s |
| 0011 – Every minute | | | | | | s s |
| 0100 – Every 10 minutes | | | | | m | s s |
| 0101 – Every hour | | | | | m m | s s |
| 0110 – Every day | | | | h h | m m | s s |
| 0111 – Every week | d | | | h h | m m | s s |
| 1000 – Every month | | | d d | h h | m m | s s |
| 1001 – Every year(1) | | m m | d d | h h | m m | s s |

**Note 1:** Annually, except when configured for February 29.

When ALRMCFG = 00 and the CHIME bit = 0 (ALRMCFG<6>), the repeat function is disabled and only a single alarm will occur. The alarm can be repeated up to 255 times by loading the ALRMRPT register with FFh.

After each alarm is issued, the ALRMRPT register is decremented by one. Once the register has reached '00', the alarm will be issued one last time.

After the alarm is issued a last time, the ALRMEN bit is cleared automatically and the alarm is turned off. Indefinite repetition of the alarm can occur if the CHIME bit = 1.

When CHIME = 1, the alarm is not disabled when the ALRMRPT register reaches '00', but it rolls over to FF and continues counting indefinitely.

### 18.3.2 ALARM INTERRUPT

At every alarm event, an interrupt is generated. Additionally, an alarm pulse output is provided that operates at half the frequency of the alarm.

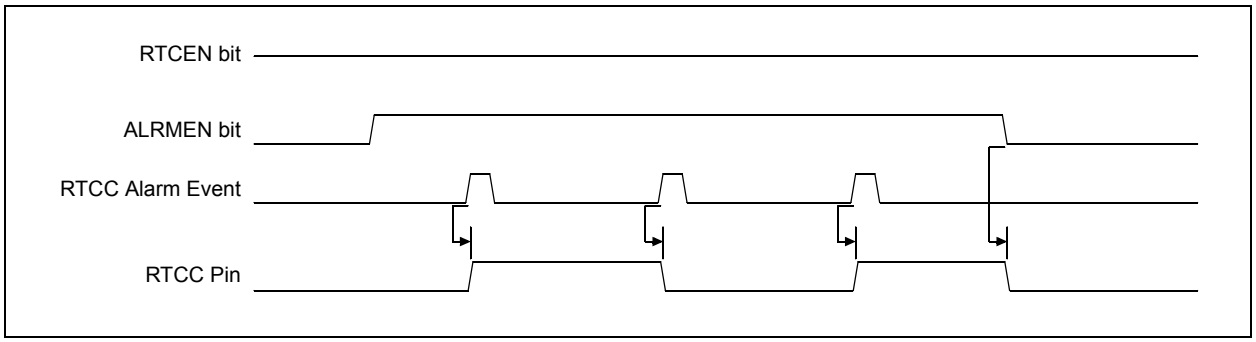
The alarm pulse output is completely synchronous with the RTCC clock and can be used as a trigger clock to other peripherals. This output is available on the RTCC pin. The output pulse is a clock with a 50% duty cycle and a frequency half that of the alarm event (see Figure 18-6).

The RTCC pin also can output the seconds clock. The user can select between the alarm pulse, generated by the RTCC module, or the seconds clock output.

The RTSECSEL<1:0> bits (PADCFG1<2:1>) select between these two outputs:

- Alarm pulse – RTSECSEL<1:0> = 00
- Seconds clock – RTSECSEL<1:0> = 01

**FIGURE 18-6: TIMER PULSE GENERATION**




## 18.4 Sleep Mode

The timer and alarm continue to operate while in Sleep mode. The operation of the alarm is not affected by Sleep, as an alarm event can always wake up the CPU.

The Idle mode does not affect the operation of the timer or alarm.

## 18.5 Reset

### 18.5.1 DEVICE RESET

When a device Reset occurs, the ALRMRPT register is forced to its Reset state, causing the alarm to be disabled (if enabled prior to the Reset). If the RTCC was enabled, it will continue to operate when a basic device Reset occurs.

### 18.5.2 POWER-ON RESET (POR)

The RTCCFG and ALRMRPT registers are reset only on a POR. Once the device exits the POR state, the clock registers should be reloaded with the desired values.

The timer prescaler values can be reset only by writing to the SECONDS register. No device Reset can affect the prescalers.


© 2009-2018 Microchip Technology Inc.

## 18.6 Register Maps

Table 18-5, Table 18-6 and Table 18-7 summarize the registers associated with the RTCC module.

**TABLE 18-5: RTCC CONTROL REGISTERS**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| RTCCFG | RTCEN | — | RTCWREN | RTCSYNC | HALFSEC | RTCOE | RTCPTR1 | RTCPTR0 |
| RTCCAL | CAL7 | CAL6 | CAL5 | CAL4 | CAL3 | CAL2 | CAL1 | CAL0 |
| PADCFG1 | RDPU | REPU | RJPU(1) | — | — | RTSECSEL1 | RTSECSEL0 | — |
| ALRMCFG | ALRMEN | CHIME | AMASK3 | AMASK2 | AMASK1 | AMASK0 | ALRMPTR1 | ALRMPTR0 |
| ALRMRPT | ARPT7 | ARPT6 | ARPT5 | ARPT4 | ARPT3 | ARPT2 | ARPT1 | ARPT0 |
| PMD1 | PSPMD | CTMUMD | RTCCMD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | EMBDM |

**Legend:** — = unimplemented, read as '0'. Reset values are shown in hexadecimal for 80-pin devices.
**Note 1:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.

**TABLE 18-6: RTCC VALUE REGISTERS**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| RTCVALH | RTCC Value High Register Window Based on RTCPTR<1:0> | | | | | | | |
| RTCVALL | RTCC Value Low Register Window Based on RTCPTR<1:0> | | | | | | | |

**TABLE 18-7: ALARM VALUE REGISTERS**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| ALRMVALH | Alarm Value High Register Window Based on ALRMPTR<1:0> | | | | | | | |
| ALRMVALL | Alarm Value Low Register Window Based on ALRMPTR<1:0> | | | | | | | |

## 19.0 CAPTURE/COMPARE/PWM (CCP) MODULES

PIC18F87K22 family devices have seven CCP (Capture/Compare/PWM) modules, designated CCP4 through CCP10. All the modules implement standard Capture, Compare and Pulse-Width Modulation (PWM) modes.

| | |
|---|---|
| **Note:** | Throughout this section, generic references are used for register and bit names that are the same, except for an 'x' variable that indicates the item's association with the specific CCP module. For example, the control register is named CCPxCON and refers to CCP4CON through CCP10CON. |

Each CCP module contains a 16-bit register that 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 through CCP10.

| | |
|---|---|
| **Note:** | The CCP9 and CCP10 modules are disabled on the devices with 32 Kbytes of program memory (PIC18FX5K22). |

**REGISTER 19-1: CCPxCON: CCPx CONTROL REGISTER (CCP4-CCP10 MODULES)(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 |
|---|---|---|---|---|---|---|---|
| PxM1 | PxM0 | DCxB1 | DCxB0 | CCPxM3(2) | CCPxM2(2) | CCPxM1(2) | CCPxM0(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 7-6 **PxM<1:0>:** PWM Output Configuration bits

If CCPxM<3:2> = 00, 01, 10:

xx = PxA is assigned as a capture/compare input/output; PxB, PxC and PxD are assigned as port pins

If CCPxM<3:2> = 11:

00 = Single output: PxA, PxB, PxC and PxD are controlled by steering

01 = Full-bridge output forward: PxD is modulated; PxA is active; PxB, PxC are inactive

10 = Half-bridge output: PxA, PxB are modulated with dead-band control; PxC and PxD are assigned as port pins

11 = Full-bridge output reverse: PxB is modulated; PxC is active; PxA and PxD are inactive

bit 5-4 **DCxB<1:0>:** PWM Duty Cycle bit 1 and bit 0 for CCPx Module

Capture mode:

Unused.

Compare mode:

Unused.

PWM mode:

These bits are the two Least Significant bits (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight Most Significant bits (DCx<9:2>) of the duty cycle are found in CCPRxL.

**Note 1:** The CCP9 and CCP10 modules are not available on devices with 32 Kbytes of program memory (PIC18FX5K22).

**2:** CCPxM<3:0> = 1011 will only reset the timer and not start AN A/D conversion on CCPx match.


© 2009-2018 Microchip Technology Inc.

**REGISTER 19-1: CCPxCON: CCPx CONTROL REGISTER (CCP4-CCP10 MODULES)(1)**

bit 3-0 **CCPxM<3:0>**: CCPx Module Mode Select bits(2)

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: Special Event Trigger; reset timer on CCPx match (CCPxIF bit is set)
11xx = PWM mode

**Note 1:** The CCP9 and CCP10 modules are not available on devices with 32 Kbytes of program memory (PIC18FX5K22).
**2:** CCPxM<3:0> = 1011 will only reset the timer and not start AN A/D conversion on CCPx match.

**REGISTER 19-2: CCPTMRS1: CCP TIMER SELECT REGISTER 1**

| R/W-0 | R/W-0 | U-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| C7TSEL1 | C7TSEL0 | — | C6TSEL0 | — | C5TSEL0 | C4TSEL1 | C4TSEL0 |
| 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 **C7TSEL<1:0>:** CCP7 Timer Selection bits
00 = CCP7 is based off of TMR1/TMR2
01 = CCP7 is based off of TMR5/TMR4
10 = CCP7 is based off of TMR5/TMR6
11 = CCP7 is based off of TMR5/TMR8

bit 5 **Unimplemented:** Read as '0'

bit 4 **C6TSEL0:** CCP6 Timer Selection bit
0 = CCP6 is based off of TMR1/TMR2
1 = CCP6 is based off of TMR5/TMR2

bit 3 **Unimplemented:** Read as '0'

bit 2 **C5TSEL0:** CCP5 Timer Selection bit
0 = CCP5 is based off of TMR1/TMR2
1 = CCP5 is based off of TMR5/TMR4

bit 1-0 **C4TSEL<1:0>:** CCP4 Timer Selection bits
00 = CCP4 is based off of TMR1/TMR2
01 = CCP4 is based off of TMR3/TMR4
10 = CCP4 is based off of TMR3/TMR6
11 = Reserved; do not use

**REGISTER 19-3: CCPTMRS2: CCP TIMER SELECT REGISTER 2**

| U-0 | U-0 | U-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | — | — | C10TSEL0(1) | — | C9TSEL0(1) | C8TSEL1 | C8TSEL0 |
| 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 **C10TSEL0:** CCP10 Timer Selection bit(1)
0 = CCP10 is based off of TMR1/TMR2
1 = CCP10 is based off of TMR7/TMR2

bit 3 **Unimplemented:** Read as '0'

bit 2 **C9TSEL0:** CCP9 Timer Selection bit(1)
0 = CCP9 is based off of TMR1/TMR2
1 = CCP9 is based off of TMR7/TMR4

bit 1-0 **C8TSEL<1:0>:** CCP8 Timer Selection bits
On Non 32-Byte Device Variants:
00 = CCP8 is based off of TMR1/TMR2
01 = CCP8 is based off of TMR7/TMR4
10 = CCP8 is based off of TMR7/TMR6
11 = Reserved; do not use
On 32-Byte Device Variants (PIC18F65K22 and PIC18F85K22):
00 = CCP8 is based off of TMR1/TMR2
01 = CCP8 is based off of TMR1/TMR4
10 = CCP8 is based off of TMR1/TMR6
11 = Reserved; do not use

**Note 1:** This bit is unimplemented and reads as '0' on devices with 32 Kbytes of program memory (PIC18FX-5K22).


© 2009-2018 Microchip Technology Inc.

**REGISTER 19-4: CCPRxL: CCPx PERIOD LOW BYTE REGISTER**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| CCPRxL7 | CCPRxL6 | CCPRxL5 | CCPRxL4 | CCPRxL3 | CCPRxL2 | CCPRxL1 | CCPRxL0 |
| 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 **CCPRxL<7:0>:** CCPx Period Register Low Byte bits
Capture Mode: Capture Register Low Byte
Compare Mode: Compare Register Low Byte
PWM Mode: Duty Cycle Register

**REGISTER 19-5: CCPRxH: CCPx PERIOD HIGH BYTE REGISTER**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| CCPRxH7 | CCPRxH6 | CCPRxH5 | CCPRxH4 | CCPRxH3 | CCPRxH2 | CCPRxH1 | CCPRxH0 |
| 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 **CCPRxH<7:0>:** CCPx Period Register High Byte bits
Capture Mode: Capture Register High Byte
Compare Mode: Compare Register High Byte
PWM Mode: Duty Cycle Buffer Register

## 19.1 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.

### 19.1.1 CCP MODULES AND TIMER RESOURCES

The CCP modules utilize Timers, 1 through 8, which vary with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in Table 19-1.

**TABLE 19-1: CCP MODE – TIMER RESOURCE**

| CCP Mode | Timer Resource |
|---|---|
| Capture | Timer1, Timer3, Timer 5 or Timer7 |
| Compare | |
| PWM | Timer2, Timer4, Timer 6 or Timer8 |

The assignment of a particular timer to a module is determined by the timer to CCP enable bits in the CCPTMRSx registers (see Register 19-2 and Register 19-3). All of the modules may be active at once and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time.

The CCPTMRS1 register selects the timers for CCP modules, 7, 6, 5 and 4, and the CCPTMRS2 register selects the timers for CCP modules, 10, 9 and 8. The possible configurations are shown in Table 19-2 and Table 19-3.

**TABLE 19-2: TIMER ASSIGNMENTS FOR CCP MODULES 4, 5, 6 AND 7**

| CCPTMRS1 Register | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CCP4 | | | CCP5 | | | CCP6 | | | CCP7 | | |
| C4TSEL <1:0> | Capture/ Compare Mode | PWM Mode | C5TSEL0 | Capture/ Compare Mode | PWM Mode | C6TSEL0 | Capture/ Compare Mode | PWM Mode | C7TSEL <1:0> | Capture/ Compare Mode | PWM Mode |
| 0 0 | TMR1 | TMR2 | 0 | TMR1 | TMR2 | 0 | TMR1 | TMR2 | 0 0 | TMR1 | TMR2 |
| 0 1 | TMR3 | TMR4 | 1 | TMR5 | TMR4 | 1 | TMR5 | TMR2 | 0 1 | TMR5 | TMR4 |
| 1 0 | TMR3 | TMR6 | | | | | | | 1 0 | TMR5 | TMR6 |
| 1 1 | Reserved(1) | | | | | | | | 1 1 | TMR5 | TMR8 |

**Note 1:** Do not use the reserved bits.

**TABLE 19-3: TIMER ASSIGNMENTS FOR CCP MODULES 8, 9 AND 10**

| CCPTMRS2 Register | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CCP8 | | | CCP8 Devices with 32 Kbytes | | | CCP9(1) | | | CCP10(1) | | |
| C8TSEL <1:0> | Capture/ Compare Mode | PWM Mode | C8TSEL <1:0> | Capture/ Compare Mode | PWM Mode | C9TSEL0 | Capture/ Compare Mode | PWM Mode | C10TSEL0 | Capture/ Compare Mode | PWM Mode |
| 0 0 | TMR1 | TMR2 | 0 0 | TMR1 | TMR2 | 0 | TMR1 | TMR2 | 0 | TMR1 | TMR2 |
| 0 1 | TMR7 | TMR4 | 0 1 | TMR1 | TMR4 | 1 | TMR7 | TMR4 | 1 | TMR7 | TMR2 |
| 1 0 | TMR7 | TMR6 | 1 0 | TMR1 | TMR6 | | | | | | |
| 1 1 | Reserved(2) | | 1 1 | Reserved(2) | | | | | | | |

**Note 1:** The module is not available for devices with 32 Kbytes of program memory (PIC18F65K22 and PIC18F85K22).

**2:** Do not use the reserved setting.


© 2009-2018 Microchip Technology Inc.

### 19.1.2 OPEN-DRAIN OUTPUT OPTION

When operating in Output mode (the Compare or PWM modes), the drivers for the CCPx pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor and allows the output to communicate with external circuits without the need for additional level shifters.

The open-drain output option is controlled by the CCPxOD bits (ODCON2<7:0>). Setting the appropriate bit configures the pin for the corresponding module for open-drain operation.

### 19.1.3 PIN ASSIGNMENT FOR CCP6, CCP7, CCP8 AND CCP9

The pin assignment for CCP6/7/8/9 (Capture input, Compare and PWM output) can change, based on the device configuration.

The ECCPMX Configuration bit (CONFIG3H<1>) determines the pin to which CCP6/7/8/9 is multiplexed. The pin assignments for these CCP modules are given in Table 19-4.

**TABLE 19-4: CCP PIN ASSIGNMENT**

| ECCPMX Value | Pin Mapped to | | | |
|---|---|---|---|---|
| | CCP6 | CCP7 | CCP8 | CC9 |
| 1 (Default) | RE6 | RE5 | RE4 | RE3 |
| 0 | RH7 | RH6 | RH5 | RH4 |

## 19.2 Capture Mode

In Capture mode, the CCPR4H:CCPR4L register pair captures the 16-bit value of the Timer register selected in the CCPTMRS1 when an event occurs on the CCP4 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, CCP4M<3:0> (CCP4CON<3:0>). When a capture is made, the interrupt request flag bit, CCP4IF (PIR4<1>), is set. (It must be cleared in software.) If another capture occurs before the value in CCPR4 is read, the old captured value is overwritten by the new captured value.

Figure 19-1 shows the Capture mode block diagram.

### 19.2.1 CCP PIN CONFIGURATION

In Capture mode, the appropriate CCPx pin should be configured as an input by setting the corresponding TRIS direction bit.

| **Note:** | If RC1 or RE7 is configured as a CCP4 output, a write to the port causes a capture condition. |
|---|---|

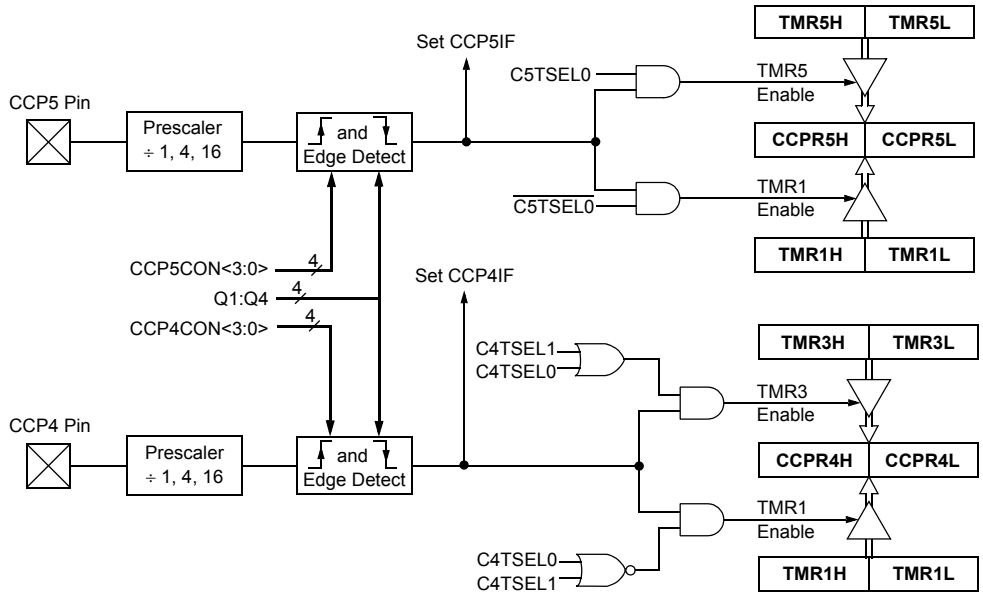
### 19.2.2 TIMER1/3/5/7 MODE SELECTION

For the available timers (1/3/5/7) to be used for the capture feature, the used timers must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work.

The timer to be used with each CCP module is selected in the CCPTMRSx registers. (See **Section 19.1.1 "CCP Modules and Timer Resources"**.)

Details of the timer assignments for the CCP modules are given in Table 19-2 and Table 19-3.

**FIGURE 19-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**




**Note:** This block diagram uses CCP4 and CCP5, and their appropriate timers as an example. For details on all of the CCP modules and their timer assignments, see Table 19-2 and Table 19-3.

### 19.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCP4IE bit (PIE4<1>) clear to avoid false interrupts and should clear the flag bit, CCP4IF, following any such change in operating mode.

### 19.2.4 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 (CCP4M<3:0>). Whenever the CCP module is turned off, or the CCP module is not in Capture mode, 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. Doing that will also not clear the prescaler counter – meaning the first capture may be from a non-zero prescaler.

Example 19-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

**EXAMPLE 19-1: CHANGING BETWEEN CAPTURE PRESCALERS**

```
CLRF   CCP4CON      ; Turn CCP module off
MOVLW  NEW_CAPT_PS  ; Load WREG with the
                    ; new prescaler mode
                    ; value and CCP ON
MOVWF  CCP4CON      ; Load CCP4CON with
                    ; this value
```


© 2009-2018 Microchip Technology Inc.

## 19.3 Compare Mode

In Compare mode, the 16-bit CCPR4 register value is constantly compared against the Timer register pair value selected in the CCPTMR1 register. When a match occurs, the CCP4 pin can be:

- Driven high
- Driven low
- Toggled (high-to-low or low-to-high)
- Unchanged (that is, reflecting the state of the I/O latch)

The action on the pin is based on the value of the mode select bits (CCP4M<3:0>). At the same time, the interrupt flag bit, CCP4IF, is set.

Figure 19-2 gives the Compare mode block diagram

### 19.3.1 CCP PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the appropriate TRIS bit.

| | |
|---|---|
| **Note:** | Clearing the CCP4CON register will force the RC1 or RE7 compare output latch (depending on device configuration) to the default low level. This is not the PORTC or PORTE I/O data latch. |

### 19.3.2 TIMER1/3/5/7 MODE SELECTION

If the CCP module is using the compare feature in conjunction with any of the Timer1/3/5/7 timers, the timers must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the compare operation may not work.

| | |
|---|---|
| **Note:** | Details of the timer assignments for the CCP modules are given in Table 19-2 and Table 19-3. |

### 19.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCP4M<3:0> = 1010), the CCP4 pin is not affected. Only a CCP interrupt is generated, if enabled, and the CCP4IE bit is set.

### 19.3.4 SPECIAL EVENT TRIGGER

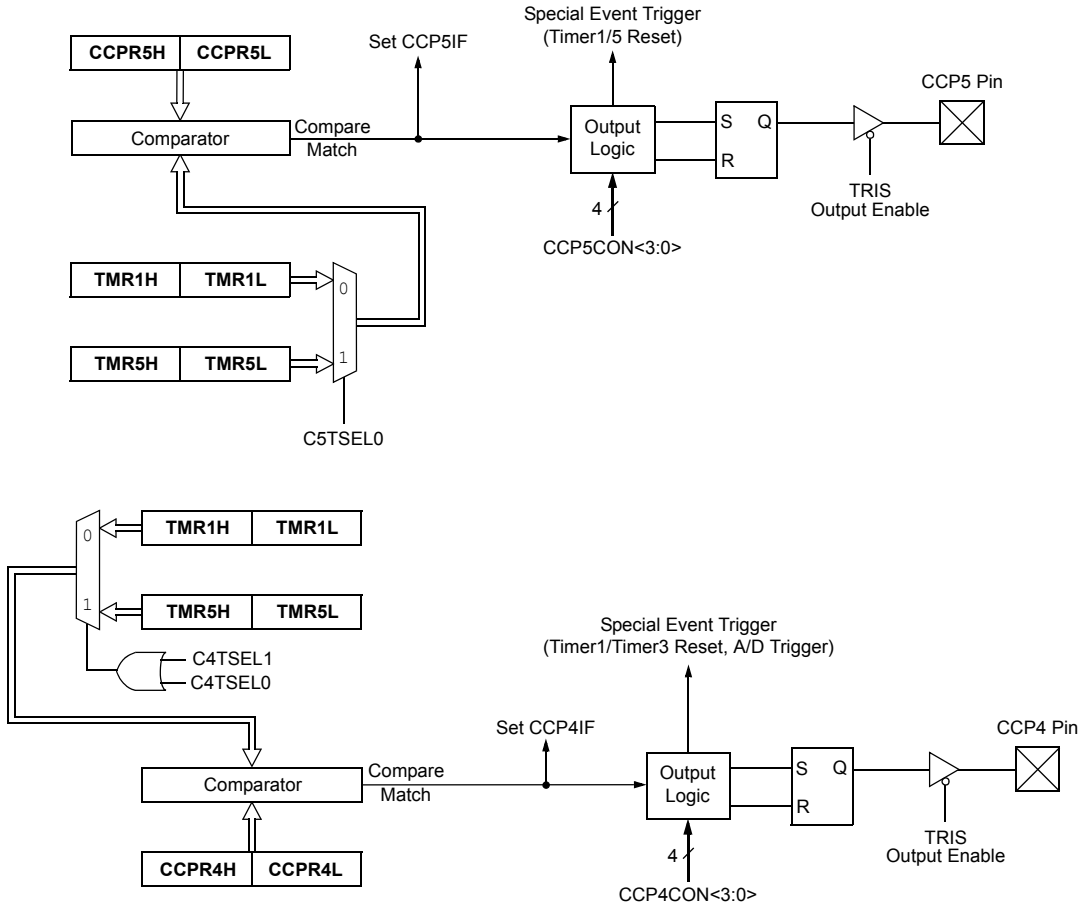
Both CCP modules are equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode (CCP4M<3:0> = 1011).

For either CCP module, the Special Event Trigger resets the Timer register pair for whichever timer resource is currently assigned as the module's time base. This allows the CCPRx registers to serve as a programmable Period register for either timer.

The Special Event Trigger for CCP4 cannot start an A/D conversion.

| | |
|---|---|
| **Note:** | The Special Event Trigger of ECCP2 can start an A/D conversion, but the A/D Converter must be enabled. For more information, see **Section 19.0 "Capture/Compare/PWM (CCP) Modules"**. |

**FIGURE 19-2: COMPARE MODE OPERATION BLOCK DIAGRAM**




**Note:** This block diagram uses CCP4 and CCP5, and their appropriate timers as an example. For details on all of the CCP modules and their timer assignments, see Table 19-2 and Table 19-3.

**TABLE 19-5: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1/3/5/7**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| RCON | IPEN | SBOREN | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| PIR4 | CCP10IF(1) | CCP9IF(1) | CCP8IF | CCP7IF | CCP6IF | CCP5IF | CCP4IF | CCP3IF |
| PIE4 | CCP10IE(1) | CCP9IE(1) | CCP8IE | CCP7IE | CCP6IE | CCP5IE | CCP4IE | CCP3IE |
| IPR4 | CCP10IP(1) | CCP9IP(1) | CCP8IP | CCP7IP | CCP6IP | CCP5IP | CCP4IP | CCP3IP |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 |
| TRISH(2) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 |
| TMR1L | Timer1 Register Low Byte | | | | | | | |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by Capture/Compare or Timer1/3/5/7.

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18F65K22 and PIC18F85K22).

**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.


© 2009-2018 Microchip Technology Inc.

**TABLE 19-5: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1/3/5/7 (CONTINUED)**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TMR1H | Timer1 Register High Byte | | | | | | | |
| TMR3L | Timer3 Register Low Byte | | | | | | | |
| TMR3H | Timer3 Register High Byte | | | | | | | |
| TMR5L | Timer5 Register Low Byte | | | | | | | |
| TMR5H | Timer5 Register High Byte | | | | | | | |
| TMR7L(1) | Timer7 Register Low Byte | | | | | | | |
| TMR7H(1) | Timer7 Register High Byte | | | | | | | |
| T1CON | TMR1CS1 | TMR1CS0 | T1CKPS1 | T1CKPS0 | SOSCEN | $\overline{\text{T1SYNC}}$ | RD16 | TMR1ON |
| T3CON | TMR3CS1 | TMR3CS0 | T3CKPS1 | T3CKPS0 | SOSCEN | $\overline{\text{T3SYNC}}$ | RD16 | TMR3ON |
| T5CON | TMR5CS1 | TMR5CS0 | T5CKPS1 | T5CKPS0 | SOSCEN | $\overline{\text{T5SYNC}}$ | RD16 | TMR5ON |
| T7CON(1) | TMR7CS1 | TMR7CS0 | T7CKPS1 | T7CKPS0 | SOSCEN | $\overline{\text{T7SYNC}}$ | RD16 | TMR7ON |
| CCPR4L | Capture/Compare/PWM Register 4 Low Byte | | | | | | | |
| CCPR4H | Capture/Compare/PWM Register 4 High Byte | | | | | | | |
| CCPR5L | Capture/Compare/PWM Register 5 Low Byte | | | | | | | |
| CCPR5H | Capture/Compare/PWM Register 5 High Byte | | | | | | | |
| CCPR6L | Capture/Compare/PWM Register 6 Low Byte | | | | | | | |
| CCPR6H | Capture/Compare/PWM Register 6 High Byte | | | | | | | |
| CCPR7L | Capture/Compare/PWM Register 7 Low Byte | | | | | | | |
| CCPR7H | Capture/Compare/PWM Register 7 High Byte | | | | | | | |
| CCPR8L | Capture/Compare/PWM Register 8 Low Byte | | | | | | | |
| CCPR8H | Capture/Compare/PWM Register 8 High Byte | | | | | | | |
| CCPR9L(1) | Capture/Compare/PWM Register 9 Low Byte | | | | | | | |
| CCPR9H(1) | Capture/Compare/PWM Register 9 High Byte | | | | | | | |
| CCPR10L(1) | Capture/Compare/PWM Register 10 Low Byte | | | | | | | |
| CCPR10H(1) | Capture/Compare/PWM Register 10 High Byte | | | | | | | |
| CCP4CON | — | — | DC4B1 | DC4B0 | CCP4M3 | CCP4M2 | CCP4M1 | CCP4M0 |
| CCP5CON | — | — | DC5B1 | DC5B0 | CCP5M3 | CCP5M2 | CCP5M1 | CCP5M0 |
| CCP6CON | — | — | DC6B1 | DC6B0 | CCP6M3 | CCP6M2 | CCP6M1 | CCP6M0 |
| CCP7CON | — | — | DC7B1 | DC7B0 | CCP7M3 | CCP7M2 | CCP7M1 | CCP7M0 |
| CCP8CON | — | — | DC8B1 | DC8B0 | CCP8M3 | CCP8M2 | CCP8M1 | CCP8M0 |
| CCP9CON(1) | — | — | DC9B1 | DC9B0 | CCP9M3 | CCP9M2 | CCP9M1 | CCP9M0 |
| CCP10CON(1) | — | — | DC10B1 | DC10B0 | CCP10M3 | CCP10M2 | CCP10M1 | CCP10M0 |
| CCPTMRS1 | C7TSEL1 | C7TSEL0 | — | C6TSEL0 | — | C5TSEL0 | C4TSEL1 | C4TSEL0 |
| CCPTMRS2 | — | — | — | C10TSEL0(1) | — | C9TSEL0(1) | C8TSEL1 | C8TSEL0 |
| PMD3 | CCP10MD(1) | CCP9MD(1) | CCP8MD | CCP7MD | CCP6MD | CCP5MD | CCP4MD | TMR12MD(1) |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by Capture/Compare or Timer1/3/5/7.

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18F65K22 and PIC18F85K22).

**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.

## 19.4 PWM Mode

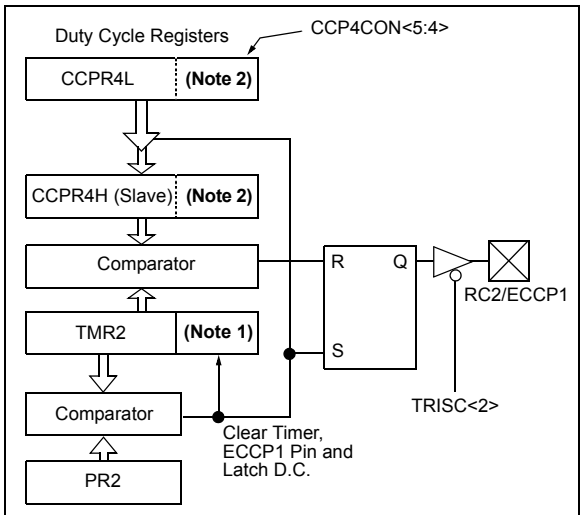
In Pulse-Width Modulation (PWM) mode, the CCP4 pin produces up to a 10-bit resolution PWM output. Since the CCP4 pin is multiplexed with a PORTC or PORTE data latch, the appropriate TRIS bit must be cleared to make the CCP4 pin an output.

| | |
|---|---|
| **Note:** | Clearing the CCP4CON register will force the RC1 or RE7 output latch (depending on device configuration) to the default low level. This is not the PORTC or PORTE I/O data latch. |

Figure 19-3 shows a simplified block diagram of the ECCP1 module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see **Section 19.4.3 "Setup for PWM Operation"**.

**FIGURE 19-3: SIMPLIFIED PWM BLOCK DIAGRAM**

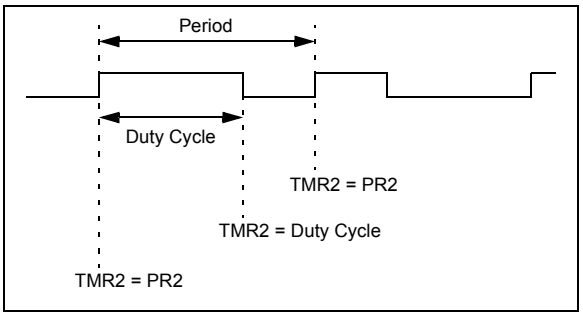



**Note 1:** The 8-bit TMR2 value is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler, to create the 10-bit time base.

**2:** CCP4 and its appropriate timers are used as an example. For details on all of the CCP modules and their timer assignments, see Table 19-2 and Table 19-3.

A PWM output (Figure 19-4) 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 19-4: PWM 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 following formula:

**EQUATION 19-1:**

$$\text{PWM Period} = [(\text{PR2}) + 1] \cdot 4 \cdot \text{TOSC} \cdot (\text{TMR2 Prescale Value})$$

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 CCP4 pin is set
  (An exception: If PWM duty cycle = 0%, the CCP4 pin will not be set)
- The PWM duty cycle is latched from CCPR4L into CCPR4H

| | |
|---|---|
| **Note:** | The Timer2 postscalers (see **Section 15.0 "Timer2 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. |


© 2009-2018 Microchip Technology Inc.

### 19.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified to use CCP4, as an example, by writing to the CCPR4L register and to the CCP4CON<5:4> bits. Up to 10-bit resolution is available. The CCPR4L contains the eight MSbs and the CCP4CON<5:4> bits contain the two LSbs. This 10-bit value is represented by CCPR4L:CCP4CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

**EQUATION 19-2:**

$$\text{PWM Duty Cycle} = (\text{CCPR4L:CCP4CON<5:4>}) \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

CCPR4L and CCP4CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR4H until after a match between PR2 and TMR2 occurs (that is, the period is complete). In PWM mode, CCPR4H is a read-only register.

The CCPR4H register and a two-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitchless PWM operation.

When the CCPR4H and two-bit latch match TMR2, concatenated with an internal two-bit Q clock or two bits of the TMR2 prescaler, the CCP4 pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is shown in 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 CCP4 pin will not be cleared.

**TABLE 19-6: 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 |

### 19.4.3 SETUP FOR PWM OPERATION

To configure the CCP module for PWM operation, using CCP4 as an example:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPR4L register and CCP4CON<5:4> bits.
3. Make the CCP4 pin an output by clearing the appropriate TRIS bit.
4. Set the TMR2 prescale value, then enable Timer2 by writing to T2CON.
5. Configure the CCP4 module for PWM operation.

**TABLE 19-7: REGISTERS ASSOCIATED WITH PWM AND TIMERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| RCON | IPEN | SBOREN | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| PIR4 | CCP10IF(1) | CCP9IF(1) | CCP8IF | CCP7IF | CCP6IF | CCP5IF | CCP4IF | CCP3IF |
| PIE4 | CCP10IE(1) | CCP9IE(1) | CCP8IE | CCP7IE | CCP6IE | CCP5IE | CCP4IE | CCP3IE |
| IPR4 | CCP10IP(1) | CCP9IP(1) | CCP8IP | CCP7IP | CCP6IP | CCP5IP | CCP4IP | CCP3IP |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PWM or Timer2/4/6/8.

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18F65K22 and PIC18F85K22).

**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.

**TABLE 19-7: REGISTERS ASSOCIATED WITH PWM AND TIMERS (CONTINUED)**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TRISH(2) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 |
| TMR2 | Timer2 Register | | | | | | | |
| TMR4 | Timer4 Register | | | | | | | |
| TMR6 | Timer6 Register | | | | | | | |
| TMR8 | Timer8 Register | | | | | | | |
| PR2 | Timer2 Period Register | | | | | | | |
| PR4 | Timer4 Period Register | | | | | | | |
| PR6 | Timer6 Period Register | | | | | | | |
| PR8 | Timer8 Period Register | | | | | | | |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| T4CON | — | T4OUTPS3 | T4OUTPS2 | T4OUTPS1 | T4OUTPS0 | TMR4ON | T4CKPS1 | T4CKPS0 |
| T6CON | — | T6OUTPS3 | T6OUTPS2 | T6OUTPS1 | T6OUTPS0 | TMR6ON | T6CKPS1 | T6CKPS0 |
| T8CON | — | T8OUTPS3 | T8OUTPS2 | T8OUTPS1 | T8OUTPS0 | TMR8ON | T8CKPS1 | T8CKPS0 |
| CCPR4L | Capture/Compare/PWM Register 4 Low Byte | | | | | | | |
| CCPR4H | Capture/Compare/PWM Register 4 High Byte | | | | | | | |
| CCPR5L | Capture/Compare/PWM Register 5 Low Byte | | | | | | | |
| CCPR5H | Capture/Compare/PWM Register 5 High Byte | | | | | | | |
| CCPR6L | Capture/Compare/PWM Register 6 Low Byte | | | | | | | |
| CCPR6H | Capture/Compare/PWM Register 6 High Byte | | | | | | | |
| CCPR7L | Capture/Compare/PWM Register 7 Low Byte | | | | | | | |
| CCPR7H | Capture/Compare/PWM Register 7 High Byte | | | | | | | |
| CCPR8L | Capture/Compare/PWM Register 8 Low Byte | | | | | | | |
| CCPR8H | Capture/Compare/PWM Register 8 High Byte | | | | | | | |
| CCPR9L(1) | Capture/Compare/PWM Register 9 Low Byte | | | | | | | |
| CCPR9H(1) | Capture/Compare/PWM Register 9 High Byte | | | | | | | |
| CCPR10L(1) | Capture/Compare/PWM Register 10 Low Byte | | | | | | | |
| CCPR10H(1) | Capture/Compare/PWM Register 10 High Byte | | | | | | | |
| CCP4CON | — | — | DC4B1 | DC4B0 | CCP4M3 | CCP4M2 | CCP4M1 | CCP4M0 |
| CCP5CON | — | — | DC5B1 | DC5B0 | CCP5M3 | CCP5M2 | CCP5M1 | CCP5M0 |
| CCP6CON | — | — | DC6B1 | DC6B0 | CCP6M3 | CCP6M2 | CCP6M1 | CCP6M0 |
| CCP7CON | — | — | DC7B1 | DC7B0 | CCP7M3 | CCP7M2 | CCP7M1 | CCP7M0 |
| CCP8CON | — | — | DC8B1 | DC8B0 | CCP8M3 | CCP8M2 | CCP8M1 | CCP8M0 |
| CCP9CON(1) | — | — | DC9B1 | DC9B0 | CCP9M3 | CCP9M2 | CCP9M1 | CCP9M0 |
| CCP10CON(1) | — | — | DC10B1 | DC10B0 | CCP10M3 | CCP10M2 | CCP10M1 | CCP10M0 |
| CCPTMRS1 | C7TSEL1 | C7TSEL0 | — | C6TSEL0 | — | C5TSEL0 | C4TSEL1 | C4TSEL0 |
| CCPTMRS2 | — | — | — | C10TSEL0(1) | — | C9TSEL0(1) | C8TSEL1 | C8TSEL0 |
| PMD3 | CCP10MD(1) | CCP9MD(1) | CCP8MD | CCP7MD | CCP6MD | CCP5MD | CCP4MD | TMR12MD(1) |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used by PWM or Timer2/4/6/8.

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18F65K22 and PIC18F85K22).

**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.


© 2009-2018 Microchip Technology Inc.

## 20.0 ENHANCED CAPTURE/COMPARE/PWM (ECCP) MODULE

PIC18F87K22 family devices have three Enhanced Capture/Compare/PWM (ECCP) modules: ECCP1, ECCP2 and ECCP3. These modules contain 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. These ECCP modules are upward compatible with CCP.

| | |
|---|---|
| **Note:** | Throughout this section, generic references are used for register and bit names that are the same, except for an 'x' variable that indicates the item's association with the ECCP1, ECCP2 or ECCP3 module. For example, the control register is named CCPxCON and refers to CCP1CON, CCP2CON and CCP3CON. |

ECCP1, ECCP2 and ECCP3 are implemented as standard CCP modules with Enhanced PWM capabilities. These include:

- Provision for two or four output channels
- Output Steering modes
- Programmable polarity
- Programmable dead-band control
- Automatic shutdown and restart

The enhanced features are discussed in detail in **Section 20.4 "PWM (Enhanced Mode)"**.

The ECCP1, ECCP2 and ECCP3 modules use the control registers: CCP1CON, CCP2CON and CCP3CON. The control registers, CCP4CON through CCP10CON, are for the modules, CCP4 through CCP10.

**REGISTER 20-1: CCPxCON: ENHANCED CAPTURE/COMPARE/PWMx CONTROL**

| 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 and PxD are assigned as port pins

If CCPxM<3:2> = 11:

00 = Single output: PxA, PxB, PxC and PxD are controlled by steering (see **Section 20.4.7 "Pulse Steering Mode"**)

01 = Full-bridge output forward: PxD is modulated; PxA is active; PxB, PxC are inactive

10 = Half-bridge output: PxA, PxB are modulated with dead-band control; PxC and PxD are assigned as port pins

11 = Full-bridge output reverse: PxB is modulated; PxC is active; PxA and PxD are inactive

bit 5-4 **DCxB<1:0>**: PWM Duty Cycle Bit 1 and Bit 0

Capture mode:

Unused.

Compare mode:

Unused.

PWM mode:

These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPRxL.

bit 3-0 **CCPxM<3:0>**: ECCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets ECCPx module)

0001 = Reserved

0010 = Compare mode: toggle output on match

0011 = Capture mode

0100 = Capture mode: every falling edge

0101 = Capture mode: every rising edge

0110 = Capture mode: every fourth rising edge

0111 = Capture mode: every 16$^{th}$ 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, starts A/D conversion, sets CCPxIF bit)

1100 = PWM mode: PxA and PxC are active-high; PxB and PxD are active-high

1101 = PWM mode: PxA and PxC are active-high; PxB and PxD are active-low

1110 = PWM mode: PxA and PxC are active-low; PxB and PxD are active-high

1111 = PWM mode: PxA and PxC are active-low; PxB and PxD are active-low


© 2009-2018 Microchip Technology Inc.

**REGISTER 20-2: CCPTMRS0: CCP TIMER SELECT 0 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 |
|---|---|---|---|---|---|---|---|
| C3TSEL1 | C3TSEL0 | C2TSEL2 | C2TSEL1 | C2TSEL0 | C1TSEL2 | C1TSEL1 | C1TSEL0 |
| 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 **C3TSEL<1:0>:** ECCP3 Timer Selection bits
00 = ECCP3 is based off of TMR1/TMR2
01 = ECCP3 is based off of TMR3/TMR4
10 = ECCP3 is based off of TMR3/TMR6
11 = ECCP3 is based off of TMR3/TMR8

bit 5-3 **C2TSEL<2:0>:** ECCP2 Timer Selection bits
000 = ECCP2 is based off of TMR1/TMR2
001 = ECCP2 is based off of TMR3/TMR4
010 = ECCP2 is based off of TMR3/TMR6
011 = ECCP2 is based off of TMR3/TMR8
100 = ECCP2 is based off of TMR3/TMR10: option reserved on the 32-Kbyte device variant; do not use
101 = Reserved; do not use
110 = Reserved; do not use
111 = Reserved; do not use

bit 2-0 **C1TSEL<2:0>:** ECCP1 Timer Selection bits
000 = ECCP1 is based off of TMR1/TMR2
001 = ECCP1 is based off of TMR3/TMR4
010 = ECCP1 is based off of TMR3/TMR6
011 = ECCP1 is based off of TMR3/TMR8
100 = ECCP1 is based off of TMR3/TMR10: option reserved on the 32-Kbyte device variant; do not use
101 = ECCP1 is based off of TMR3/TMR12: option reserved on the 32-Kbyte device variant; do not use
110 = Reserved; do not use
111 = Reserved; do not use

In addition to the expanded range of modes available through the CCPxCON and ECCPxAS registers, the ECCP modules have two additional registers associated with Enhanced PWM operation and auto-shutdown features. They are:

- ECCPxDEL – Enhanced PWM Control
- PSTRxCON – Pulse Steering Control

## 20.1 ECCP Outputs and Configuration

The Enhanced CCP module may have up to four PWM outputs, depending on the selected operating mode. The CCPxCON register is modified to allow control over four PWM outputs: ECCPx/PxA, PxB, PxC and PxD. Applications can use one, two or four of these outputs.

The outputs that are active depend on the ECCP selected operating mode. The pin assignments are summarized in Table 20-3.

To configure the I/O pins as PWM outputs, the proper PWM mode must be selected by setting the PxM<1:0> and CCPxM<3:0> bits. The appropriate TRIS direction bits for the port pins must also be set as outputs.

### 20.1.1 ECCP MODULE AND TIMER RESOURCES

The ECCP modules use Timers, 1, 2, 3, 4, 6, 8, 10 or 12, depending on the mode selected. These timers are available to CCP modules in Capture, Compare or PWM modes, as shown in Table 20-1.

**TABLE 20-1: ECCP MODE – TIMER RESOURCE**

| ECCP Mode | Timer Resource |
|---|---|
| Capture | Timer1 or Timer3 |
| Compare | Timer1 or Timer3 |
| PWM | Timer2, Timer4, Timer6, Timer8, Timer10 or Timer12 |

The assignment of a particular timer to a module is determined by the timer to ECCP enable bits in the CCPTMRSx register (Register 20-2). The interactions between the two modules are depicted in Figure 20-1. Capture operations are designed to be used when the timer is configured for Synchronous Counter mode. Capture operations may not work as expected if the associated timer is configured for Asynchronous Counter mode.

### 20.1.2 ECCP PIN ASSIGNMENT

The pin assignment for ECCPx (Capture input, Compare and PWM output) can change, based on device configuration. The ECCPMX (CONFIG3H<1>) Configuration bit determines which pins ECCP1 and ECCP3 are multiplexed to.

- Default/ECCPMX = 1:
  - ECCP1 (P1B/P1C) is multiplexed onto RE6 and RE5
  - ECCP3 (P3B/P3C) is multiplexed onto RE4 and RE3
- ECCPMX = 0:
  - ECCP1 (P1B/P1C) is multiplexed onto RH7 and RH6
  - ECCP3 (P3B/P3C) is multiplexed onto RH5 and RH4.

The pin assignment for ECCP2 (Capture input, Compare and PWM output) can change, based on the device configuration.

The CCP2MX Configuration bit (CONFIG3H<0>) determines which pin ECCP2 is multiplexed to.

- If CCP2MX = 1 (default) – ECCP2 is multiplexed to RC1
- If CCP2MX = 0 – ECCP2 is multiplexed to RE7


© 2009-2018 Microchip Technology Inc.

## 20.2 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 ECCPx pin. An event is defined as one of the following:

- Every falling edge
- Every rising edge
- Every fourth 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 (see Table 20-2). The flag must be cleared by software. If another capture occurs before the value in the CCPRxH/L register is read, the old captured value is overwritten by the new captured value.

**TABLE 20-2: ECCP1/2/3 INTERRUPT FLAG BITS**

| ECCP Module | Flag Bit |
|---|---|
| 1 | PIR3<1> |
| 2 | PIR3<2> |
| 3 | PIR4<0> |

### 20.2.1 ECCP PIN CONFIGURATION

In Capture mode, the appropriate ECCPx pin should be configured as an input by setting the corresponding TRIS direction bit.

| | |
|---|---|
| **Note:** | If the ECCPx pin is configured as an output, a write to the port can cause a capture condition. |

### 20.2.2 TIMER1/2/3/4/6/8/10/12 MODE SELECTION

The timers that are to be used with the capture feature (Timer1 2, 3, 4, 6, 8, 10 or 12) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work. The timer to be used with each ECCP module is selected in the CCPTMRS0 register (Register 20-2).

### 20.2.3 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.

### 20.2.4 ECCP PRESCALER

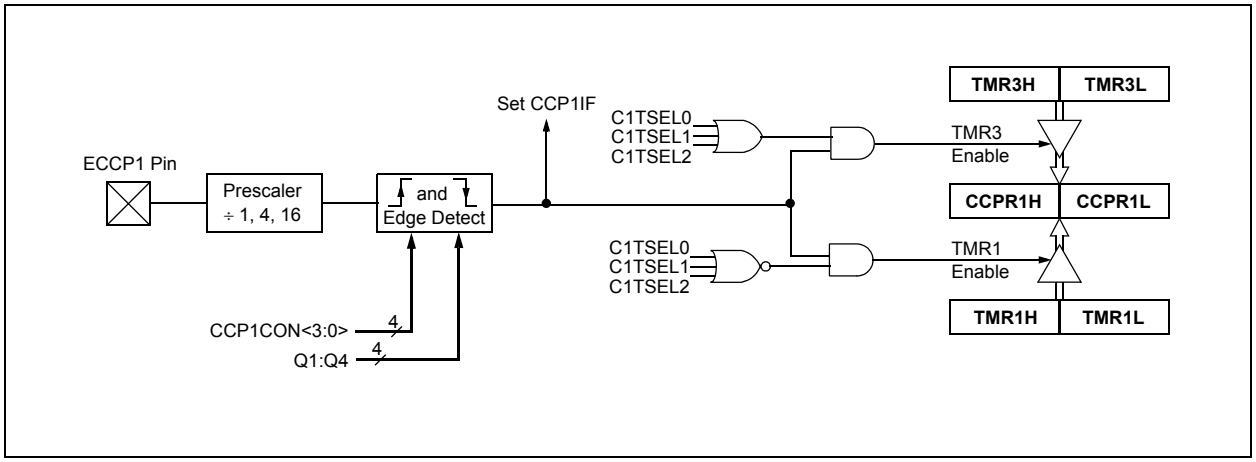
There are four prescaler settings in Capture mode; they are specified as part of the operating mode selected by the mode select bits (CCPxM<3:0>). Whenever the ECCP module is turned off, or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler 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 20-1 provides the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

**EXAMPLE 20-1: CHANGING BETWEEN CAPTURE PRESCALERS**

```
CLRF   CCP1CON     ; Turn ECCP module off
MOVLW  NEW_CAPT_PS ; Load WREG with the
                   ; new prescaler mode
                   ; value and ECCP ON
MOVWF  CCP1CON     ; Load CCP1CON with
                   ; this value
```

**FIGURE 20-1: CAPTURE MODE OPERATION BLOCK DIAGRAM**





© 2009-2018 Microchip Technology Inc.

## 20.3 Compare Mode

In Compare mode, the 16-bit CCPRx register value is constantly compared against the Timer register pair value selected in the CCPTMR1 register. When a match occurs, the ECCPx pin can be:

- Driven high
- Driven low
- Toggled (high-to-low or low-to-high)
- Unchanged (that is, reflecting 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.

### 20.3.1 ECCP PIN CONFIGURATION

Users must configure the ECCPx pin as an output by clearing the appropriate TRIS bit.

| | |
|---|---|
| **Note:** | Clearing the CCPxCON register will force the ECCPx compare output latch (depending on device configuration) to the default low level. This is not the PORTx I/O data latch. |

### 20.3.2 TIMER1/2/3/4/6/8/10/12 MODE SELECTION

Timer1 2, 3, 4, 6, 8, 10 or 12 must be running in Timer mode or Synchronized Counter mode if the ECCP module is using the compare feature. In Asynchronous Counter mode, the compare operation will not work reliably.

### 20.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCPxM<3:0> = `1010`), the ECCPx pin is not affected; only the CCPxIF interrupt flag is affected.

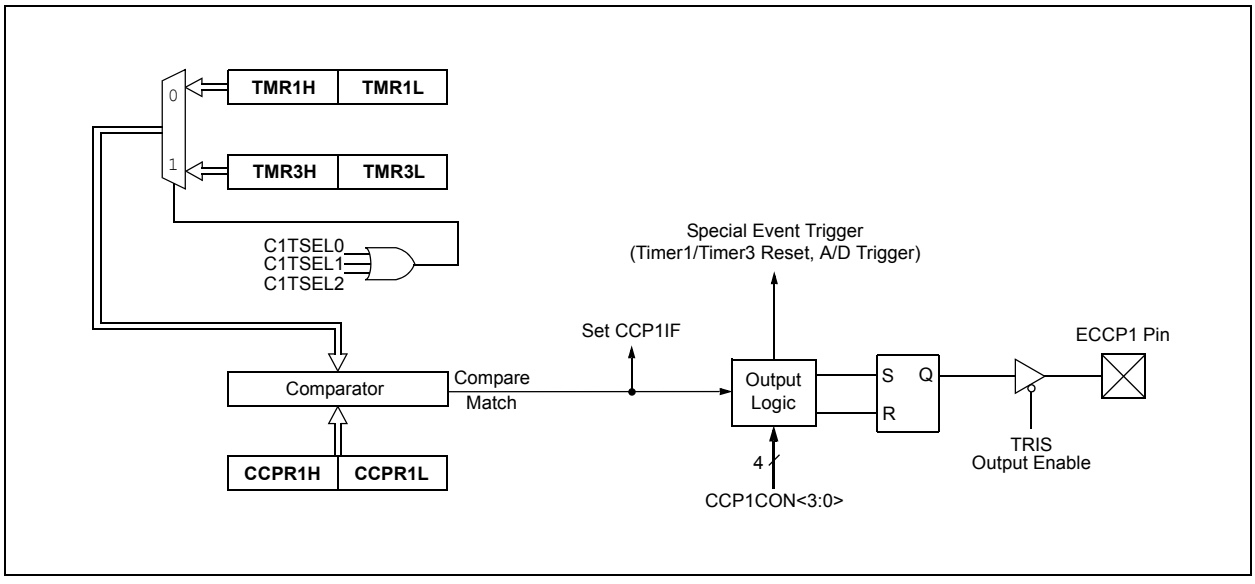
### 20.3.4 SPECIAL EVENT TRIGGER

The ECCP module is equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode (CCPxM<3:0> = `1011`).

The Special Event Trigger resets the Timer register pair for whichever timer resource is currently assigned as the module's time base. This allows the CCPRx registers to serve as a programmable Period register for either timer.

The Special Event Trigger can also start an A/D conversion. In order to do this, the A/D Converter must already be enabled.

**FIGURE 20-2: COMPARE MODE OPERATION BLOCK DIAGRAM**



## 20.4 PWM (Enhanced Mode)

The Enhanced PWM mode can generate a PWM signal on up to four different output pins with up to 10 bits of resolution. It can do this through four different PWM Output modes:

- Single PWM
- Half-Bridge PWM
- Full-Bridge PWM, Forward mode
- Full-Bridge PWM, Reverse mode

To select an Enhanced PWM mode, the PxM bits of the CCPxCON register must be set appropriately.

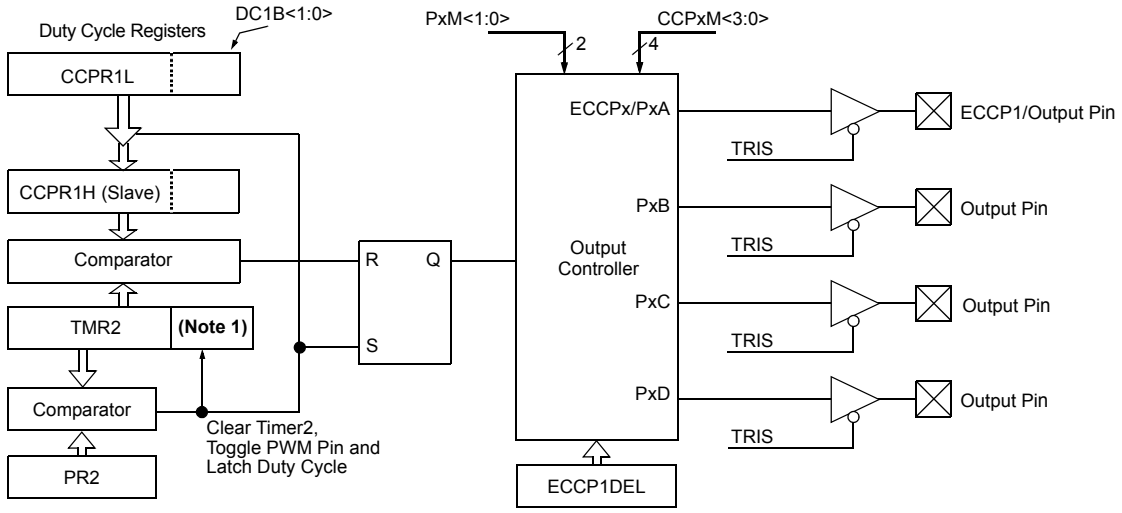
The PWM outputs are multiplexed with I/O pins and are designated: PxA, PxB, PxC and PxD. The polarity of the PWM pins is configurable and is selected by setting the CCPxM bits in the CCPxCON register appropriately.

Table 20-1 provides the pin assignments for each Enhanced PWM mode.

Figure 20-3 provides an example of a simplified block diagram of the Enhanced PWM module.

**Note:** To prevent the generation of an incomplete waveform when the PWM is first enabled, the ECCP module waits until the start of a new PWM period before generating a PWM signal.

**FIGURE 20-3: EXAMPLE SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODE**




**Note 1:** 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.

**Note 1:** The TRIS register value for each PWM output must be configured appropriately.

**2:** Any pin not used by an Enhanced PWM mode is available for alternate pin functions.

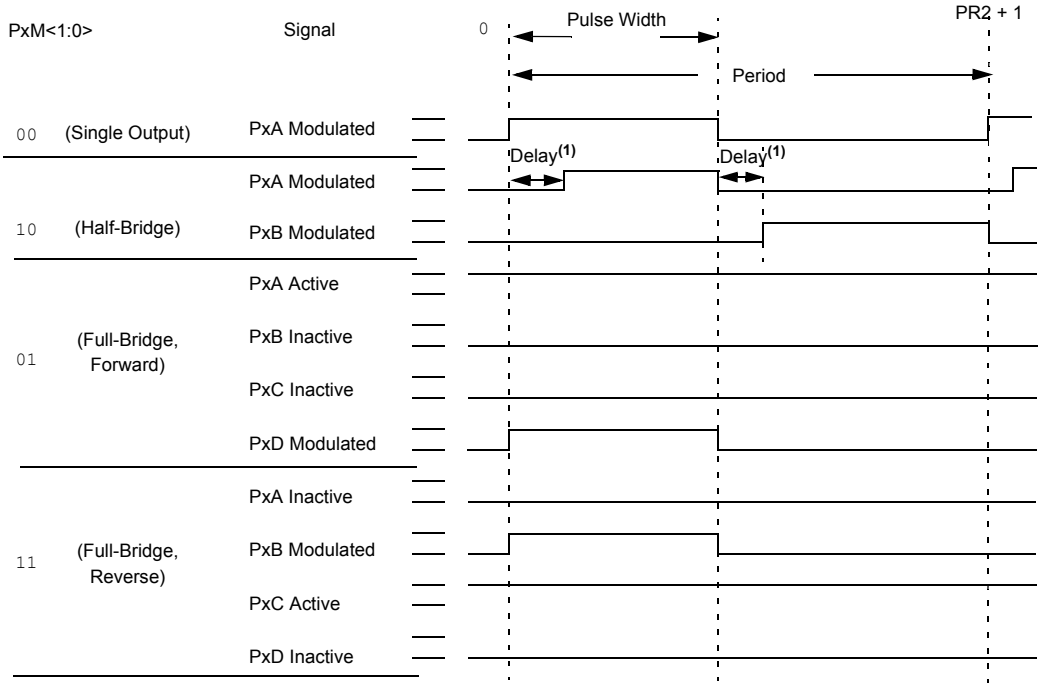

© 2009-2018 Microchip Technology Inc.

**TABLE 20-3: EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES**

| ECCP Mode | PxM<1:0> | PxA | PxB | PxC | PxD |
|---|---|---|---|---|---|
| Single | 00 | Yes(1) | Yes(1) | Yes(1) | Yes(1) |
| Half-Bridge | 10 | Yes | Yes | No | No |
| Full-Bridge, Forward | 01 | Yes | Yes | Yes | Yes |
| Full-Bridge, Reverse | 11 | Yes | Yes | Yes | Yes |

**Note 1:** Outputs are enabled by pulse steering in Single mode (see Register 20-5).

**FIGURE 20-4: EXAMPLE PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)**

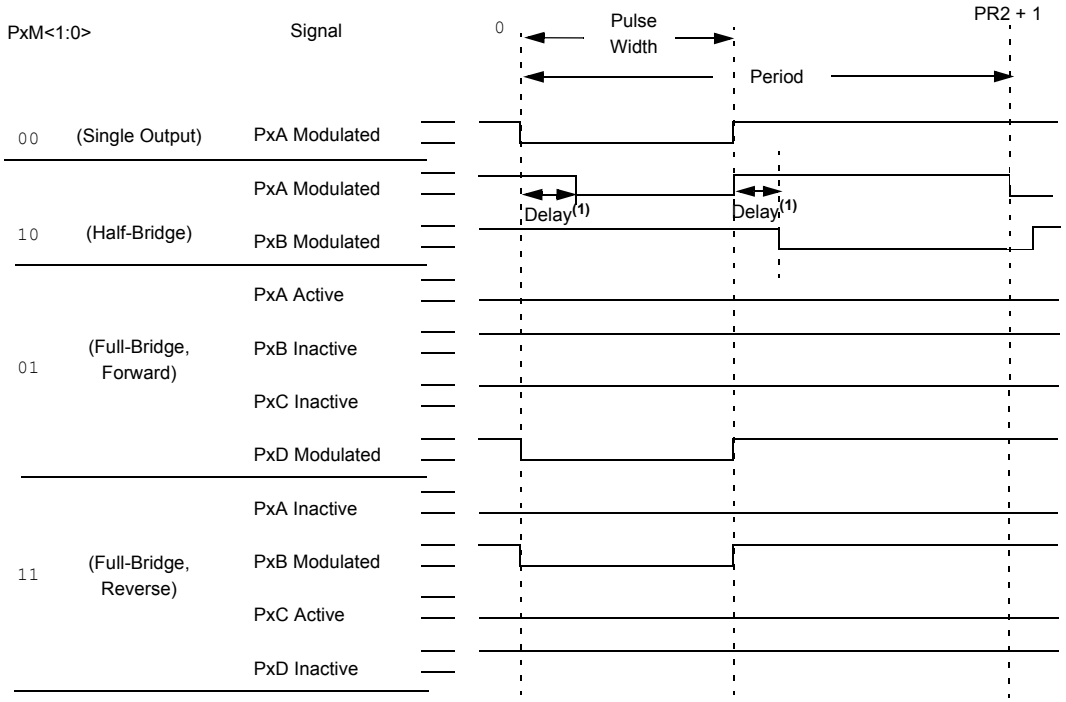



Relationships:

- Period = 4 * TOSC * (PR2 + 1) * (TMR2 Prescale Value)
- Pulse Width = TOSC * (CCPRxL<7:0>:CCPxCON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * TOSC * (ECCPxDEL<6:0>)

**Note 1:** Dead-band delay is programmed using the ECCPxDEL register (see **Section 20.4.6 "Programmable Dead-Band Delay Mode"**).

**FIGURE 20-5: EXAMPLE ENHANCED PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)**




Relationships:

- Period = 4 * TOSC * (PR2 + 1) * (TMR2 Prescale Value)
- Pulse Width = TOSC * (CCPRxL<7:0>:CCPxCON<5:4>) * (TMR2 Prescale Value)
- Delay = 4 * TOSC * (ECCPxDEL<6:0>)

**Note 1:** Dead-band delay is programmed using the ECCP1DEL register (see **Section 20.4.6 "Programmable Dead-Band Delay Mode"**).


© 2009-2018 Microchip Technology Inc.

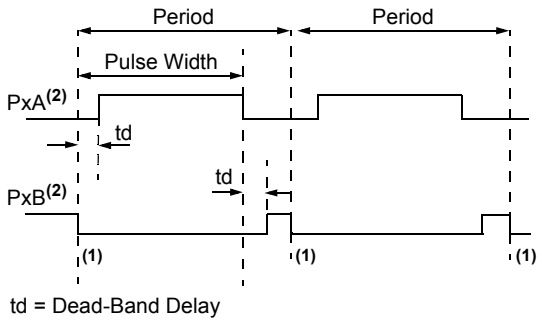
### 20.4.1 HALF-BRIDGE MODE

In Half-Bridge mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the PxA pin, while the complementary PWM output signal is output on the PxB pin (see Figure 20-6). This mode can be used for half-bridge applications, as shown in Figure 20-7, or for full-bridge applications, where four power switches are being modulated with two PWM signals.

In Half-Bridge mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of the PxDC<6:0> bits of the ECCPxDEL register sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. For more details on the dead-band delay operations, see **Section 20.4.6 "Programmable Dead-Band Delay Mode"**.

Since the PxA and PxB outputs are multiplexed with the port data latches, the associated TRIS bits must be cleared to configure PxA and PxB as outputs.

**FIGURE 20-6: EXAMPLE OF HALF-BRIDGE PWM OUTPUT**

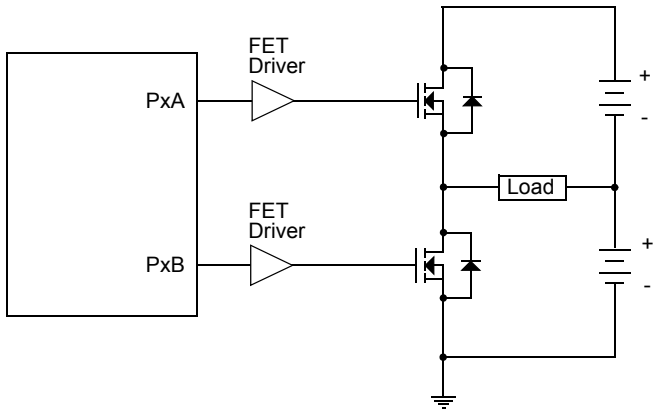



**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

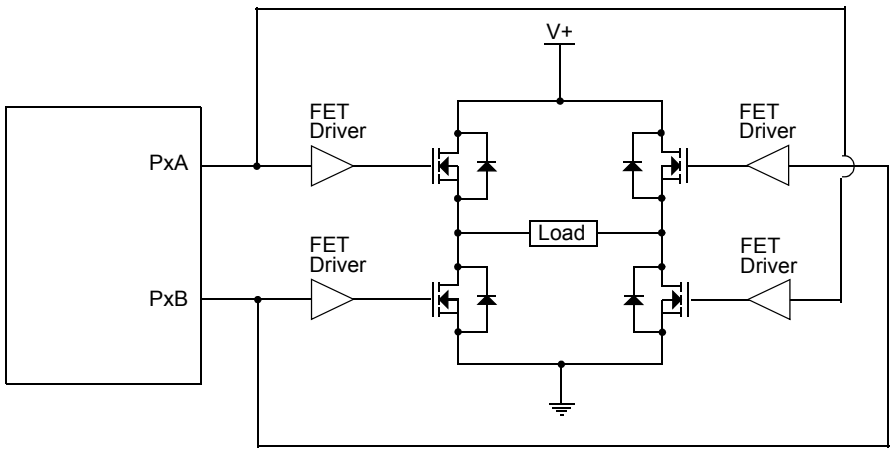
**2:** Output signals are shown as active-high.

**FIGURE 20-7: EXAMPLE OF HALF-BRIDGE APPLICATIONS**

**Standard Half-Bridge Circuit ("Push-Pull")**




**Half-Bridge Output Driving a Full-Bridge Circuit**



### 20.4.2 FULL-BRIDGE MODE

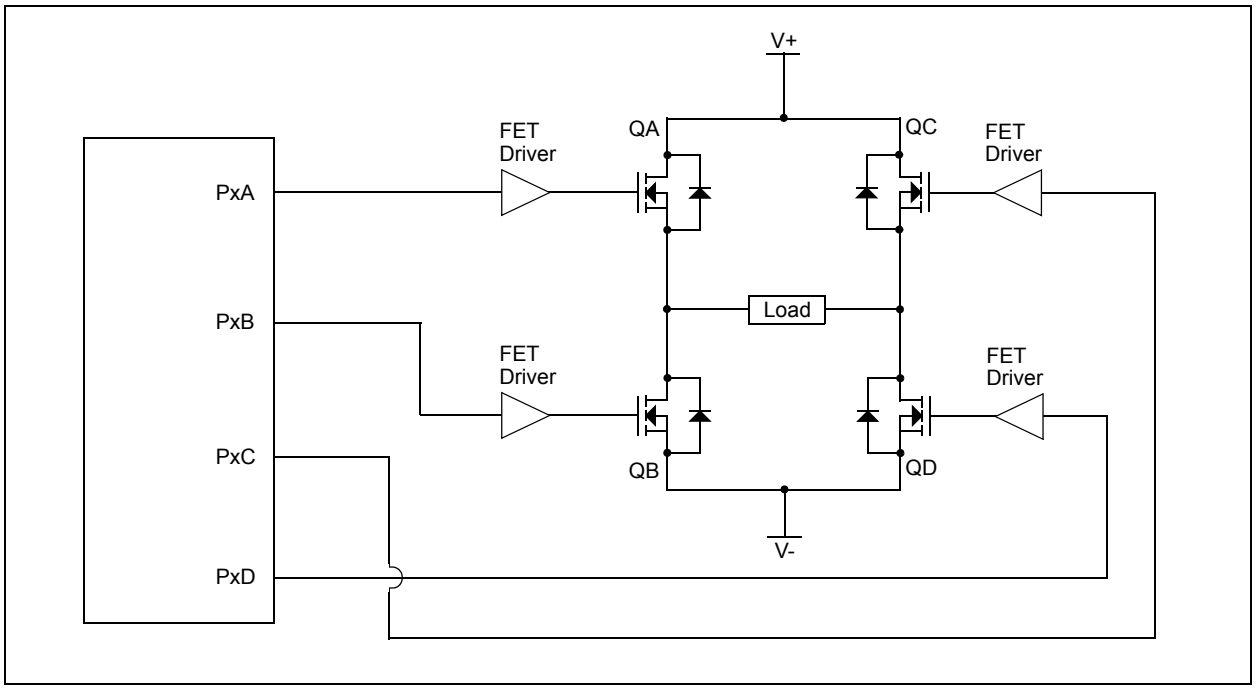
In Full-Bridge mode, all four pins are used as outputs. An example of a full-bridge application is provided in Figure 20-8.

In the Forward mode, the PxA pin is driven to its active state and the PxD pin is modulated, while the PxB and PxC pins are driven to their inactive state, as provided in Figure 20-9.

In the Reverse mode, the PxC pin is driven to its active state and the PxB pin is modulated, while the PxA and PxD pins are driven to their inactive state, as provided Figure 20-9.

The PxA, PxB, PxC and PxD outputs are multiplexed with the port data latches. The associated TRIS bits must be cleared to configure the PxA, PxB, PxC and PxD pins as outputs.

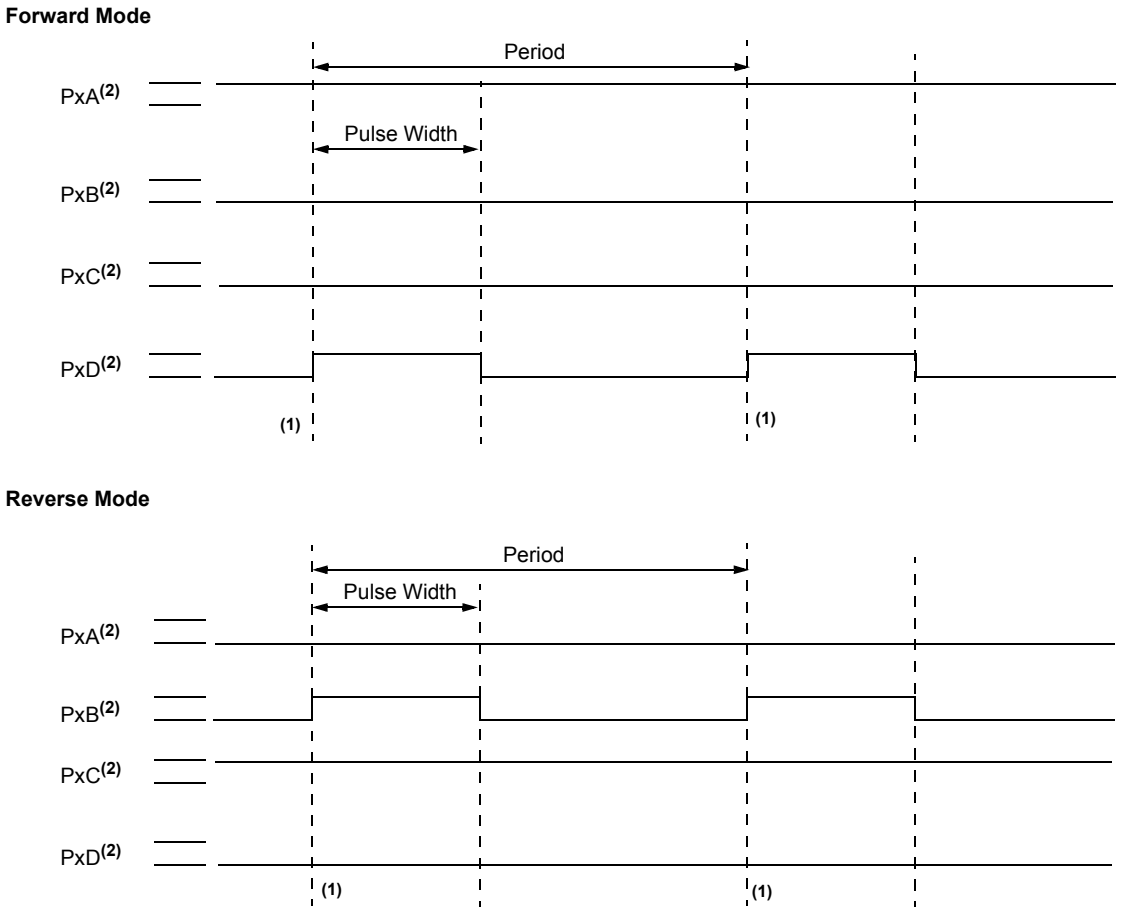
**FIGURE 20-8: EXAMPLE OF FULL-BRIDGE APPLICATION**





© 2009-2018 Microchip Technology Inc.

**FIGURE 20-9: EXAMPLE OF FULL-BRIDGE PWM OUTPUT**




**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

**2:** The output signal is shown as active-high.

#### 20.4.2.1 Direction Change in Full-Bridge Mode

In Full-Bridge mode, the PxM1 bit in the CCPxCON register allows users to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will change to the new direction on the next PWM cycle.

A direction change is initiated in software by changing the PxM1 bit of the CCPxCON register. The following sequence occurs prior to the end of the current PWM period:

- The modulated outputs (PxB and PxD) are placed in their inactive state.
- The associated unmodulated outputs (PxA and PxC) are switched to drive in the opposite direction.
- PWM modulation resumes at the beginning of the next period.

For an illustration of this sequence, see Figure 20-10.

The Full-Bridge mode does not provide a dead-band delay. As one output is modulated at a time, a dead-band delay is generally not required. There is a situation where a dead-band delay is required. This situation occurs when both of the following conditions are true:

- The direction of the PWM output changes when the duty cycle of the output is at or near 100%.
- The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

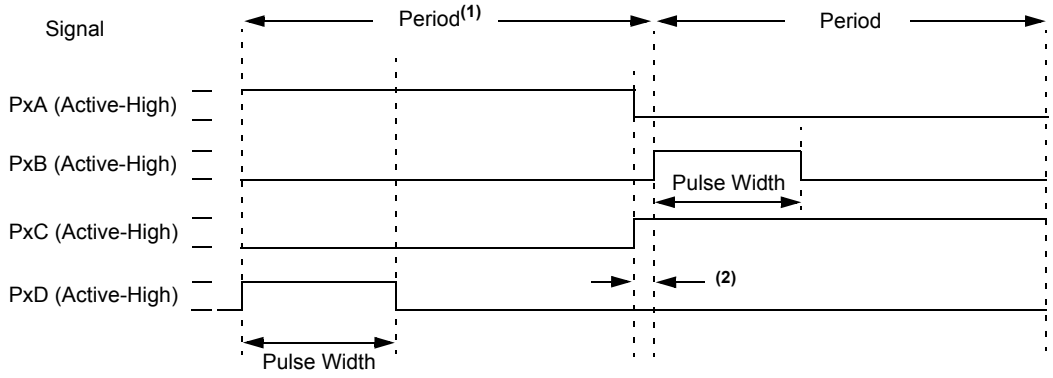
Figure 20-11 shows an example of the PWM direction changing from forward to reverse, at a near 100% duty cycle. In this example, at time, t1, the PxA and PxD outputs become inactive, while the PxC output becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through power devices, QC and QD (see Figure 20-8), for the duration of 't'. The same phenomenon will occur to power devices, QA and QB, for PWM direction change from reverse to forward.

If changing PWM direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

- Reduce PWM duty cycle for one PWM period before changing directions.
- 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 20-10: EXAMPLE OF PWM DIRECTION CHANGE**

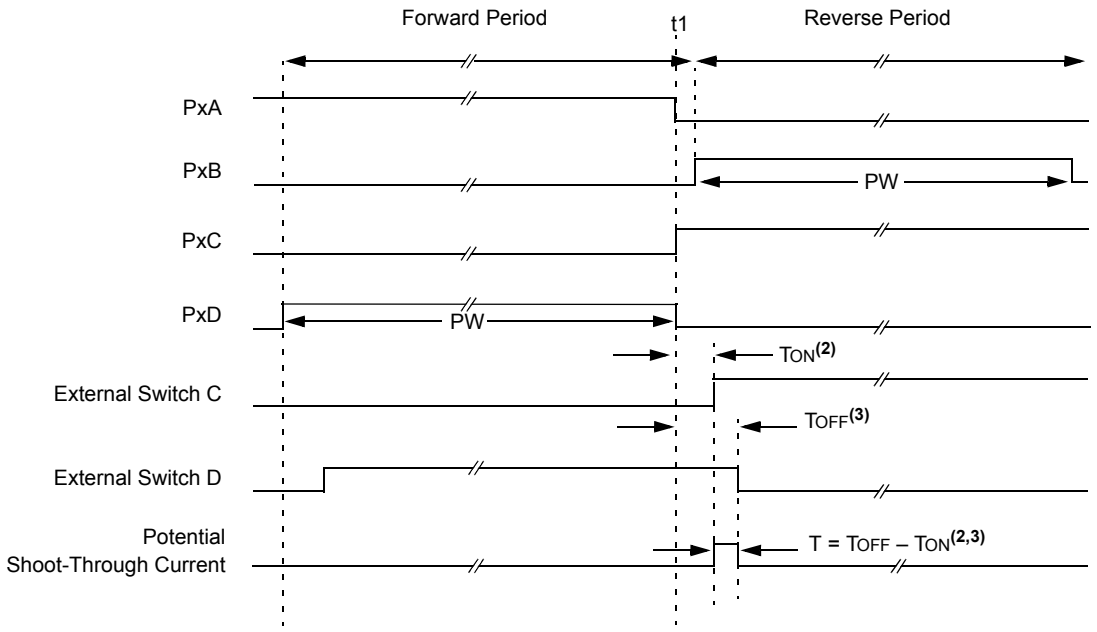



**Note 1:** The direction bit, PxM1 of the CCPxCON register, is written any time during the PWM cycle.

**2:** When changing directions, the PxA and PxC signals switch before the end of the current PWM cycle. The modulated PxB and PxD signals are inactive at this time. The length of this time is: (1/FOSC) • TMR2 Prescale Value.


© 2009-2018 Microchip Technology Inc.

**FIGURE 20-11: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE(1)**




**Note 1:** All signals are shown as active-high.

**2:** TON is the turn-on delay of power switch, QC, and its driver.

**3:** TOFF is the turn-off delay of power switch, QD, and its driver.

### 20.4.3 START-UP CONSIDERATIONS

When any PWM mode is used, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins.

| **Note:** | When the microcontroller is released from Reset, all of the I/O pins are in the high-impedance state. The external circuits must keep the power switch devices in the OFF state until the microcontroller drives the I/O pins with the proper signal levels or activates the PWM output(s). |
|---|---|

The CCPxM<1:0> bits of the CCPxCON register allow the user to choose whether the PWM output signals are active-high or active-low for each pair of PWM output pins (PxA/PxC and PxB/PxD). The PWM output polarities must be selected before the PWM pin output drivers are enabled. Changing the polarity configuration while the PWM pin output drivers are enabled is not recommended since it may result in damage to the application circuits.

The PxA, PxB, PxC and PxD output latches may not be in the proper states when the PWM module is initialized. Enabling the PWM pin output drivers at the same time as the Enhanced PWM modes may cause damage to the application circuit. The Enhanced PWM modes must be enabled in the proper Output mode and complete a full PWM cycle before enabling the PWM pin output drivers. The completion of a full PWM cycle is indicated by the TMR2IF or TMR4IF bit of the PIR1 or PIR5 register being set as the second PWM period begins.

### 20.4.4 ENHANCED PWM AUTO-SHUTDOWN MODE

The PWM mode supports an Auto-Shutdown mode that will disable the PWM outputs when an external shutdown event occurs. Auto-Shutdown mode places the PWM output pins into a predetermined state. This mode is used to help prevent the PWM from damaging the application.

The auto-shutdown sources are selected using the ECCPxAS<2:0> bits (ECCPxAS<6:4>). A shutdown event may be generated by:

- A logic '0' on the pin that is assigned the FLT0 input function
- C1 Comparator
- C2 Comparator
- Setting the ECCPxASE bit in firmware

A shutdown condition is indicated by the ECCPxASE (Auto-Shutdown Event Status) bit (ECCPxAS<7>). If the bit is a '0', the PWM pins are operating normally. If the bit is a '1', the PWM outputs are in the shutdown state.

When a shutdown event occurs, two things happen:

- The ECCPxASE bit is set to '1'. The ECCPxASE will remain set until cleared in firmware or an auto-restart occurs. (See **Section 20.4.5 "Auto-Restart Mode"**.)
- The enabled PWM pins are asynchronously placed in their shutdown states. The PWM output pins are grouped into pairs (PxA/PxC) and (PxB/PxD). The state of each pin pair is determined by the PSSxAC and PSSxBD bits (ECCPxAS<3:2> and <1:0>, respectively).

Each pin pair may be placed into one of three states:

- Drive logic '1'
- Drive logic '0'
- Tri-state (high-impedance)

**REGISTER 20-3: 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:** ECCP Auto-Shutdown Event Status bit
1 = A shutdown event has occurred; ECCP outputs are in a shutdown state
0 = ECCP outputs are operating

bit 6-4 **ECCPxAS<2:0>:** ECCP Auto-Shutdown Source Select bits
000 = Auto-shutdown is disabled
001 = Comparator C1OUT output is high
010 = Comparator C2OUT output is high
011 = Either Comparator C1OUT or C2OUT is high
100 = VIL on FLT0 pin
101 = VIL on FLT0 pin or Comparator C1OUT output is high
110 = VIL on FLT0 pin or Comparator C2OUT output is high
111 = VIL on FLT0 pin or Comparator C1OUT or Comparator C2OUT is high

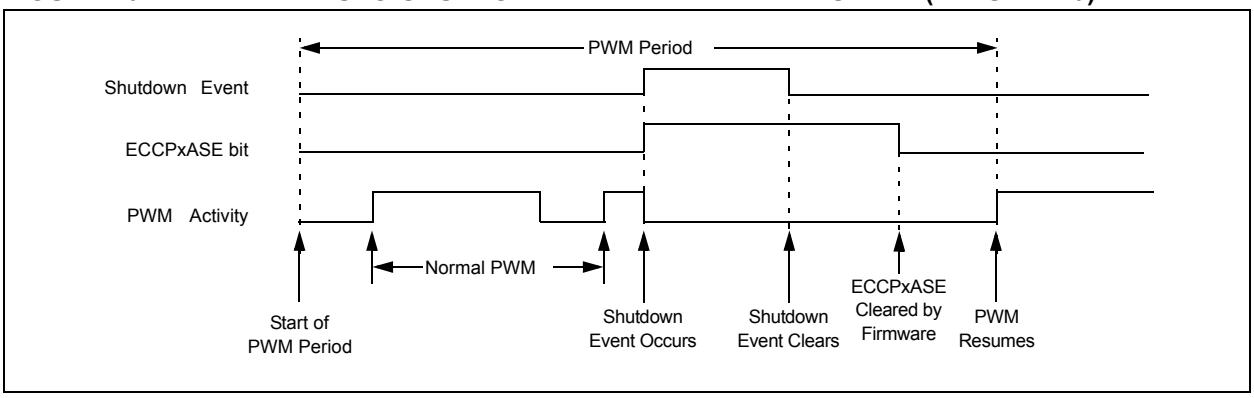
bit 3-2 **PSSxAC<1:0>:** PxA and PxC Pins Shutdown State Control bits
00 = Drive pins, PxA and PxC, to '0'
01 = Drive pins, PxA and PxC, to '1'
1x = PxA and PxC pins tri-state

bit 1-0 **PSSxBD<1:0>:** Pins PxB and PxD Shutdown State Control bits
00 = Drive pins, PxB and PxD, to '0'
01 = Drive pins, PxB and PxD, to '1'
1x = PxB and PxD pins tri-state

**Note 1:** The auto-shutdown condition is a level-based signal, not an edge-based signal. As long as the level is present, the auto-shutdown will persist.
**2:** Writing to the ECCPxASE bit is disabled while an auto-shutdown condition persists.
**3:** Once the auto-shutdown condition has been removed and the PWM restarted (either through firmware or auto-restart), the PWM signal will always restart at the beginning of the next PWM period.


© 2009-2018 Microchip Technology Inc.

**FIGURE 20-12: PWM AUTO-SHUTDOWN WITH FIRMWARE RESTART (PxRSEN = 0)**



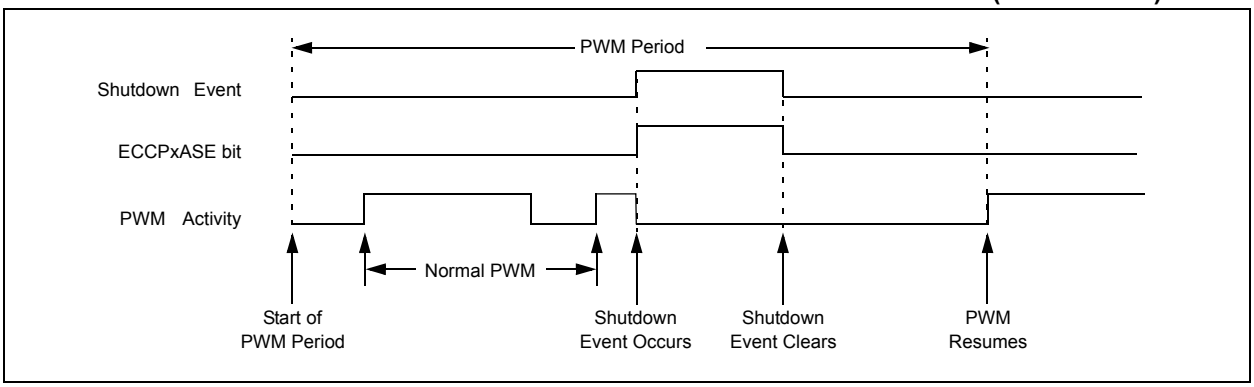

### 20.4.5 AUTO-RESTART MODE

The Enhanced PWM can be configured to automatically restart the PWM signal once the auto-shutdown condition has been removed. Auto-restart is enabled by setting the PxRSEN bit (ECCPxDEL<7>).

If auto-restart is enabled, the ECCPxASE bit will remain set as long as the auto-shutdown condition is active. When the auto-shutdown condition is removed, the ECCPxASE bit will be cleared via hardware and normal operation will resume.

The module will wait until the next PWM period begins, however, before re-enabling the output pin. This behavior allows the auto-shutdown with auto-restart features to be used in applications based on current mode of PWM control.

**FIGURE 20-13: PWM AUTO-SHUTDOWN WITH AUTO-RESTART ENABLED (PxRSEN = 1)**

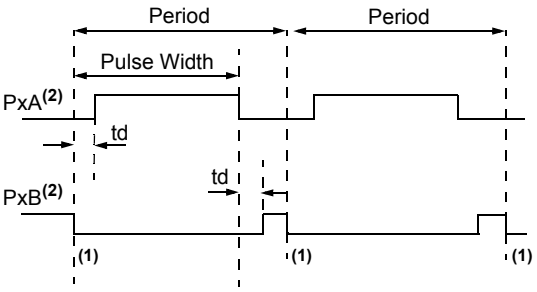


### 20.4.6 PROGRAMMABLE DEAD-BAND DELAY MODE

In half-bridge applications, where all power switches are modulated at the PWM frequency, the power switches normally require more time to turn off than to turn on. If both the upper and lower power switches are switched at the same time (one turned on and the other turned off), both switches may be on for a short period until one switch completely turns off. During this brief interval, a very high current (shoot-through current) will flow through both power switches, shorting the bridge supply. To avoid this potentially destructive shoot-through current from flowing during switching, turning on either of the power switches is normally delayed to allow the other switch to completely turn off.

In Half-Bridge mode, a digitally programmable dead-band delay is available to avoid shoot-through current from destroying the bridge power switches. The delay occurs at the signal transition from the non-active state to the active state. For an illustration, see Figure 20-14. The lower seven bits of the associated ECCPxDEL register (Register 20-4) set the delay period in terms of microcontroller instruction cycles (TCY or 4 TOSC).

**FIGURE 20-14: EXAMPLE OF HALF-BRIDGE PWM OUTPUT**

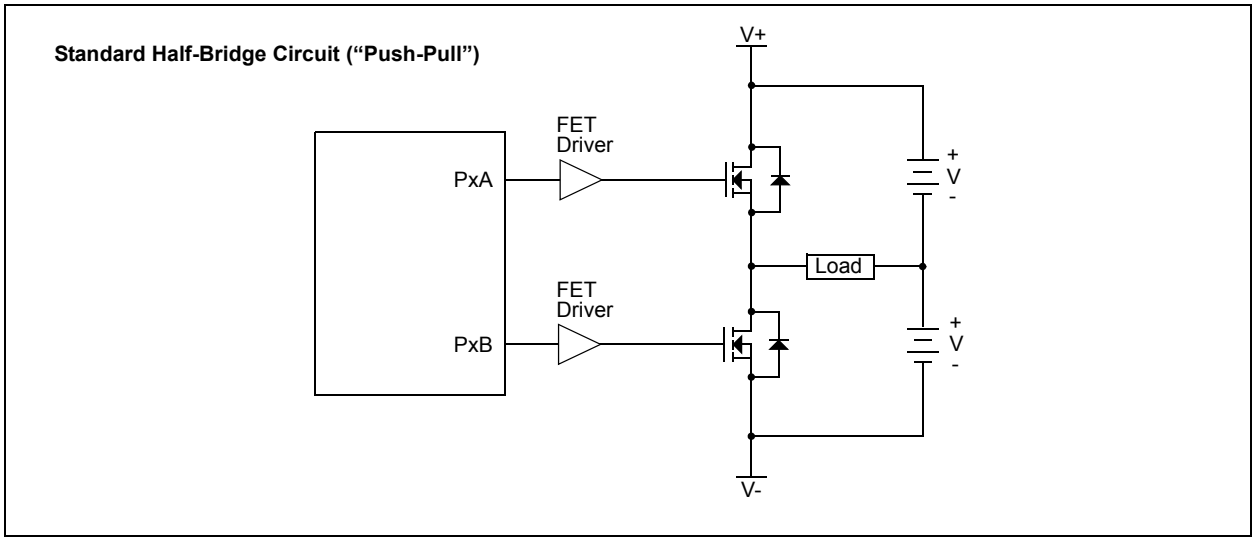



td = Dead-Band Delay

**Note 1:** At this time, the TMR2 register is equal to the PR2 register.

**2:** Output signals are shown as active-high.

**FIGURE 20-15: EXAMPLE OF HALF-BRIDGE APPLICATIONS**





© 2009-2018 Microchip Technology Inc.

**REGISTER 20-4: ECCPxDEL: ENHANCED PWM 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 |
|---|---|---|---|---|---|---|---|
| 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 by software to restart the PWM

bit 6-0 **PxDC<6:0>:** PWM Delay Count bits

PxDCn = Number of FOSC/4 (4 * TOSC) cycles between the scheduled time when a PWM signal **should** transition active and the **actual** time it does transition active.

### 20.4.7 PULSE STEERING MODE

In Single Output mode, pulse steering allows any of the PWM pins to be the modulated signal. Additionally, the same PWM signal can simultaneously be available on multiple pins.

Once the Single Output mode is selected (CCPxM<3:2> = 11 and PxM<1:0> = 00 of the CCPxCON register), the user firmware can bring out the same PWM signal to one, two, three or four output pins by setting the appropriate STR<D:A> bits (PSTRxCON<3:0>), as provided in Table 20-3.

| **Note:** | The associated TRIS bits must be set to output ('0') to enable the pin output driver in order to see the PWM signal on the pin. |
|---|---|

While the PWM Steering mode is active, the CCPxM<1:0> bits (CCPxCON<1:0>) select the PWM output polarity for the Px<D:A> pins.

The PWM auto-shutdown operation also applies to the PWM Steering mode, as described in **Section 20.4.4 "Enhanced PWM Auto-shutdown mode"**. An auto-shutdown event will only affect pins that have PWM outputs enabled.

**REGISTER 20-5: PSTRxCON: PULSE STEERING CONTROL(1)**

| R/W-0 | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-1 |
|---|---|---|---|---|---|---|---|
| CMPL1 | CMPL0 | — | STRSYNC | STRD | STRC | STRB | STRA |
| 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 **CMPL<1:0>:** Complementary Mode Output Assignment Steering Sync bits
00 = See STR<D:A>
01 = PA and PB are selected as the complementary output pair
10 = PA and PC are selected as the complementary output pair
11 = PA and PD are selected as the complementary output pair

bit 5 **Unimplemented:** Read as '0'

bit 4 **STRSYNC:** Steering Sync bit
1 = Output steering update occurs on the next PWM period
0 = Output steering update occurs at the beginning of the instruction cycle boundary

bit 3 **STRD:** Steering Enable bit D
1 = PxD pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxD pin is assigned to the port pin

bit 2 **STRC:** Steering Enable bit C
1 = PxC pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxC pin is assigned to the port pin

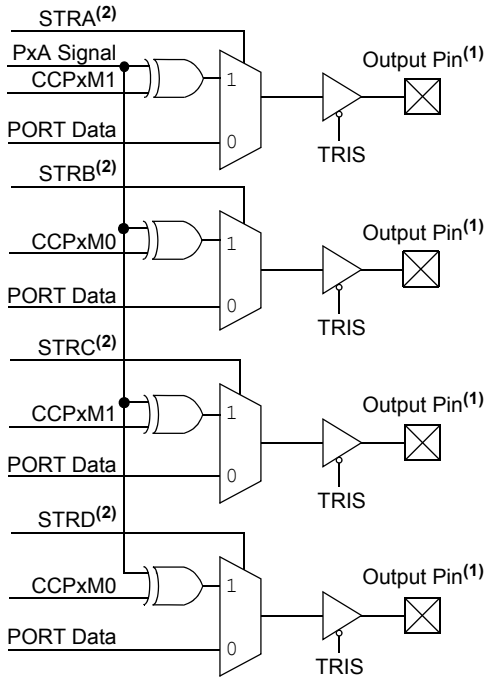
bit 1 **STRB:** Steering Enable bit B
1 = PxB pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxB pin is assigned to the port pin

bit 0 **STRA:** Steering Enable bit A
1 = PxA pin has the PWM waveform with polarity control from CCPxM<1:0>
0 = PxA pin is assigned to the port pin

**Note 1:** The PWM Steering mode is available only when the CCPxCON register bits, CCPxM<3:2> = 11 and PxM<1:0> = 00.


© 2009-2018 Microchip Technology Inc.

**FIGURE 20-16: SIMPLIFIED STEERING BLOCK DIAGRAM**




**Note 1:** Port outputs are configured as displayed when the CCPxCON register bits, PxM<1:0> = 00 and CCP1M<3:2> = 11.

**2:** Single PWM output requires setting at least one of the STRx bits.

### 20.4.7.1 Steering Synchronization

The STRSYNC bit of the PSTRxCON register gives the user two choices for when the steering event will happen. When the STRSYNC bit is '0', the steering event will happen at the end of the instruction that writes to the PSTRxCON register. In this case, the output signal at the Px<D:A> pins may be an incomplete PWM waveform. This operation is useful when the user firmware needs to immediately remove a PWM signal from the pin.

When the STRSYNC bit is '1', the effective steering update will happen at the beginning of the next PWM period. In this case, steering on/off the PWM output will always produce a complete PWM waveform.

Figures 20-17 and 20-18 illustrate the timing diagrams of the PWM steering depending on the STRSYNC setting.

**FIGURE 20-17: EXAMPLE OF STEERING EVENT AT END OF INSTRUCTION (STRSYNC = 0)**

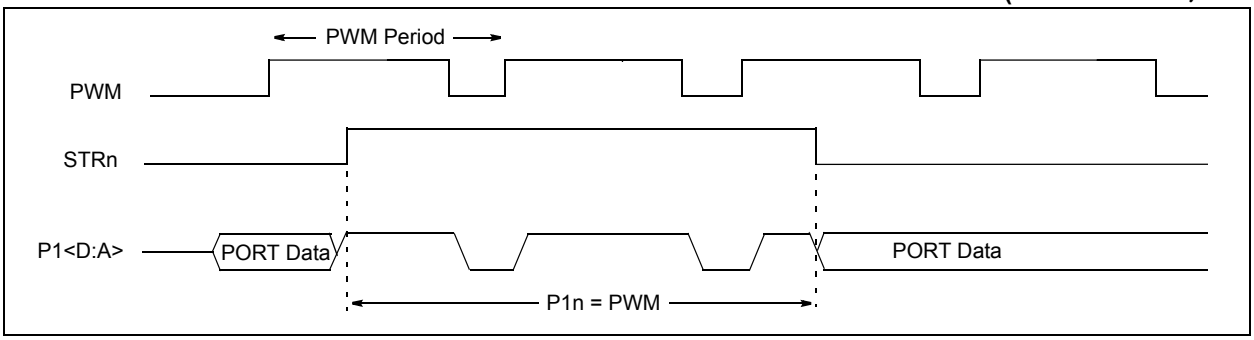
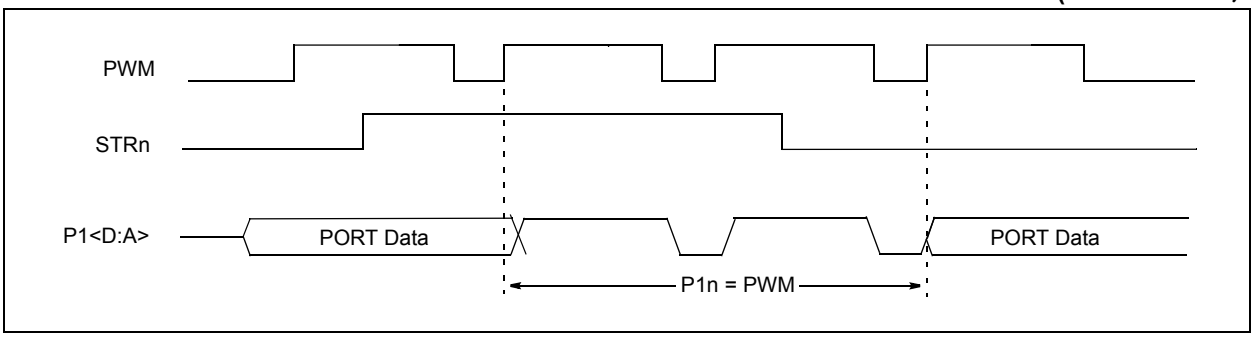



**FIGURE 20-18: EXAMPLE OF STEERING EVENT AT BEGINNING OF INSTRUCTION (STRSYNC = 1)**



### 20.4.8 OPERATION IN POWER-MANAGED MODES

In Sleep mode, all clock sources are disabled. Timer2/4/6/8 will not increment and the state of the module will not change. If the ECCPx pin is driving a value, it will continue to drive that value. When the device wakes up, it will continue from this state. If Two-Speed Start-ups are enabled, the initial start-up frequency from HF-INTOSC and the postscaler may not be immediately stable.

In PRI_IDLE mode, the primary clock will continue to clock the ECCPx module without change.

#### 20.4.8.1 Operation with Fail-Safe Clock Monitor (FSCM)

If the Fail-Safe Clock Monitor (FSCM) is enabled, a clock failure will force the device into the power-managed RC_RUN mode and the OSCFIF bit of the PIR2 register will be set. The ECCPx will then be clocked from the internal oscillator clock source, which may have a different clock frequency than the primary clock.

### 20.4.9 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 ECCP module to reset to a state compatible with previous, non-Enhanced CCP modules used on other PIC18 and PIC16 devices.

**TABLE 20-4: REGISTERS ASSOCIATED WITH ECCP1/2/3 MODULE AND TIMER1/2/3/4/6/8/10/12**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| RCON | IPEN | SBOREN | $\overline{CM}$ | $\overline{RI}$ | $\overline{TO}$ | $\overline{PD}$ | $\overline{POR}$ | $\overline{BOR}$ |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIR4 | CCP10IF(1) | CCP9IF(1) | CCP8IF | CCP7IF | CCP6IF | CCP5IF | CCP4IF | CCP3IF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| PIE4 | CCP10IE(1) | CCP9IE(1) | CCP8IE | CCP7IE | CCP6IE | CCP5IE | CCP4IE | CCP3IE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| IPR4 | CCP10IP(1) | CCP9IP(1) | CCP8IP | CCP7IP | CCP6IP | CCP5IP | CCP4IP | CCP3IP |
| TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 |
| TRISH(2) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 |
| TMR1H | Timer1 Register High Byte | | | | | | | |
| TMR1L | Timer1 Register Low Byte | | | | | | | |
| TMR2 | Timer2 Register | | | | | | | |
| TMR3H | Timer3 Register High Byte | | | | | | | |
| TMR3L | Timer3 Register Low Byte | | | | | | | |
| TMR4 | Timer4 Register | | | | | | | |
| TMR6 | Timer6 Register | | | | | | | |
| TMR8 | Timer8 Register | | | | | | | |
| TMR10(1) | TMR10 Register | | | | | | | |
| TMR12(1) | TMR10 Register | | | | | | | |
| PR2 | Timer2 Period Register | | | | | | | |
| PR4 | Timer4 Period Register | | | | | | | |
| PR6 | Timer6 Period Register | | | | | | | |
| PR8 | Timer8 Period Register | | | | | | | |
| PR10(1) | Timer10 Period Register | | | | | | | |
| PR12(1) | Timer12 Period Register | | | | | | | |

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18F65K22 and PIC18F85K22).
**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.


© 2009-2018 Microchip Technology Inc.

**TABLE 20-4: REGISTERS ASSOCIATED WITH ECCP1/2/3 MODULE AND TIMER1/2/3/4/6/8/10/12 (CONTINUED)**

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| T1CON | TMR1CS1 | TMR1CS0 | T1CKPS1 | T1CKPS0 | SOSCEN | $\overline{\text{T1SYNC}}$ | RD16 | TMR1ON |
| T2CON | — | T2OUTPS3 | T2OUTPS2 | T2OUTPS1 | T2OUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| T3CON | TMR3CS1 | TMR3CS0 | T3CKPS1 | T3CKPS0 | SOSCEN | $\overline{\text{T3SYNC}}$ | RD16 | TMR3ON |
| T4CON | — | T4OUTPS3 | T4OUTPS2 | T4OUTPS1 | T4OUTPS0 | TMR4ON | T4CKPS1 | T4CKPS0 |
| T6CON | — | T6OUTPS3 | T6OUTPS2 | T6OUTPS1 | T6OUTPS0 | TMR6ON | T6CKPS1 | T6CKPS0 |
| T8CON | — | T8OUTPS3 | T8OUTPS2 | T8OUTPS1 | T8OUTPS0 | TMR8ON | T8CKPS1 | T8CKPS0 |
| T10CON(1) | — | T10OUTPS3 | T10OUTPS2 | T10OUTPS1 | T10OUTPS0 | TMR10ON | T10CKPS1 | T10CKPS0 |
| T12CON(1) | — | T12OUTPS3 | T12OUTPS2 | T12OUTPS1 | T12OUTPS0 | TMR12ON | T12CKPS1 | T12CKPS0 |
| CCPR1H | Capture/Compare/PWM Register 1 High Byte | | | | | | | |
| CCPR1L | Capture/Compare/PWM Register 1 Low Byte | | | | | | | |
| CCPR2H | Capture/Compare/PWM Register 2 High Byte | | | | | | | |
| CCPR2L | Capture/Compare/PWM Register 2 Low Byte | | | | | | | |
| CCPR3H | Capture/Compare/PWM Register 3 High Byte | | | | | | | |
| CCPR3L | Capture/Compare/PWM Register 3 Low Byte | | | | | | | |
| CCP1CON | P1M1 | P1M0 | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 |
| CCP2CON | P2M1 | P2M0 | DC2B1 | DC2B0 | CCP2M3 | CCP2M2 | CCP2M1 | CCP2M0 |
| CCP3CON | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**Note 1:** Unimplemented on devices with a program memory of 32 Kbytes (PIC18F65K22 and PIC18F85K22).

**2:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.

## 21.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

### 21.1 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 ($I^2C$)
  - Full Master mode
  - Slave mode (with general address call)

The $I^2C$ 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 PIC18F87K22 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.

### 21.2 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 $I^2C$ 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.

**Note:** The SSPxBUF register cannot be used with read-modify-write instructions, such as `BCF`, `COMF`, etc.

To avoid lost data in Master mode, a read of the SSPxBUF must be performed to clear the Buffer Full (BF) detect bit (SSPSTAT<0>) between each transmission.

### 21.3 SPI Mode

The SPI mode allows eight 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:

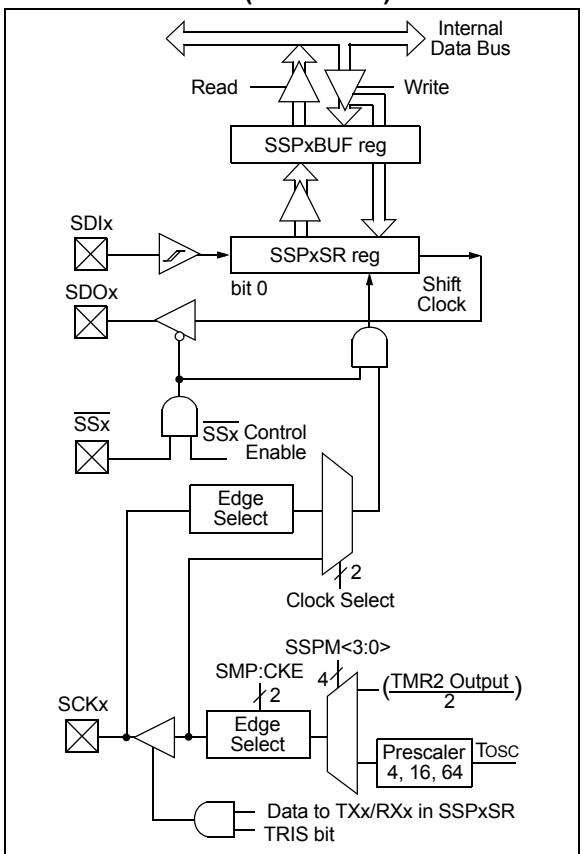
- Serial Data Out (SDOx) – RC5/SDO1 or RD4/PSP4/SDO2
- Serial Data In (SDIx) – RC4/SDI1/SDA1 or RD5/PSP5/SDI2/SDA2
- Serial Clock (SCKx) – RC3/SCK1/SCL1 or RD6/PSP6/SCK2/SCL2

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{SSx}$) – RF7/AN5/$\overline{SS1}$ or RD7/$\overline{SS2}$

Figure 21-1 shows the block diagram of the MSSP module when operating in SPI mode.

**FIGURE 21-1: MSSP 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.


© 2009-2018 Microchip Technology Inc.

### 21.3.1 REGISTERS

Each MSSP module has four registers for SPI mode operation. These are:

- MSSPx Control Register 1 (SSPxCON1)
- MSSPx Status Register (SSPxSTAT)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSPx Shift Register (SSPxSR) – Not directly accessible

SSPxCON1 and SSPxSTAT are the control and status registers in SPI mode operation. The SSPxCON1 register is readable and writable. The lower 6 bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

SSPxSR is the shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPxSR and SSPxBUF together create a double-buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

**REGISTER 21-1: 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 the transition from active to Idle clock state
0 = Transmit occurs on the transition from Idle to active clock state

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit
Used in $I^2C$ mode only.

bit 4 **P:** Stop bit
Used in $I^2C$ mode only. This bit is cleared when the MSSPx module is disabled; SSPEN is cleared.

bit 3 **S:** Start bit
Used in $I^2C$ mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{\text{Write}}$ Information bit
Used in $I^2C$ mode only.

bit 1 **UA:** Update Address bit
Used in $I^2C$ 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:** Polarity of clock state is set by the CKP bit (SSPxCON1<4>).

**REGISTER 21-2: 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 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 the clock is a high level
- 0 = Idle state for the clock is a low level

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits(3)
- 1010 = SPI Master mode: clock = FOSC/8
- 0101 = SPI Slave mode: clock = SCKx pin; $\overline{SSx}$ pin control disabled; $\overline{SSx}$ can be used as I/O pin
- 0100 = SPI Slave mode: clock = SCKx pin; $\overline{SSx}$ pin control 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.


© 2009-2018 Microchip Technology Inc.

### 21.3.2 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 Serial Input 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 21-1 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.

### 21.3.3 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 12.1.3 "Open-Drain Outputs"**.

The open-drain output option is controlled by the SSP2OD and SSP1OD bits (ODCON3<1:0>). Setting an SSPxOD bit configures the SDOx and SCKx pins for the corresponding module for open-drain operation.

| **Note:** | To avoid lost data in Master mode, a read of the SSPxBUF must be performed to clear the Buffer Full (BF) detect bit (SSPxSTAT<0>) between each transmission. |
|---|---|

**EXAMPLE 21-1: 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
```

### 21.3.4 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPxCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPxCON registers and then set the SSPEN bit. This configures the SDIx, SDOx, SCKx and $\overline{SS}$x 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
- $\overline{SS}$x must have the TRISF<7> or TRISD<7> bit set

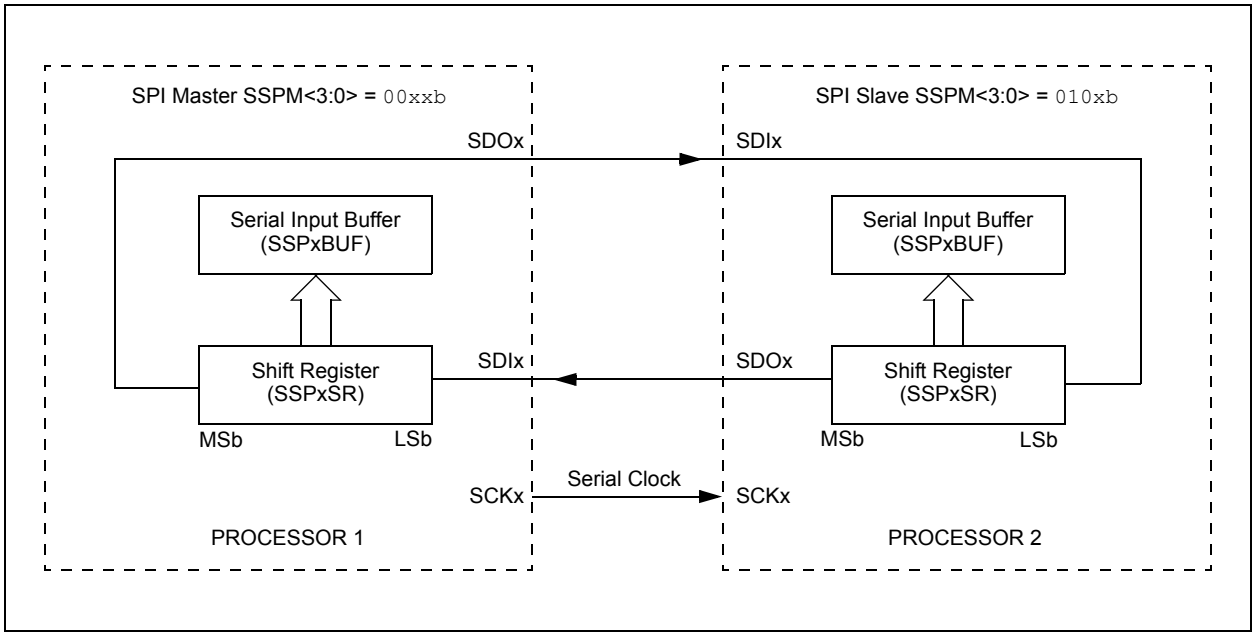
Any serial port function that is not desired may be overridden by programming the corresponding Data Direction (TRIS) register to the opposite value.

### 21.3.5 TYPICAL CONNECTION

Figure 21-2 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 21-2: SPI MASTER/SLAVE CONNECTION**





© 2009-2018 Microchip Technology Inc.

### 21.3.6 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 21-2) 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.

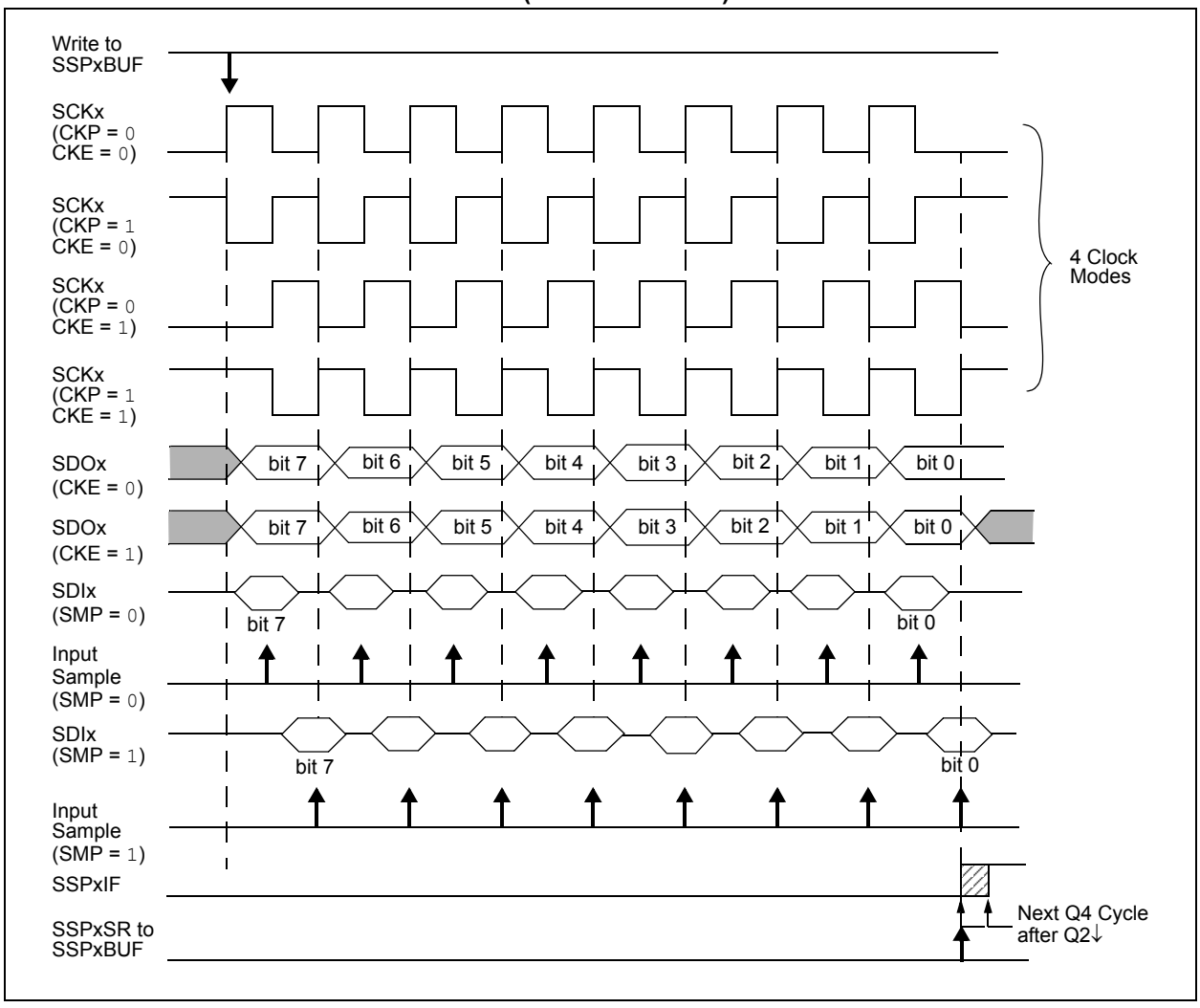
The clock polarity is selected by appropriately programming the CKP bit (SSPxCON1<4>). This, then, would give waveforms for SPI communication, as shown in Figure 21-3, Figure 21-5 and Figure 21-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user-programmable to 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 21-3 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 21-3: SPI MODE WAVEFORM (MASTER MODE)**



### 21.3.7 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.

### 21.3.8 SLAVE SELECT SYNCHRONIZATION

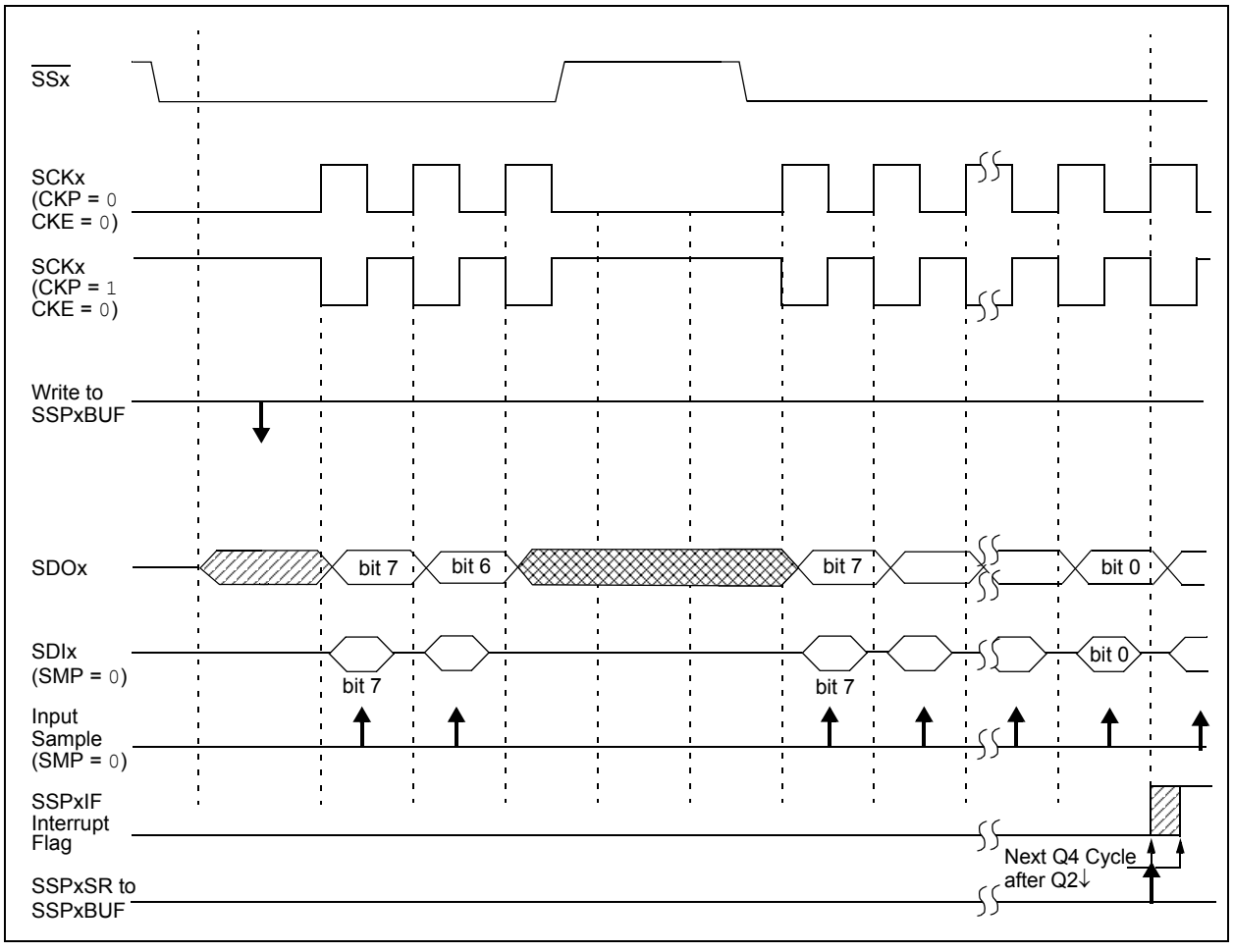
The $\overline{SSx}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with the $\overline{SSx}$ pin control enabled (SSPxCON1<3:0> = 04h). When the $\overline{SSx}$ pin is low, transmission and reception are enabled and the SDOx pin is driven. When the $\overline{SSx}$ pin goes high, 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:** | When the SPI is in Slave mode with the $\overline{SSx}$ pin control enabled (SSPxCON1<3:0> = 0100), the SPI module will reset if the $\overline{SSx}$ pin is set to $V_{DD}$.<br><br>If the SPI is used in Slave mode with CKE set, then the $\overline{SSx}$ 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 $\overline{SSx}$ 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 (SDIx function) since it cannot create a bus conflict.

**FIGURE 21-4: SLAVE SYNCHRONIZATION WAVEFORM**





© 2009-2018 Microchip Technology Inc.

**FIGURE 21-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)**

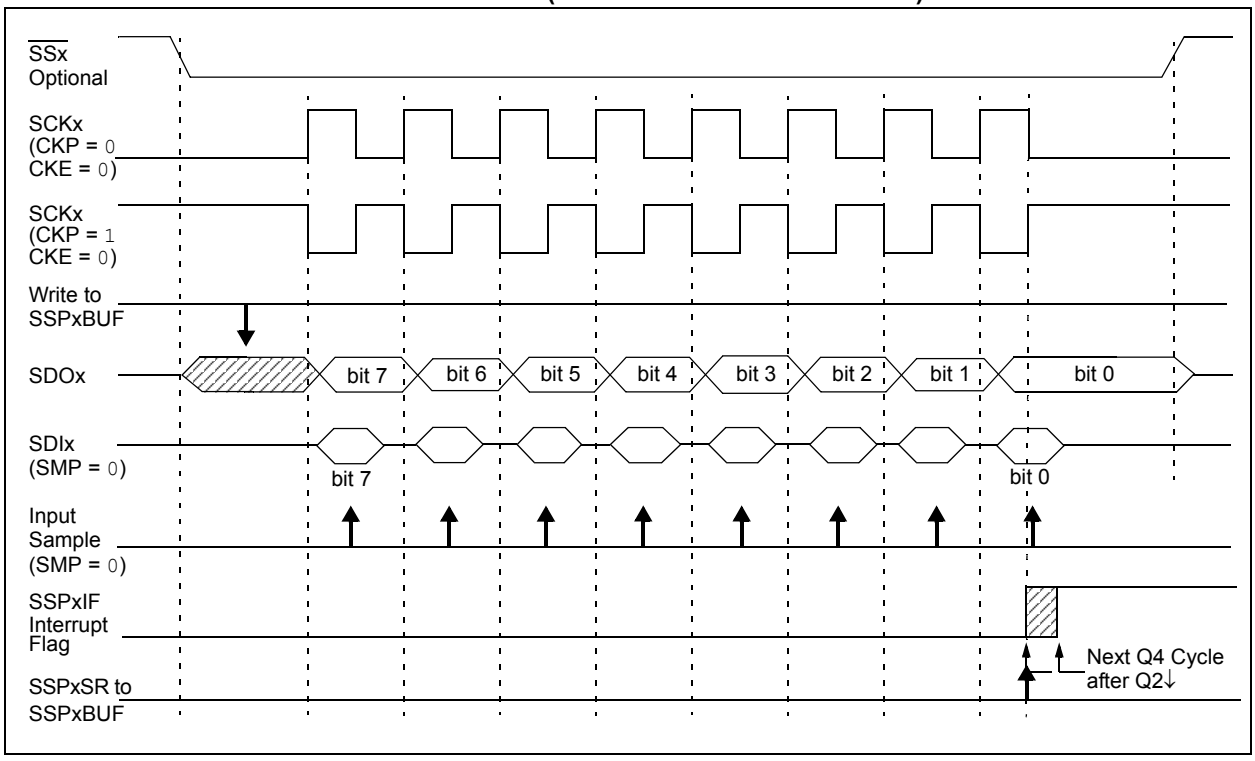
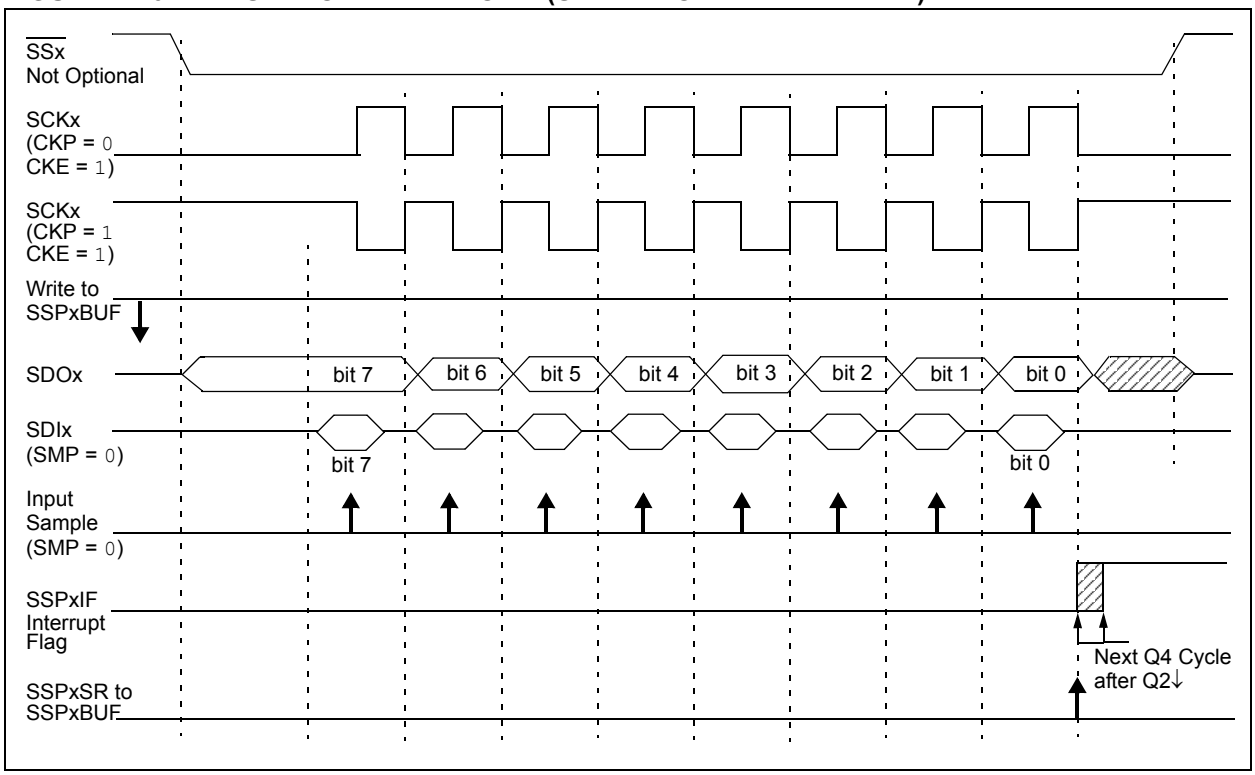



**FIGURE 21-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)**



### 21.3.9 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 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 (SOSC 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 MSSP interrupt flag bit will be set, and if enabled, will wake the device.

### 21.3.10 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

### 21.3.11 BUS MODE COMPATIBILITY

Table 21-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

**TABLE 21-1: 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.

### 21.3.12 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.


© 2009-2018 Microchip Technology Inc.

**TABLE 21-2: REGISTERS ASSOCIATED WITH SPI OPERATION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR2 | OSCFIF | — | SSP2IF | BCL2IF | BCL1IF | HLVDIF | TMR3IF | TMR3GIF |
| PIE2 | OSCFIE | — | SSP2IE | BCL2IE | BCL1IE | HLVDIE | TMR3IE | TMR3GIE |
| IPR2 | OSCFIP | — | SSP2IP | BCL2IP | BCL1IP | HLVDIP | TMR3IP | TMR3GIP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — |
| SSP1BUF | MSSP1 Receive Buffer/Transmit Register | | | | | | | |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |
| SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| SSP1STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| SSP2CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |
| SSP2CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| SSP2STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| SSP2BUF | MSSP2 Receive Buffer/Transmit Register | | | | | | | |
| ODCON1 | SSP1OD | CCP2OD | CCP1OD | — | — | — | — | SSP2OD |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**Legend:** Shaded cells are not used by the MSSP module in SPI mode.

## 21.4 I²C Mode

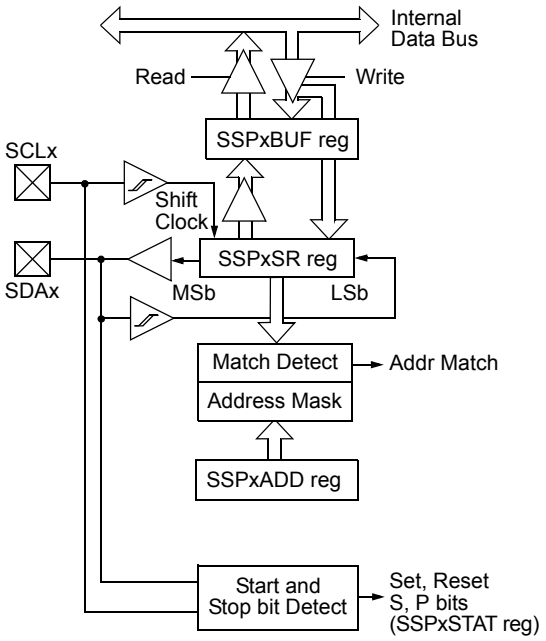
The MSSP module in I²C mode fully implements all master and slave functions (including general call support), and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit 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 21-7: MSSP 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.

### 21.4.1 REGISTERS

The MSSP module has seven 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 MSSP is configured in I²C Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPxADD act as the Baud Rate Generator reload value.

SSPxMSK holds the slave address mask value when the module is configured for 7-Bit Address Masking mode. While it is a separate register, it shares the same SFR address as SSPxADD; it is only accessible when the SSPM<3:0> bits are specifically set to permit access. Additional details are provided in **Section 21.4.3.4 “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.


© 2009-2018 Microchip Technology Inc.

**REGISTER 21-3: 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/$\overline{A}$ | P(1) | S(1) | R/$\overline{W}$(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 = Enable SMBus-specific inputs
0 = Disable SMBus-specific inputs

bit 5 **D/$\overline{A}$:** 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/$\overline{W}$:** 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/$\overline{W}$ 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.

© 2009-2018 Microchip Technology Inc.

**REGISTER 21-4: SSPxCON1: MSSPx CONTROL REGISTER 1 ($I^2C$ 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 $I^2C$ 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 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 = $I^2C$ Slave mode: 10-bit address with Start and Stop bit interrupts enabled
1110 = $I^2C$ Slave mode: 7-bit address with Start and Stop bit interrupts enabled
1011 = $I^2C$ Firmware Controlled Master mode (slave Idle)
1001 = Load SSPMSK register at SSPxADD SFR address(3,4)
1000 = $I^2C$ Master mode: clock = FOSC/(4 * (SSPxADD + 1))
0111 = $I^2C$ Slave mode: 10-bit address
0110 = $I^2C$ 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').


© 2009-2018 Microchip Technology Inc.

**REGISTER 21-5: 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 Stop condition on 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 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 SDAx and SCLx pins. Automatically cleared by hardware.
0 = Start condition is Idle

**Note 1:** 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 21-6: 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 | ACKDT(1) | ACKEN(1) | RCEN(1) | PEN(1) | RSEN(1) | 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 **ACKDT:** Acknowledge Data bit (Master Receive mode only)(1)
1 = Not Acknowledge
0 = Acknowledge

bit 4 **ACKEN:** Acknowledge Sequence Enable bit(1)
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)(1)
1 = Enables Receive mode for $I^2C$
0 = Receive is Idle

bit 2 **PEN:** Stop Condition Enable bit(1)
1 = Initiates Stop condition on SDAx and SCLx pins. Automatically cleared by hardware.
0 = Stop condition is Idle

bit 1 **RSEN:** Repeated Start Condition Enable bit(1)
1 = Initiates Repeated Start condition on SDAx and SCLx pins. Automatically cleared by hardware.
0 = Repeated Start condition is Idle

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 21-7: SSPxMSK: $I^2C$ 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
1 = Masking of corresponding bit of SSPxADD is enabled
0 = Masking of 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 21.4.3.4 "7-Bit Address Masking Mode"** for more details.
**2:** MSK0 is not used as a mask bit in 7-bit addressing.


© 2009-2018 Microchip Technology Inc.

### 21.4.2 OPERATION

The MSSP module functions are enabled by setting the MSSP Enable bit, SSPEN (SSPxCON1<5>).

The SSPxCON1 register allows control of the $I^2C$ operation. Four mode selection bits (SSPxCON1<3:0>) allow one of the following $I^2C$ modes to be selected:

- $I^2C$ Master mode, clock
- $I^2C$ Slave mode (7-bit address)
- $I^2C$ Slave mode (10-bit address)
- $I^2C$ Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- $I^2C$ Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- $I^2C$ Firmware Controlled Master mode, slave is Idle

Selection of any $I^2C$ 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.

### 21.4.3 SLAVE MODE

In Slave mode, the SCLx and SDAx pins must be configured as inputs (TRISC<4:3> set). The MSSP module will override the input state with the output data when required (slave-transmitter).

The $I^2C$ 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 MSSP 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 $I^2C$ specification, as well as the requirement of the MSSP module, are shown in timing Parameter 100 and Parameter 101.

#### 21.4.3.1 Addressing

Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the 8 bits are shifted into the SSPxSR register. All incoming bits are sampled with the rising edge of the clock (SCLx) line. The value of register, SSPxSR<7:1>, is compared to the value of the SSPxADD register. The address is compared on the falling edge of the eighth clock (SCLx) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

1. The SSPxSR register value is loaded into the SSPxBUF register.
2. The Buffer Full bit, BF, is set.
3. An $\overline{ACK}$ pulse is generated.
4. The MSSP Interrupt Flag bit, SSPxIF, is set (and an 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. The R/$\overline{W}$ (SSPxSTAT<2>) bit 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 address match).
2. Update the SSPxADD register with second (low) byte of address (clears bit, UA, and releases the SCLx line).
3. Read the SSPxBUF register (clears bit, BF) and clear flag bit, 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 address. If match releases 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.

### 21.4.3.2 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 $I^2C$ slave behaves the same way whether address masking is used or not. However, when address masking is used, the $I^2C$ 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 PIC18F87K22 family of devices is capable of using two different Address Masking modes in $I^2C$ 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 provides 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.

### 21.4.3.3 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 21-2). 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 $I^2C$ Slave mode (Register 21-6). 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 (SPxADD<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 21-2: 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


© 2009-2018 Microchip Technology Inc.

#### 21.4.3.4 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 that 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 21-3). This mode is the default configuration of the module, which 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 SSPADD 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 21-3: 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<5:1> = 1111 0011

Addresses Acknowledged = ACh, A8h, A4h, A0h

#### 21.4.3.5 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 21.4.4 "Clock Stretching"** for more details.

#### 21.4.3.6 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 21.4.4 "Clock Stretching"** for more details). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPxBUF register which also loads the SSPxSR register. Then, pin SCLx should be enabled by setting bit, CKP (SSPxCON1<4>). The eight data bits are shifted out on the falling edge of the SCLx input. This ensures that the SDAx signal is valid during the SCLx high time (Figure 21-10).

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.


© 2009-2018 Microchip Technology Inc.

**FIGURE 21-8: I²C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)**

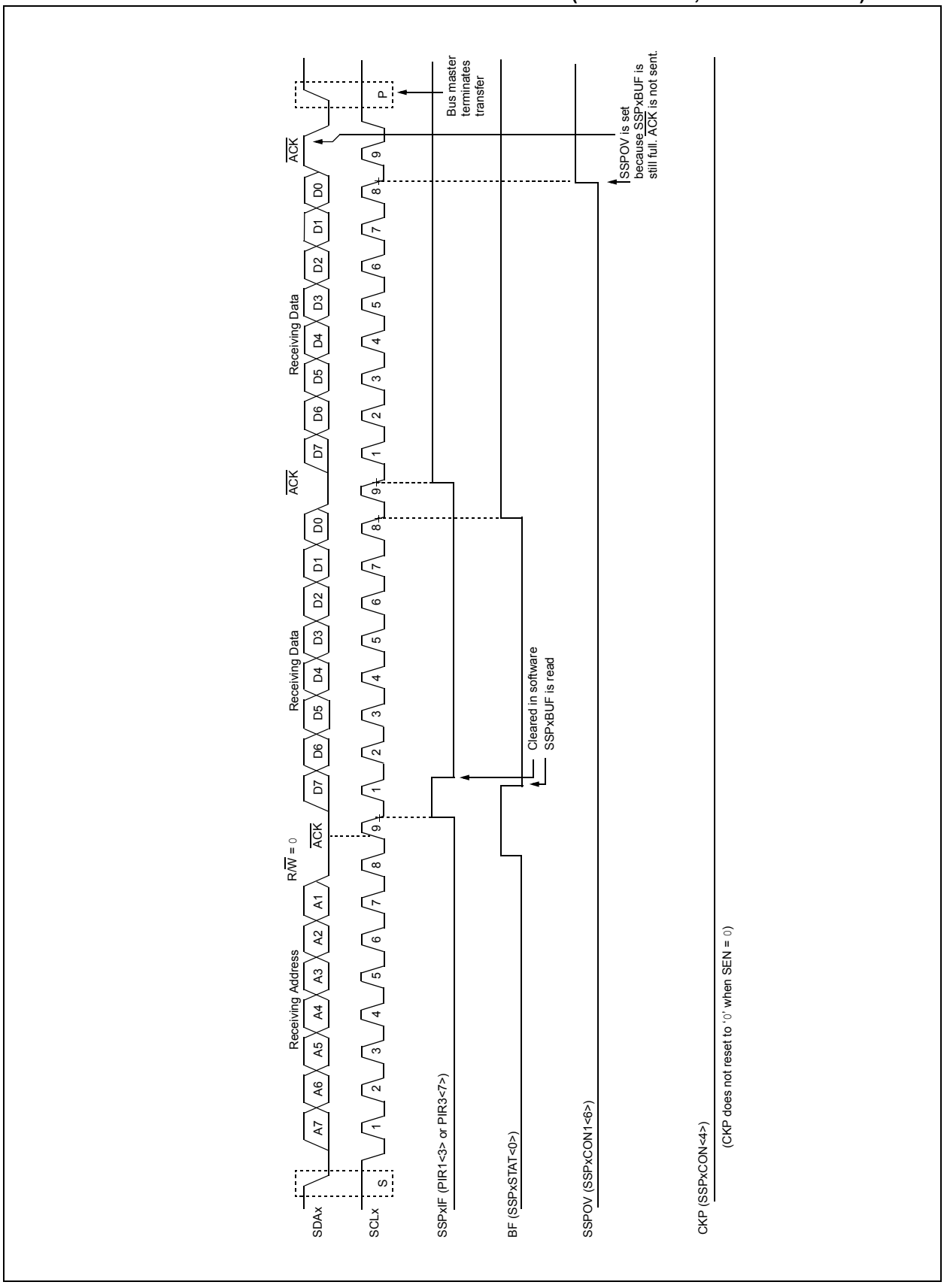
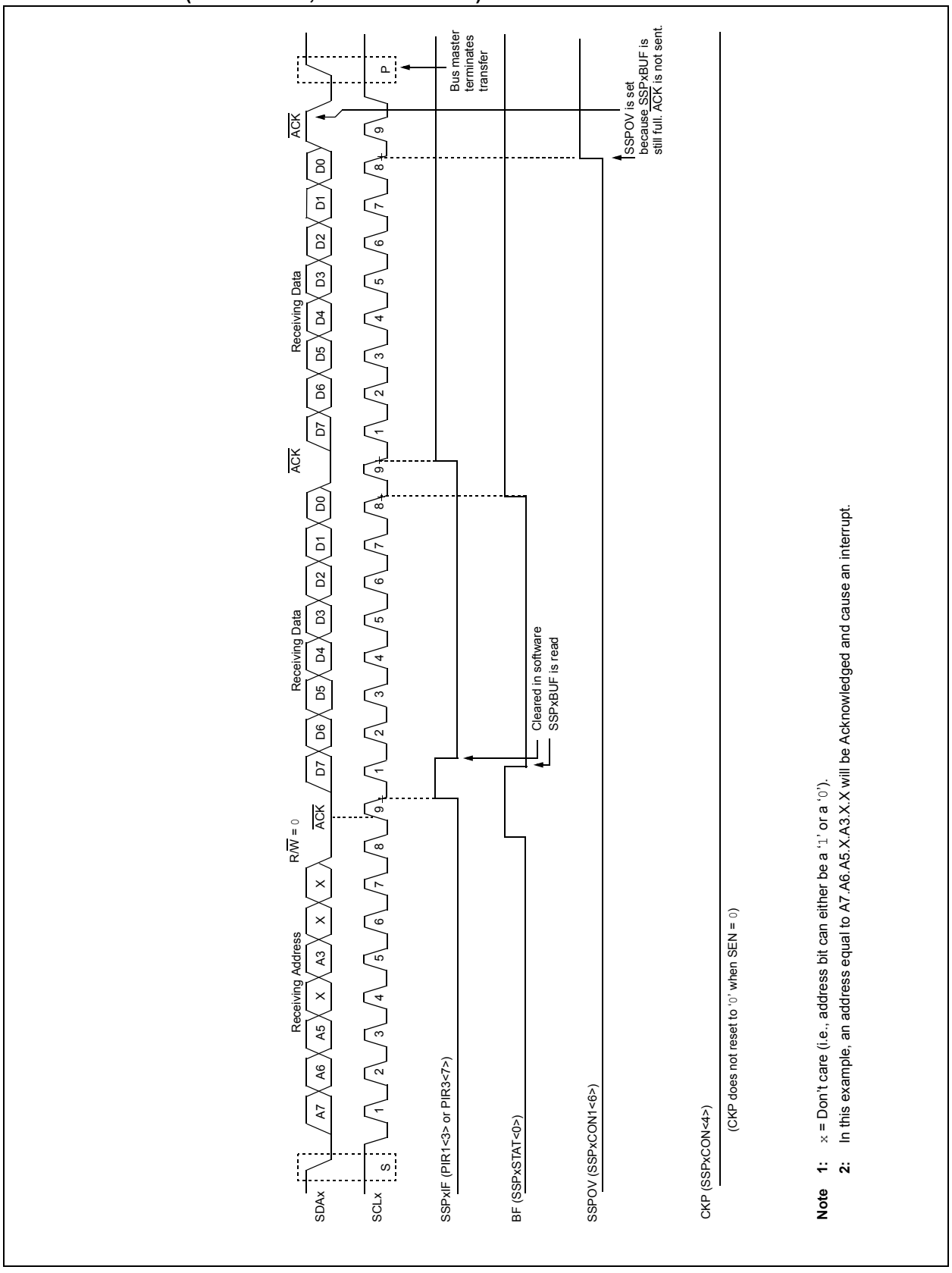


**FIGURE 21-9: I²C SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01011 (RECEPTION, 7-BIT ADDRESS)**




**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.


© 2009-2018 Microchip Technology Inc.

**FIGURE 21-10: I²C SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)**

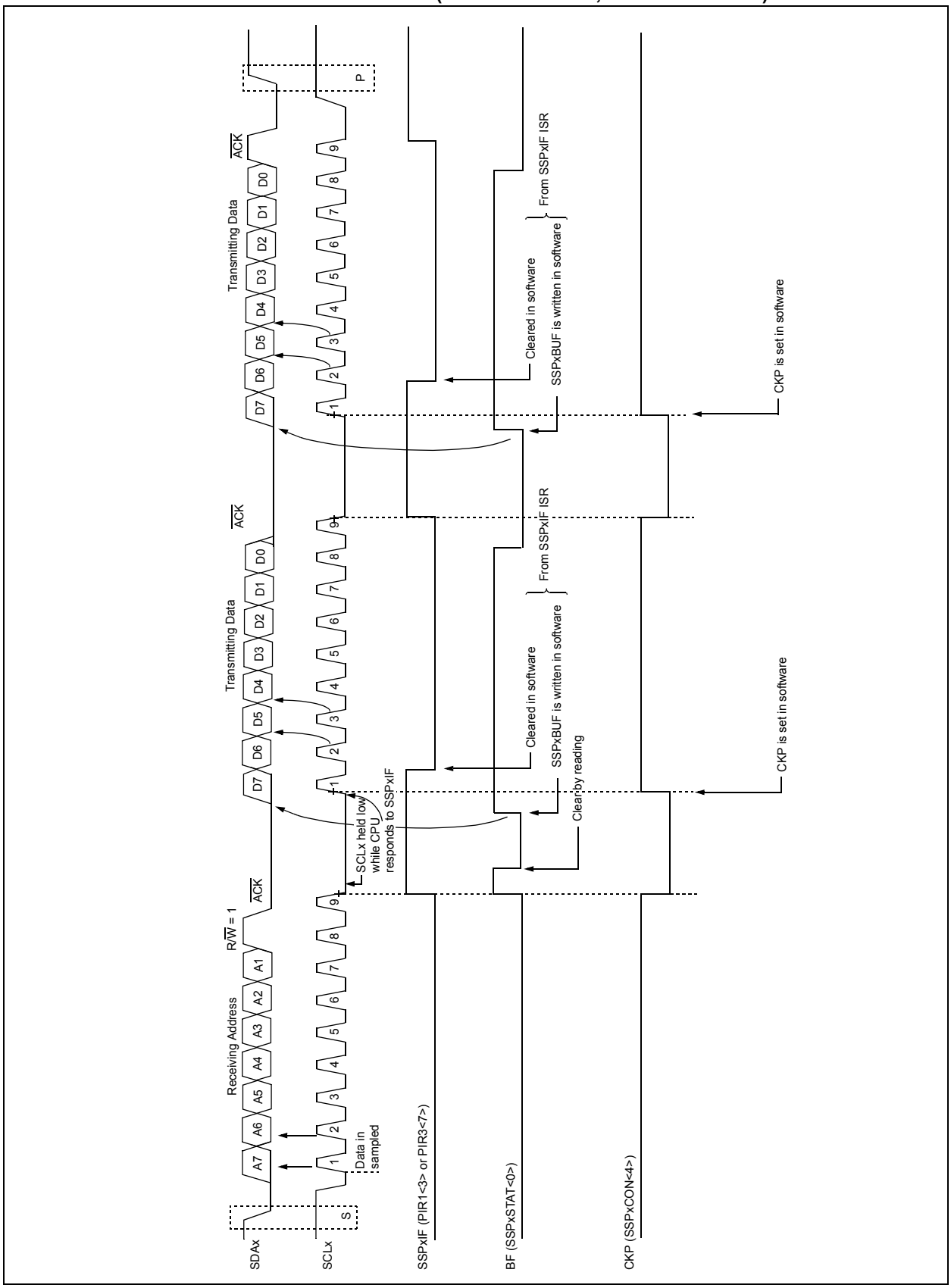
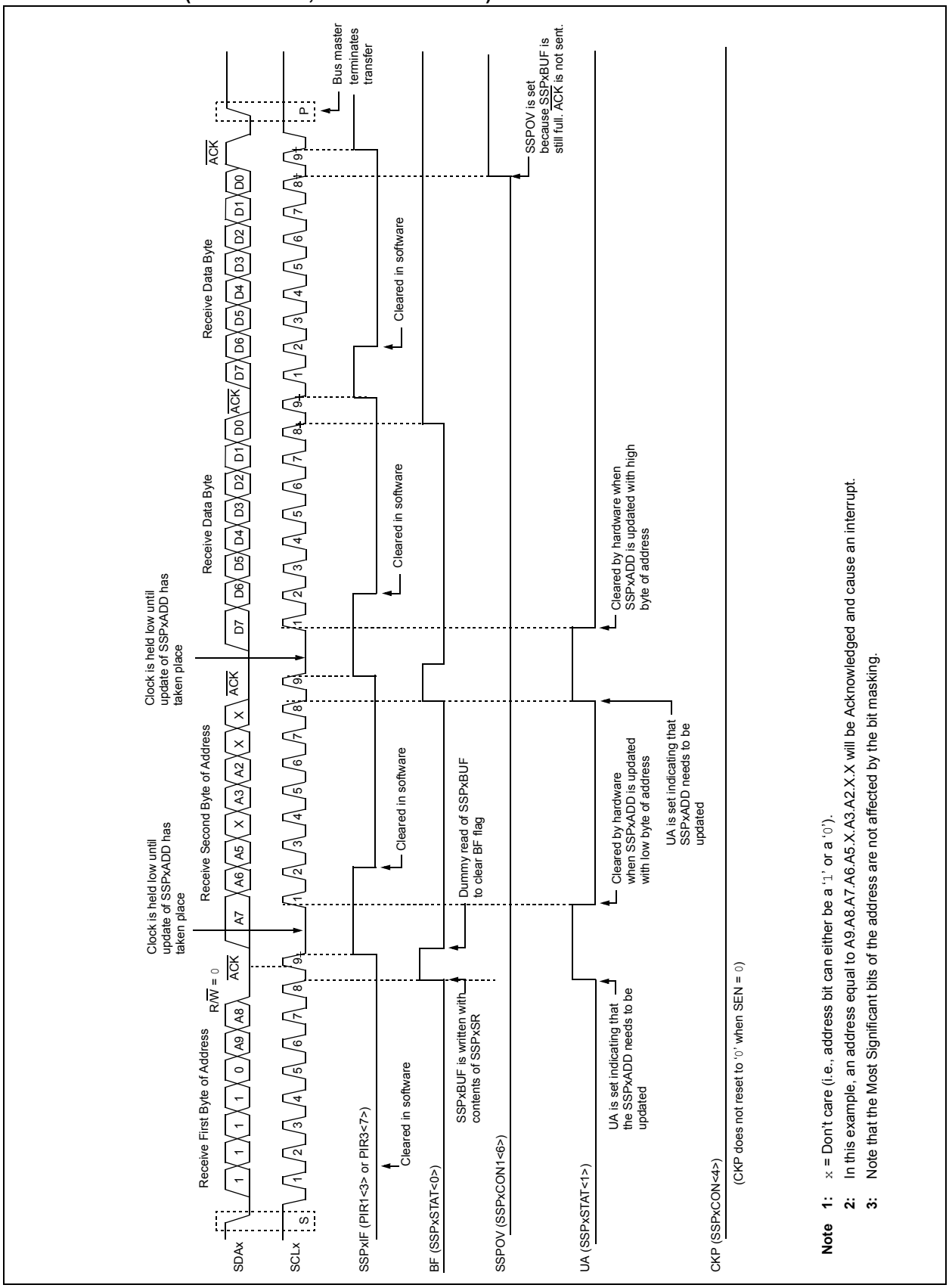


**FIGURE 21-11: I²C SLAVE MODE TIMING WITH SEN = 0 AND ADMSK<5:1> = 01001 (RECEPTION, 10-BIT ADDRESS)**




**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.


© 2009-2018 Microchip Technology Inc.

**FIGURE 21-12: I²C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)**

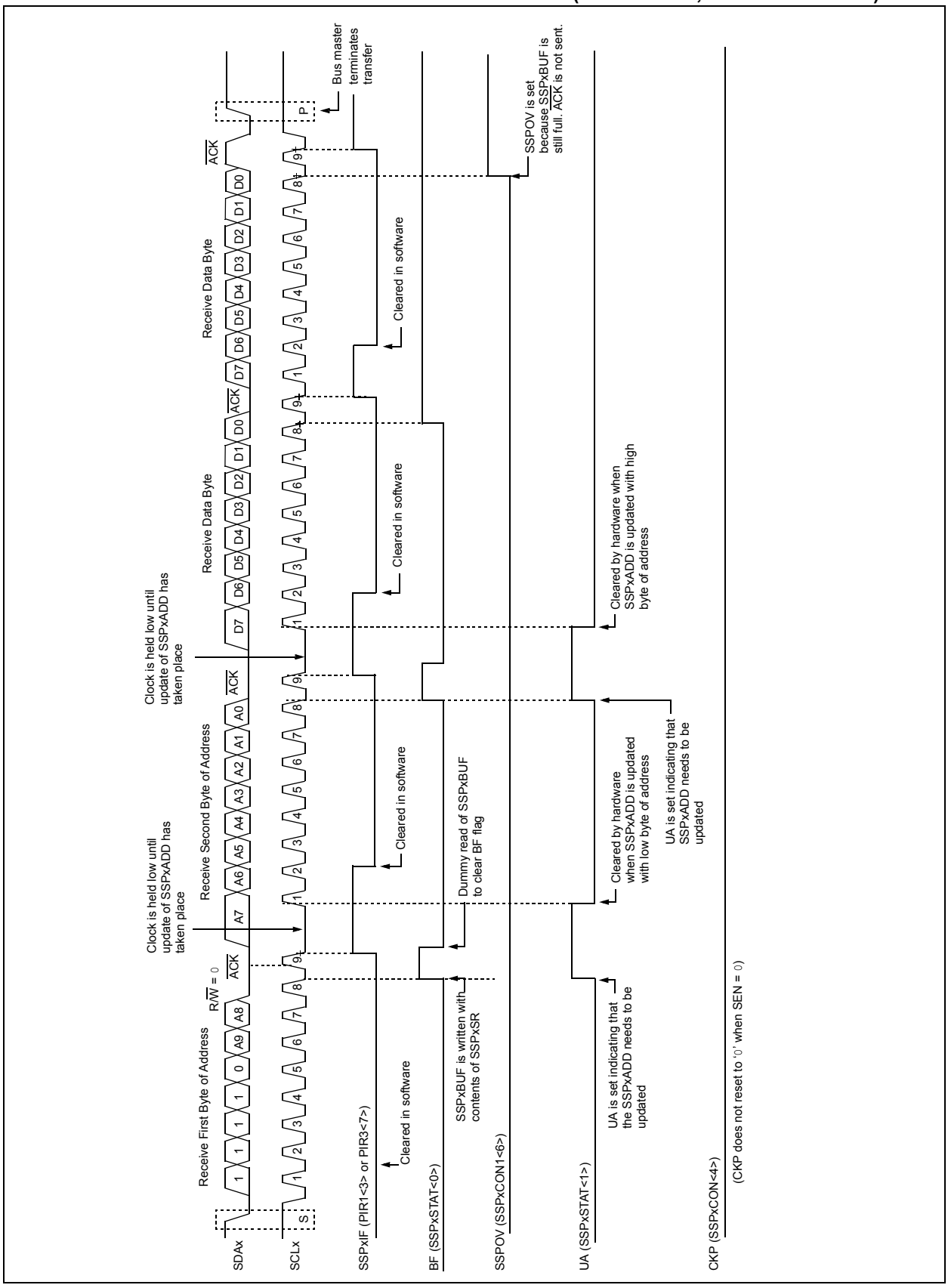
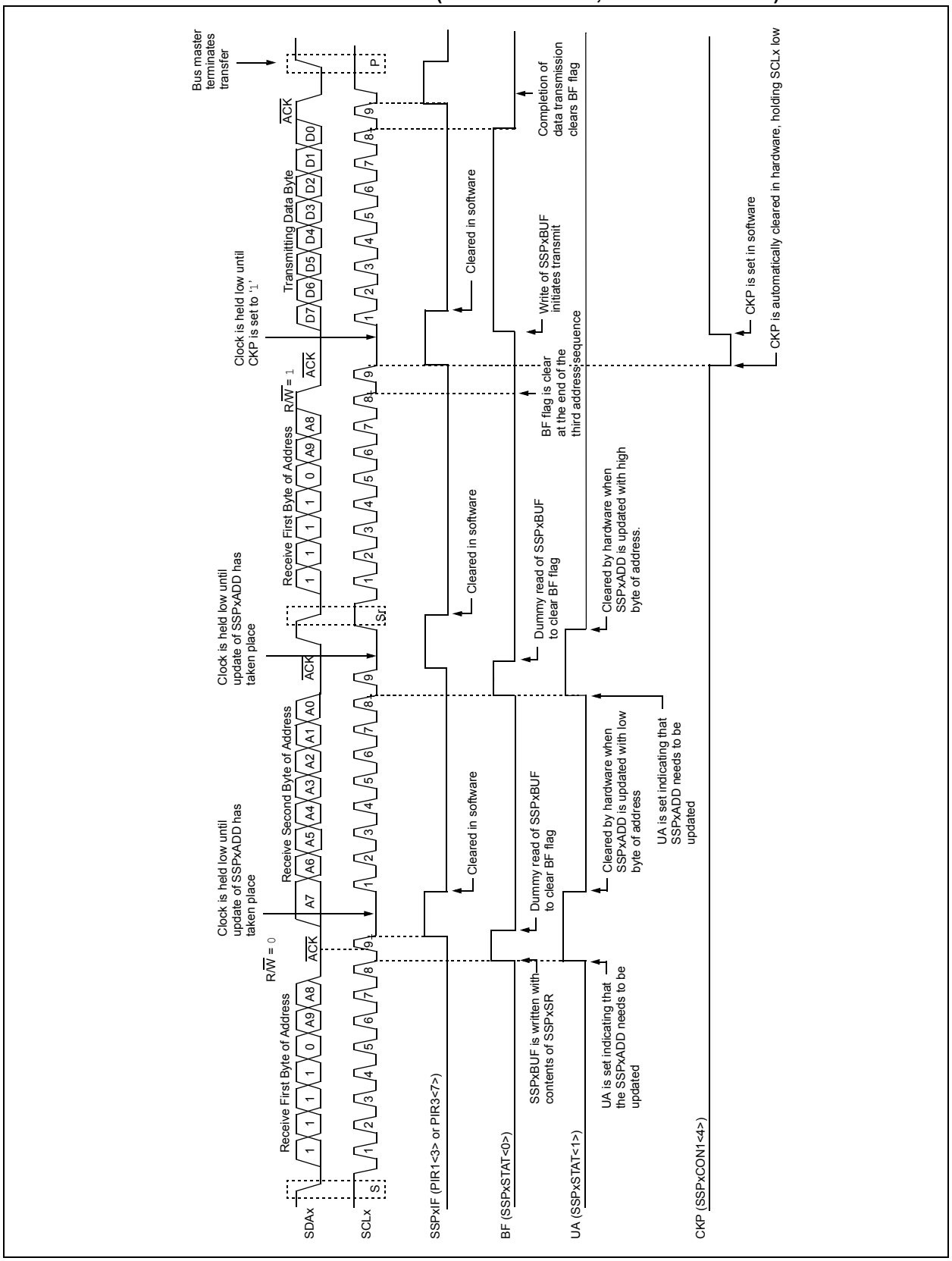


**FIGURE 21-13: I²C SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)**





© 2009-2018 Microchip Technology Inc.

### 21.4.4 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.

#### 21.4.4.1 Clock Stretching for 7-Bit Slave Receive Mode (SEN = 1)

In 7-Bit Slave Receive mode, on the falling edge of the ninth clock at the end of the ACK sequence, if the BF bit is set, the CKP bit in the SSPxCON1 register is automatically cleared, forcing the SCLx output to be held low. The CKP bit being cleared to '0' will assert the SCLx line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and read the contents of the SSPxBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 21-15).

**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.

#### 21.4.4.2 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/$\overline{W}$ 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.

#### 21.4.4.3 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 21-10).

**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.

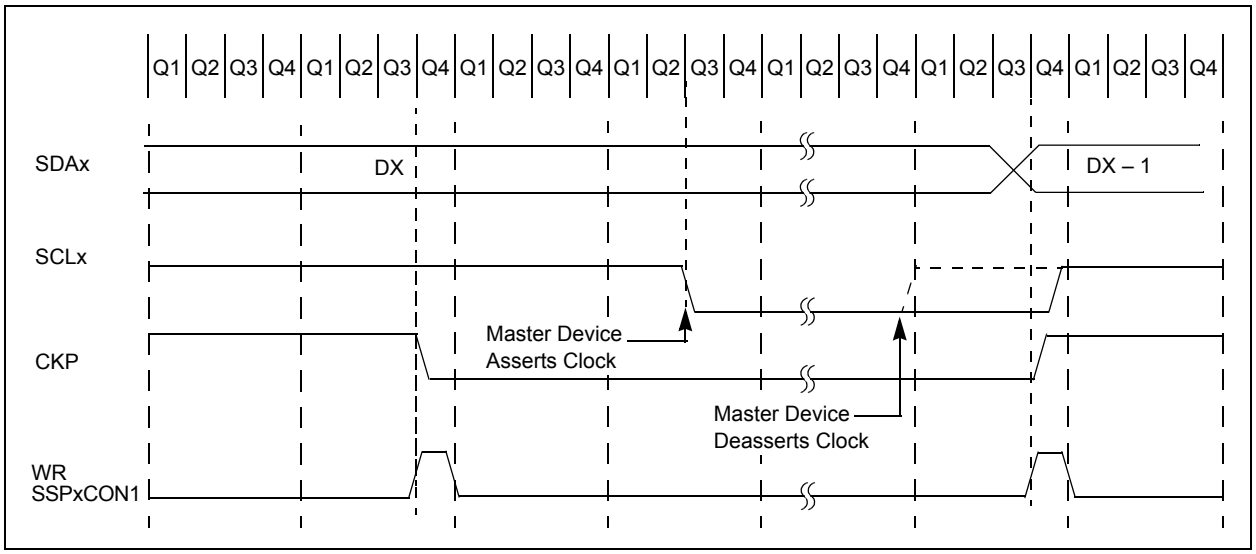
#### 21.4.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

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/$\overline{W}$ 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 21-13).

#### 21.4.4.5 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 21-14).

**FIGURE 21-14: CLOCK SYNCHRONIZATION TIMING**





© 2009-2018 Microchip Technology Inc.

**FIGURE 21-15: I²C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)**

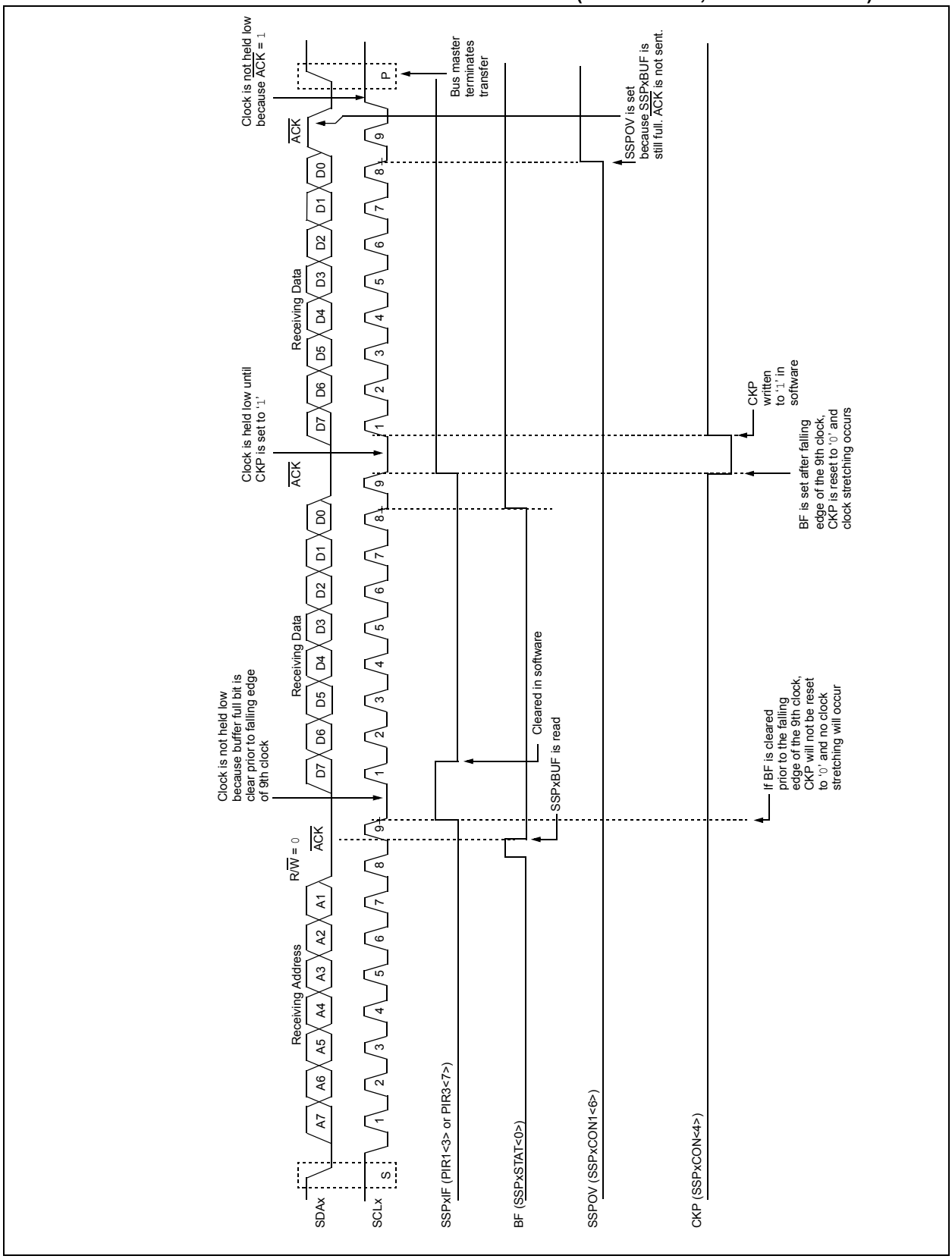
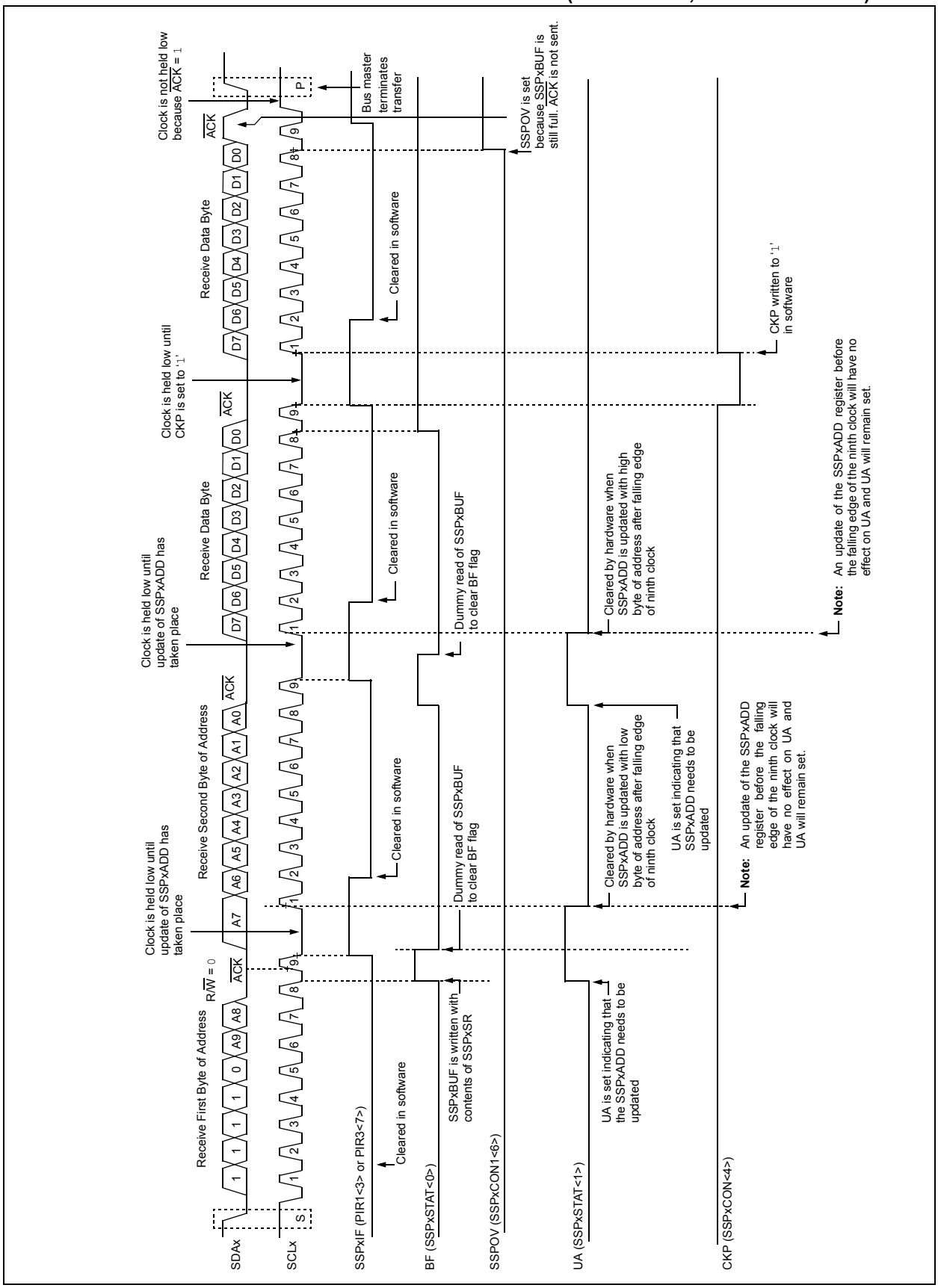


**FIGURE 21-16: I²C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESS)**





© 2009-2018 Microchip Technology Inc.

### 21.4.5 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the $I^2C$ bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master. 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 $I^2C$ protocol. It consists of all '0's with R/$\overline{W}$ = 0.

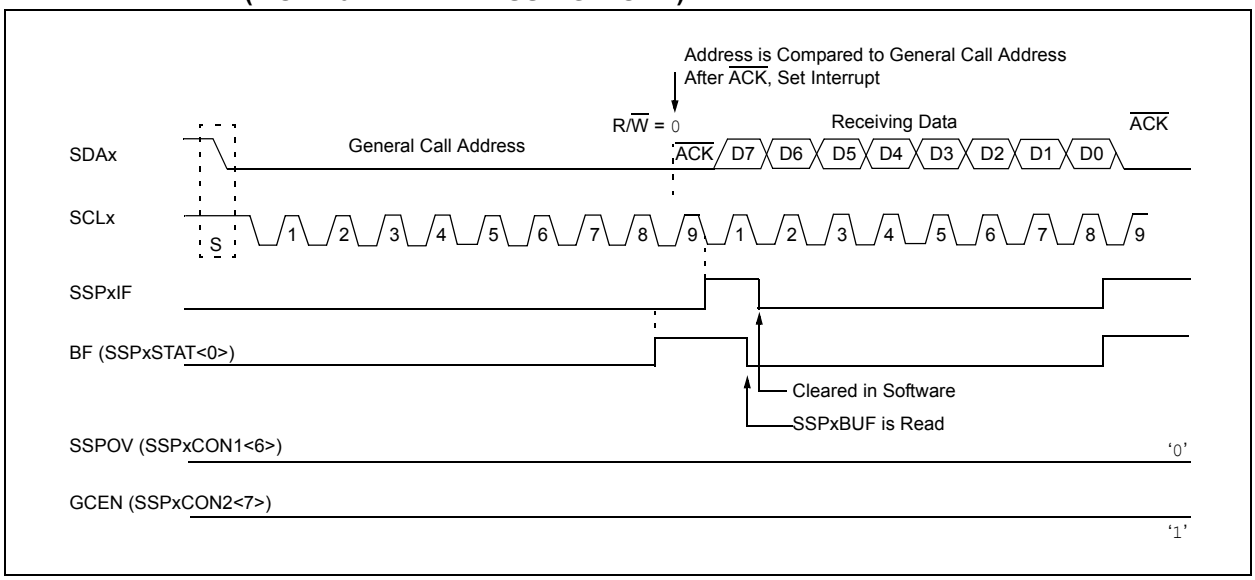
The general call address is recognized when the General Call Enable bit, GCEN, is enabled (SSPxCON2<7> set). Following a Start bit detect, eight bits are shifted into the SSPxSR and the address is compared against the SSPxADD. It is also compared to the general call address and fixed in hardware.

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 21-17).

**FIGURE 21-17: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESSING MODE)**



### 21.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPxCON1, and by setting the SSPEN bit. In Master mode, the SCLx and SDAx lines are manipulated by the MSSP hardware if the TRIS bits are set.

The 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 $I^2C$ 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 $I^2C$ bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

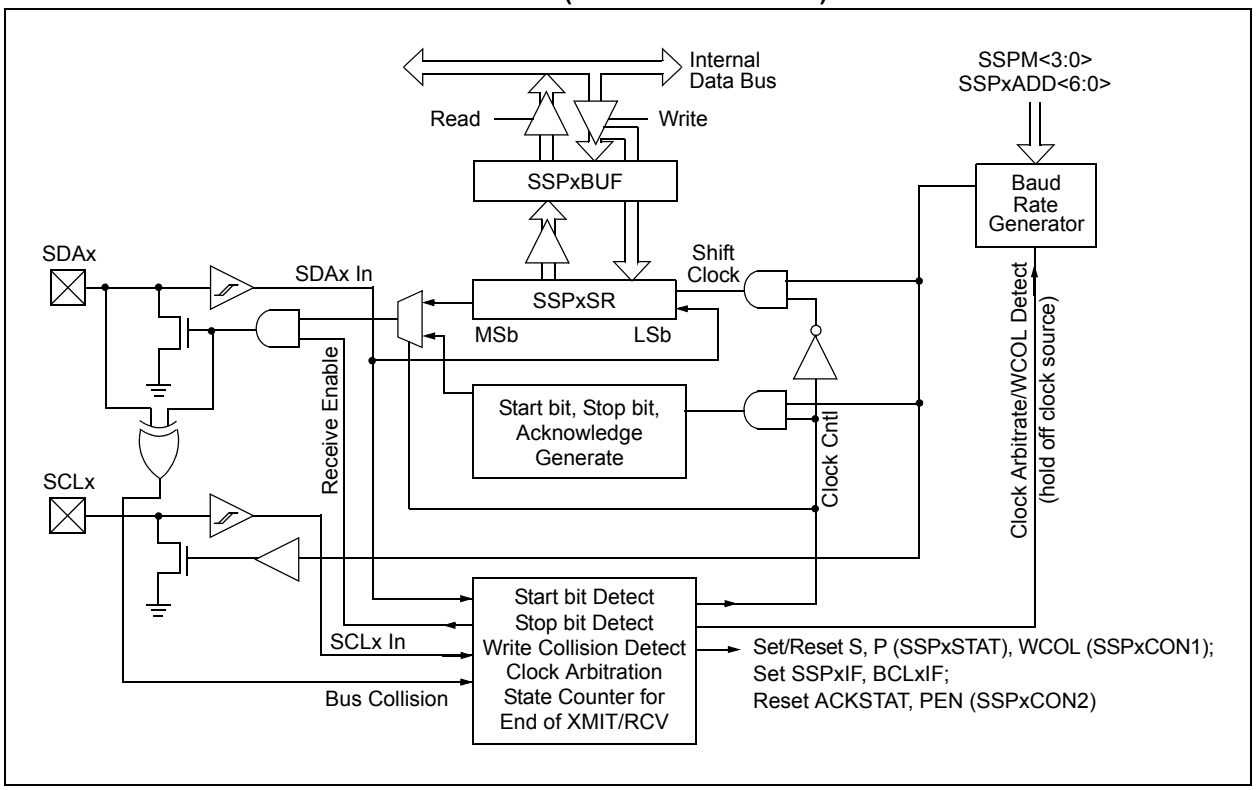
1. Assert a Start condition on SDAx and SCLx.
2. Assert a Repeated Start condition on SDAx and SCLx.
3. Write to the SSPxBUF register, initiating transmission of data/address.
4. Configure the $I^2C$ port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDAx and SCLx.

| **Note:** | The MSSP module, when configured in $I^2C$ Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur. |
|---|---|

The following events will cause the MSSP Interrupt Flag bit, SSPxIF, to be set (and MSSP interrupt, if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmitted
- Repeated Start

**FIGURE 21-18: MSSP BLOCK DIAGRAM ($I^2C$ MASTER MODE)**





© 2009-2018 Microchip Technology Inc.

#### 21.4.6.1 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/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 21.4.7 "Baud Rate"** for more details.

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the Start Enable bit, SEN (SSPxCON2<0>).
2. SSPxIF is set. The MSSP module will wait the required start time before any other operation takes place.
3. The user loads the SSPxBUF with the slave address to transmit.
4. Address is shifted out the SDAx pin until all 8 bits are transmitted.
5. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPxCON2 register (SSPxCON2<6>).
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
7. The user loads the SSPxBUF with eight bits of data.
8. Data is shifted out the SDAx pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPxCON2 register (SSPxCON2<6>).
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
11. The user generates a Stop condition by setting the Stop Enable bit, PEN (SSPxCON2<2>).
12. Interrupt is generated once the Stop condition is complete.

### 21.4.7 BAUD RATE

In $I^2C$ Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPxADD register (Figure 21-19). 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 $I^2C$ Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by $\overline{ACK}$), the internal clock will automatically stop counting and the SCLx pin will remain in its last state.

Table 21-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD. The SSPxADD BRG value of 0x00 is not supported.

#### 21.4.7.1 Baud Rate and Module Interdependence

Because MSSP1 and MSSP2 are independent, they can operate simultaneously in $I^2C$ 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 21-19: BAUD RATE GENERATOR BLOCK DIAGRAM**

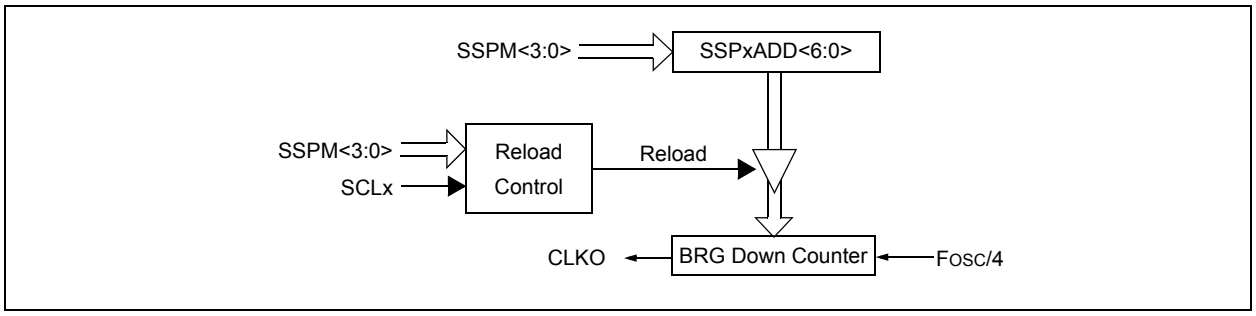



**TABLE 21-3: $I^2C$ CLOCK RATE w/BRG**

| FOSC | FCY | FCY * 2 | BRG Value | FSCL (2 Rollovers of BRG) |
|---|---|---|---|---|
| 40 MHz | 10 MHz | 20 MHz | 18h | 400 kHz |
| 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 |
| 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 |
| 4 MHz | 1 MHz | 2 MHz | 09h | 100 kHz |
| 16 MHz | 4 MHz | 8 MHz | 03h | 1 MHz(1) |

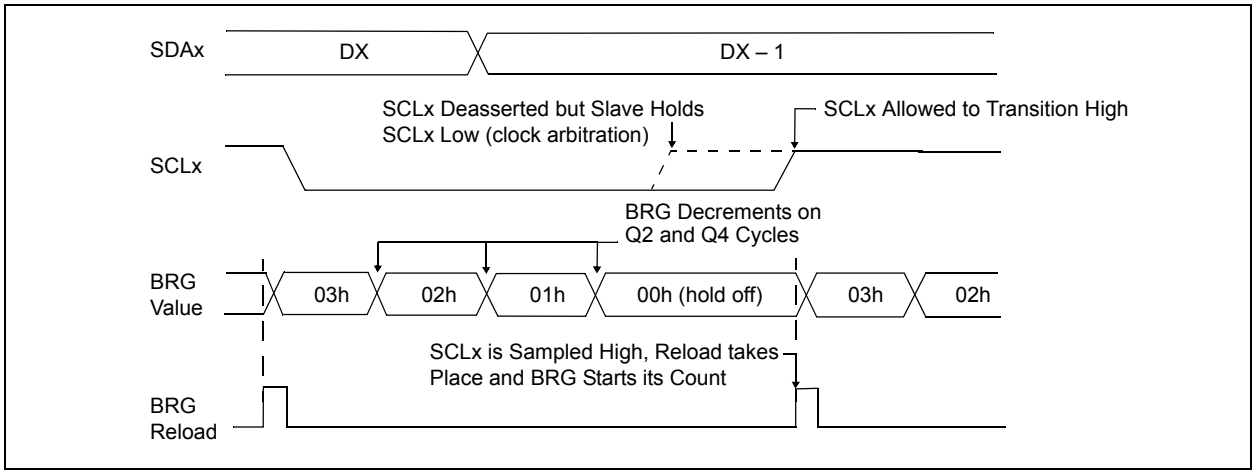
**Note 1:** A minimum of 16 MHz FOSC is required to get 1 MHz $I^2C$.


© 2009-2018 Microchip Technology Inc.

### 21.4.7.2 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts the SCLx pin (SCLx allowed to float high). When the SCLx pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCLx pin is actually sampled high. When the SCLx pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<6:0> and begins counting. This ensures that the SCLx high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 21-20).

**FIGURE 21-20: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION**



### 21.4.8 I²C 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.

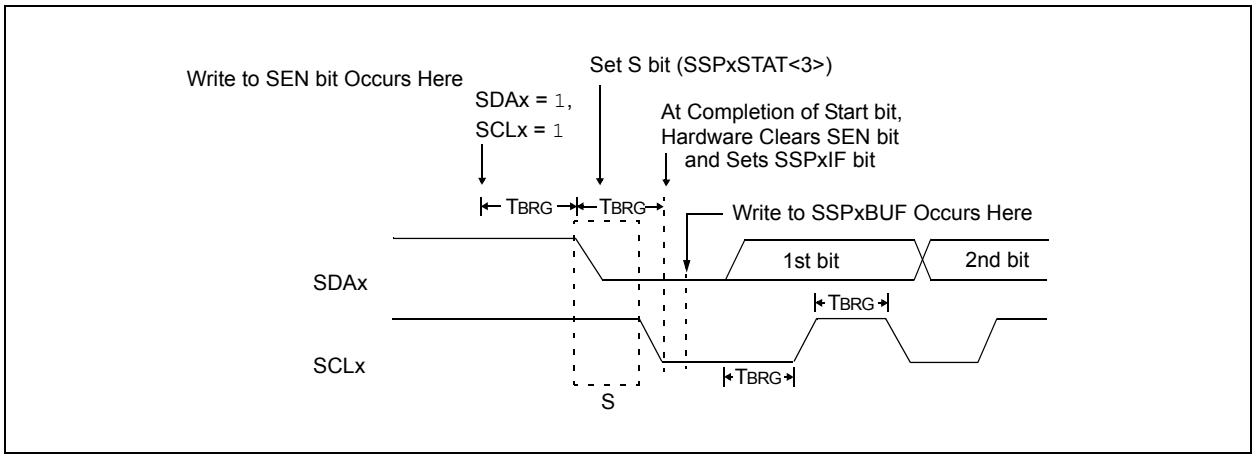
| **Note:** | If, at the beginning of the Start condition, the SDAx and SCLx pins are already sampled low, or if during the Start condition, the SCLx line is sampled low before the SDAx line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLxIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state. |
|---|---|

#### 21.4.8.1 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 21-21: FIRST START BIT TIMING**





© 2009-2018 Microchip Technology Inc.

### 21.4.9 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPxCON2<1>) is programmed high and the I²C logic module is in the Idle state. When the RSEN bit is set, the SCLx pin is asserted low. When the SCLx pin is sampled low, the Baud Rate Generator is loaded with the contents of SSPxADD<5:0> and begins counting. The SDAx pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, and if SDAx is sampled high, the SCLx pin will be deasserted (brought high). When SCLx is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<6:0> and begins counting. SDAx and SCLx must be sampled high for one TBRG. This action is then followed by assertion of the SDAx pin (SDAx = 0) for one TBRG while SCLx is high. Following this, the RSEN bit (SSPxCON2<1>) will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDAx pin held low. As soon as a Start condition is detected on the SDAx and SCLx pins, the S bit (SSPxSTAT<3>) will be set. The SSPxIF bit will not be set until the Baud Rate Generator has timed out.

**Note 1:** If RSEN is programmed while any other event is in progress, it will not take effect.

**2:** A bus collision during the Repeated Start condition occurs if:

- SDAx is sampled low when SCLx goes from low-to-high.
- SCLx goes low before SDAx is asserted low. This may indicate that another master is attempting to transmit a data '1'.

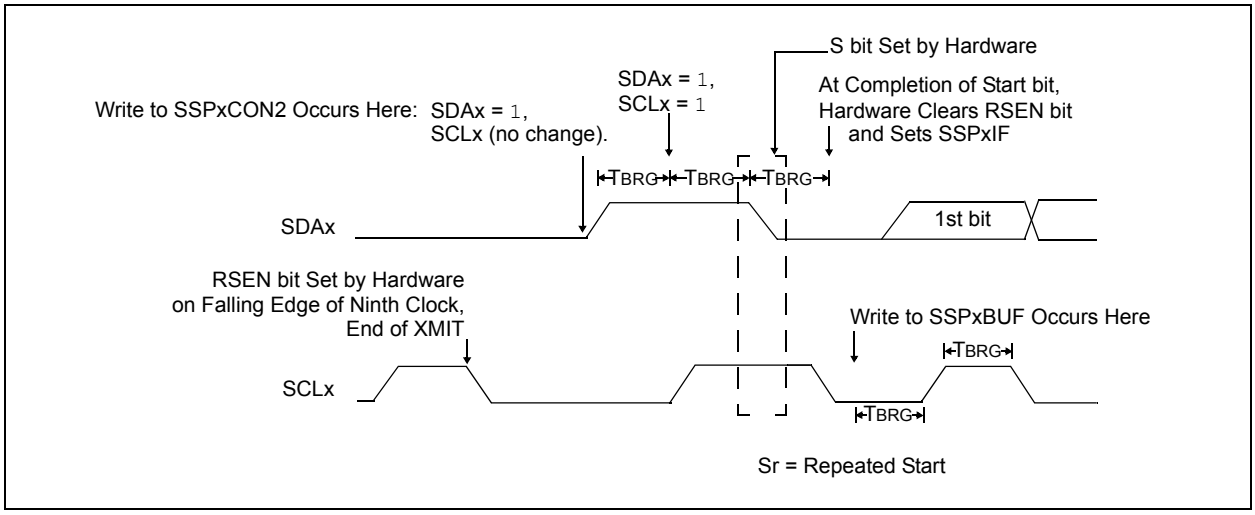
Immediately following the SSPxIF bit getting set, the user may write the SSPxBUF with the 7-bit address in 7-bit mode or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

#### 21.4.9.1 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 21-22: REPEATED START CONDITION WAVEFORM**



### 21.4.10 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 21-23).

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.

#### 21.4.10.1 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.

#### 21.4.10.2 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.

#### 21.4.10.3 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.

### 21.4.11 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the Receive Enable bit, RCEN (SSPxCON2<3>).

| **Note:** | The MSSP 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 MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN (SSPxCON2<4>).

#### 21.4.11.1 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.

#### 21.4.11.2 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.

#### 21.4.11.3 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).


© 2009-2018 Microchip Technology Inc.

**FIGURE 21-23: I²C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)**

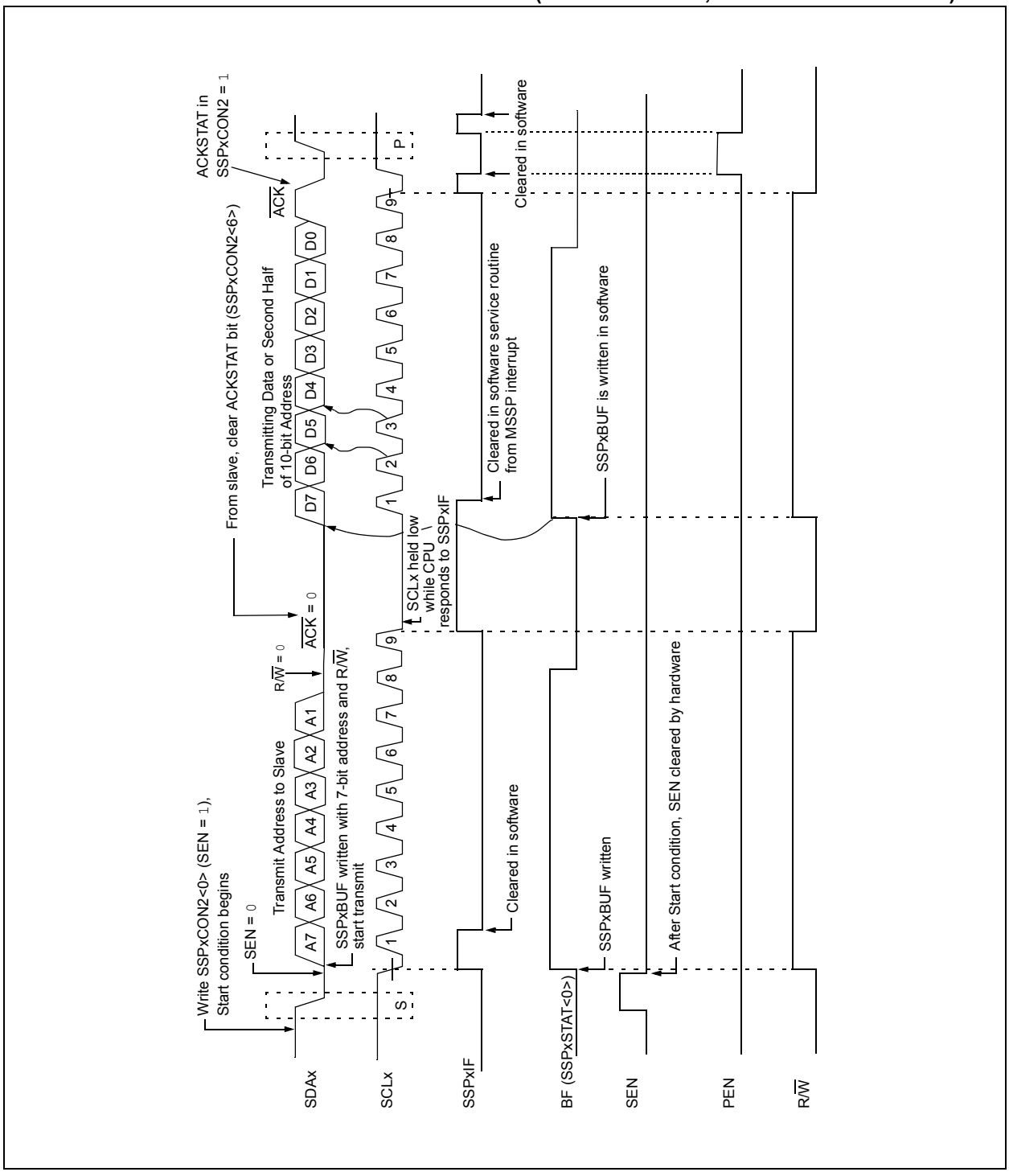
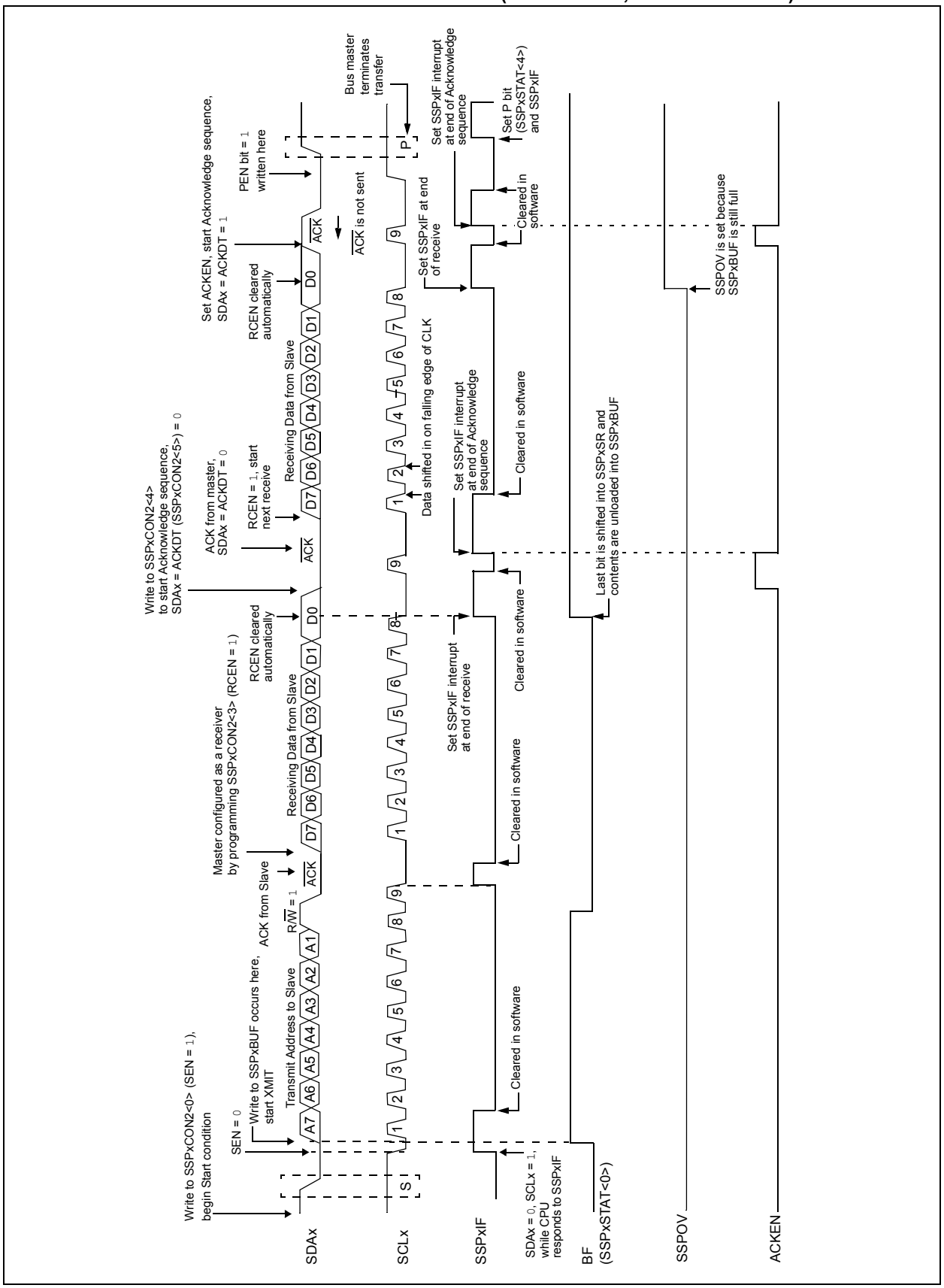


**FIGURE 21-24: I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)**





© 2009-2018 Microchip Technology Inc.

### 21.4.12 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN (SSPxCON2<4>). When this bit is set, the SCLx pin is pulled low and the contents of the Acknowledge data bit are presented on the SDAx pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCLx pin is deasserted (pulled high). When the SCLx pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG; the SCLx pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into an inactive state (Figure 21-25).

#### 21.4.12.1 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).

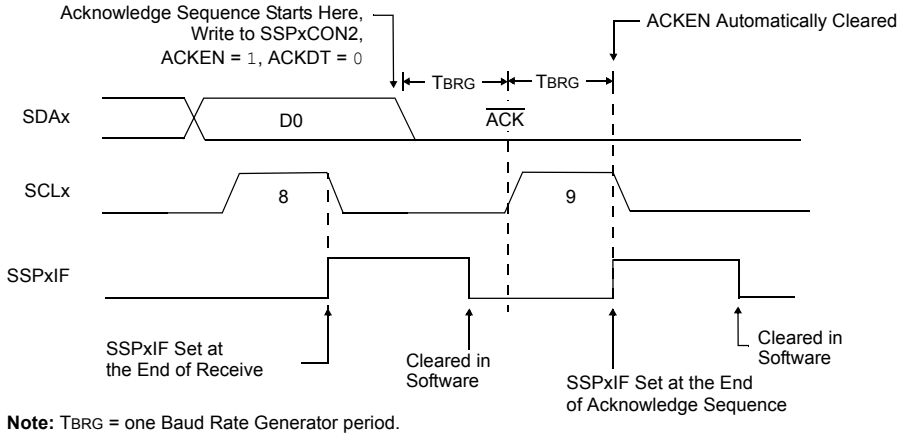
### 21.4.13 STOP CONDITION TIMING

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 (see Figure 21-26).

#### 21.4.13.1 WCOL Status Flag

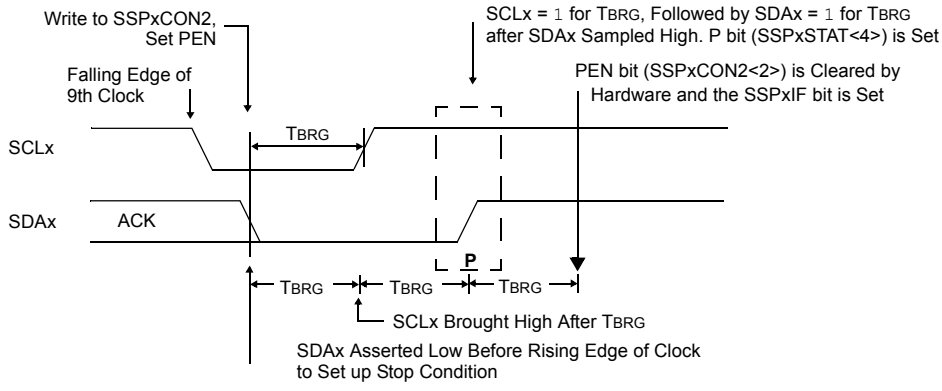
If the user writes the SSPxBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

**FIGURE 21-25: ACKNOWLEDGE SEQUENCE WAVEFORM**




**Note:** TBRG = one Baud Rate Generator period.

**FIGURE 21-26: STOP CONDITION RECEIVE OR TRANSMIT MODE**




**Note:** TBRG = one Baud Rate Generator period.

### 21.4.14 SLEEP OPERATION

While in Sleep mode, the I²C 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).

### 21.4.15 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

### 21.4.16 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 I²C 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

### 21.4.17 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 I²C port to its Idle state (Figure 21-27).

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 I²C bus is free, the user can resume communication by asserting a Start condition.

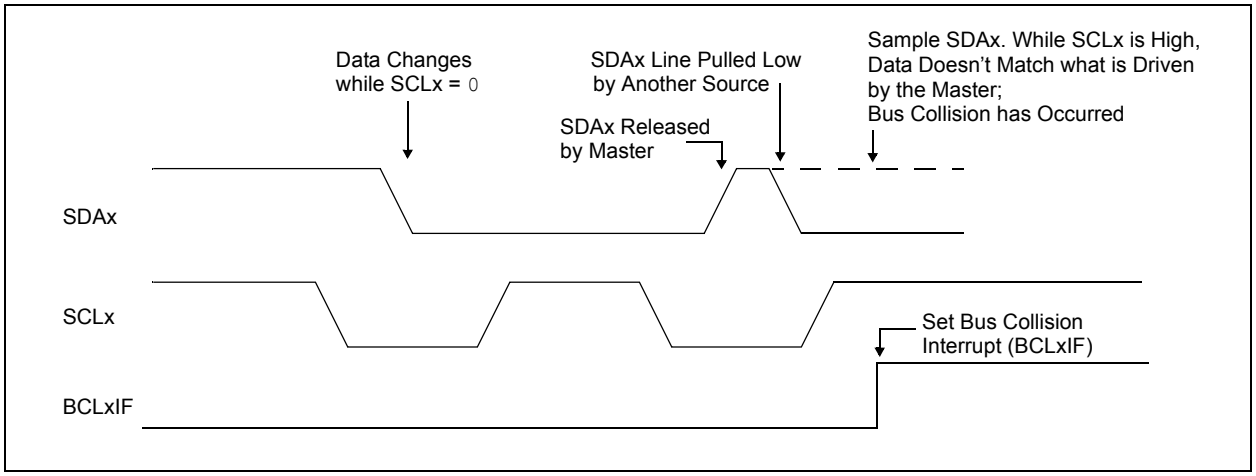
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDAx and SCLx lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine (ISR), and if the I²C 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 I²C 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 21-27: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE**





© 2009-2018 Microchip Technology Inc.

#### 21.4.17.1 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 21-28).

b) SCLx is sampled low before SDAx is asserted low (Figure 21-29).

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
- The MSSP module is reset to its inactive state (Figure 21-28)

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 21-30). 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 a bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert 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 21-28: BUS COLLISION DURING START CONDITION (SDAx ONLY)**

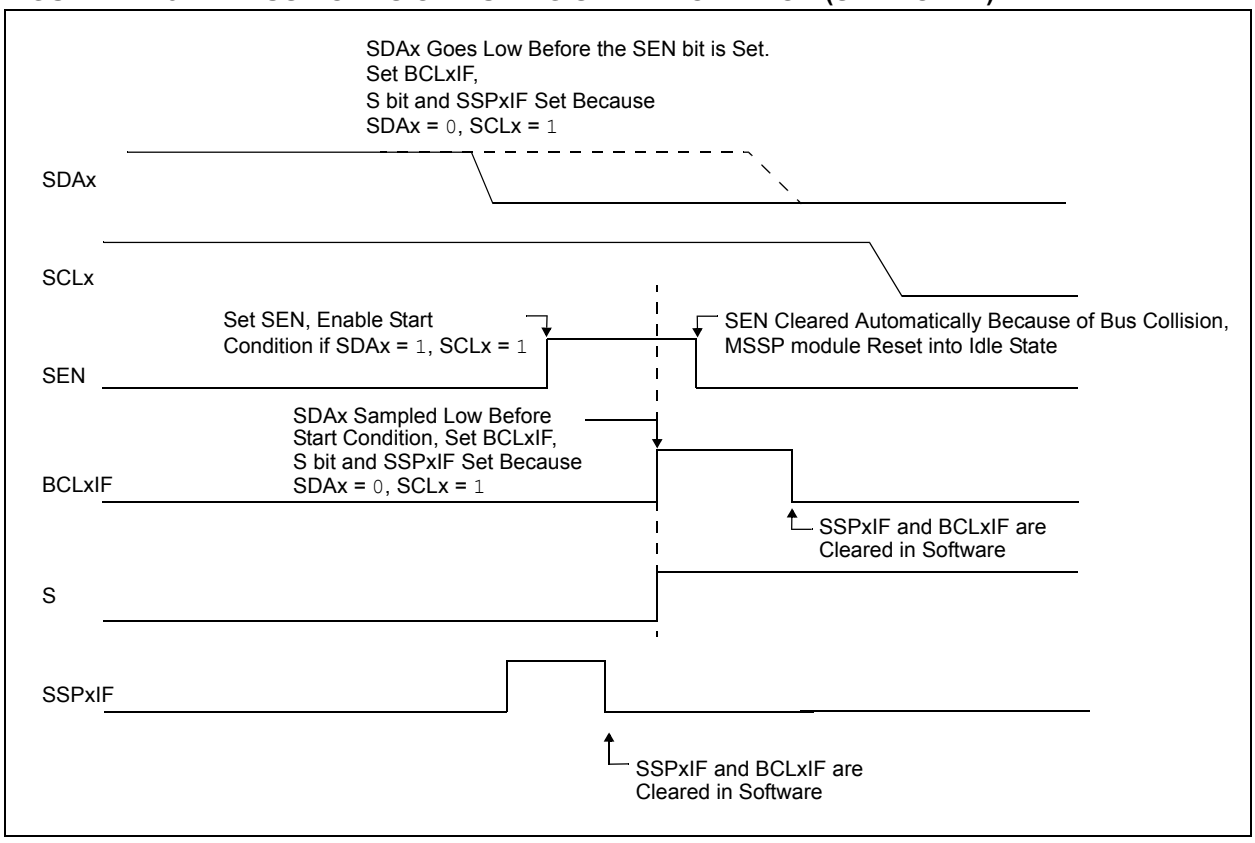


**FIGURE 21-29: BUS COLLISION DURING START CONDITION (SCLx = 0)**

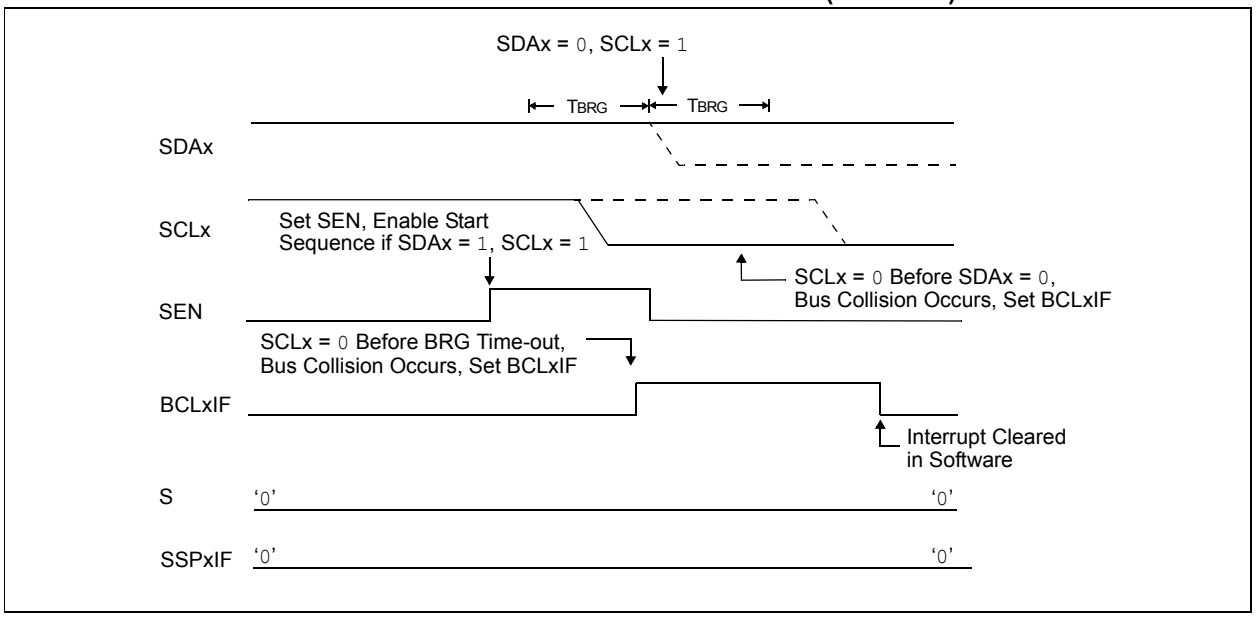
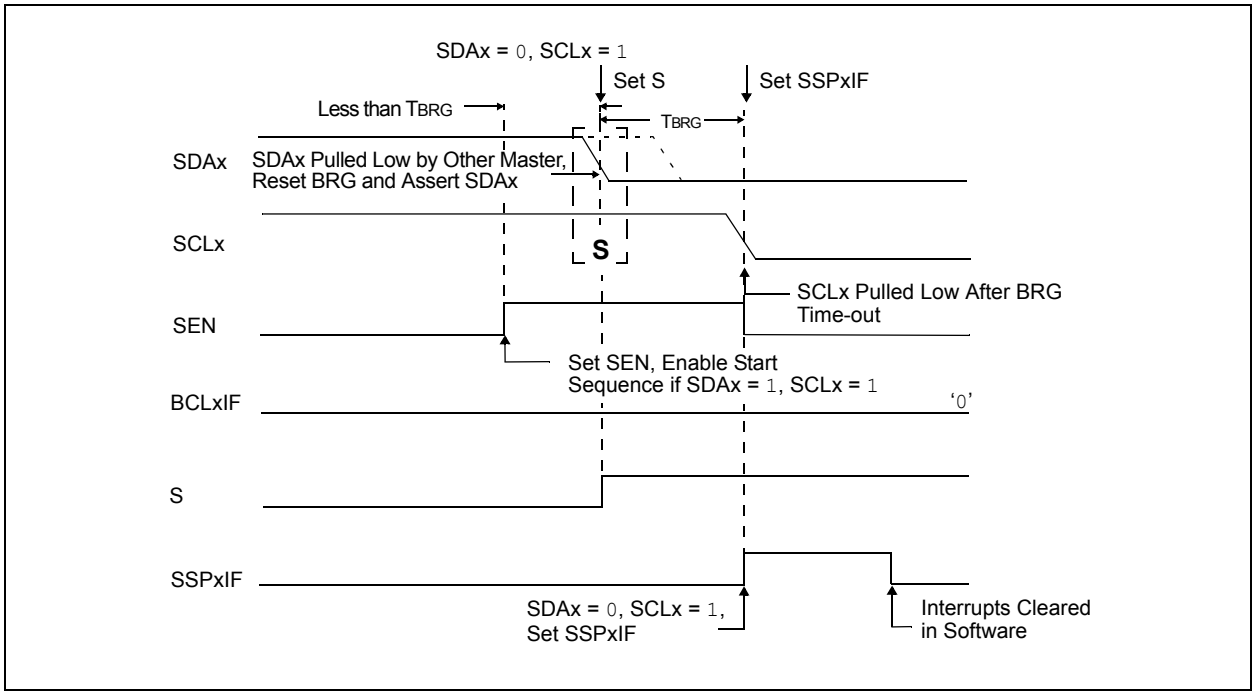



**FIGURE 21-30: BRG RESET DUE TO SDAx ARBITRATION DURING START CONDITION**





© 2009-2018 Microchip Technology Inc.

### 21.4.17.2 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 21-31). 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 21-32).

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 21-31: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)**

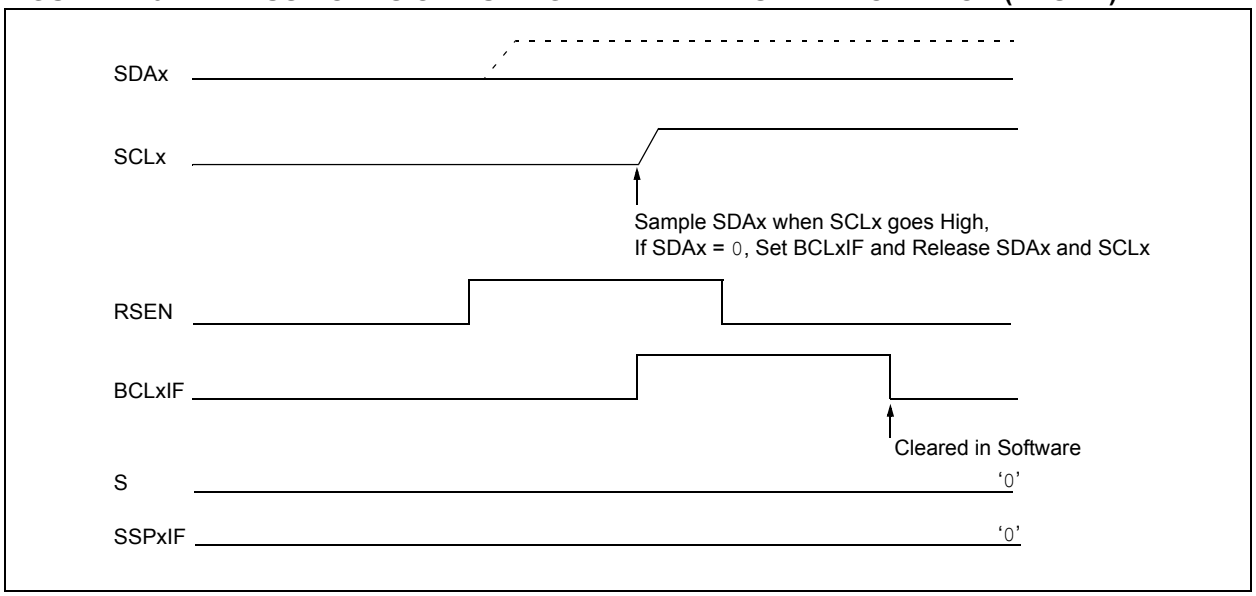
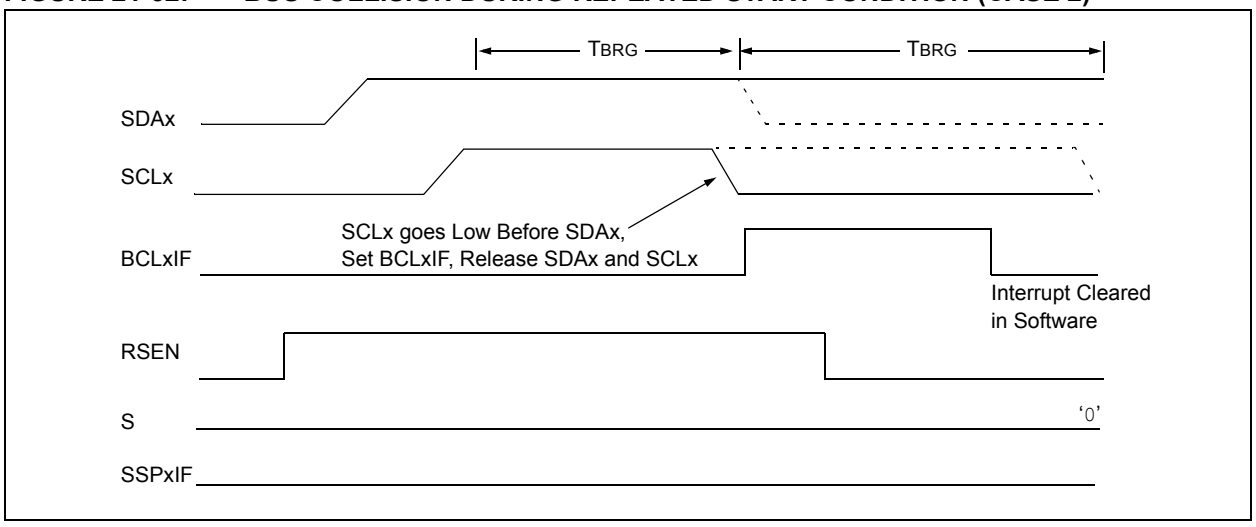



**FIGURE 21-32: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)**



#### 21.4.17.3 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 21-33). 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 21-34).

**FIGURE 21-33: BUS COLLISION DURING A STOP CONDITION (CASE 1)**

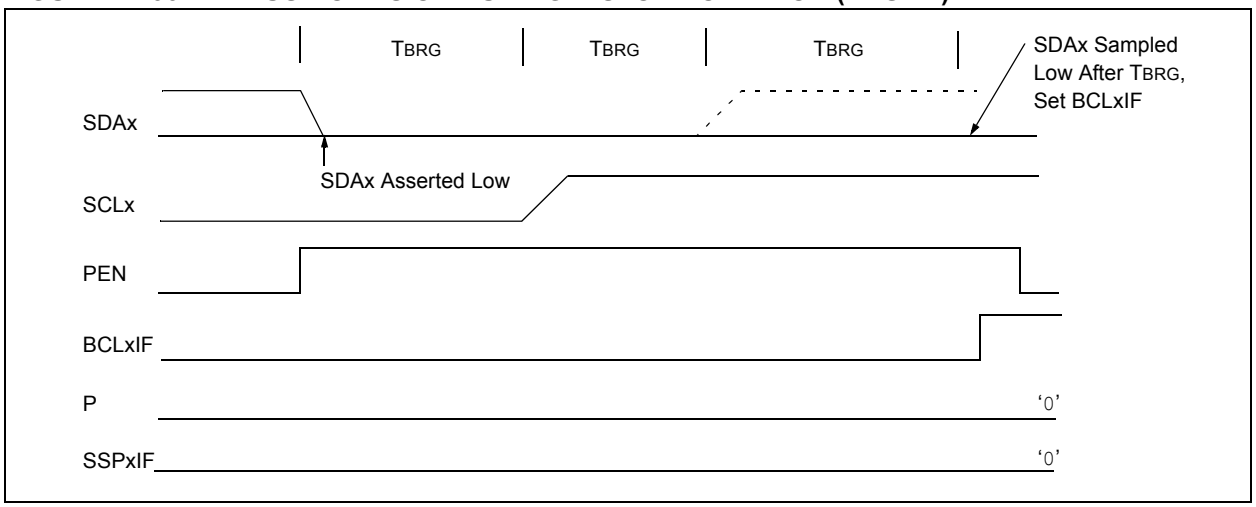
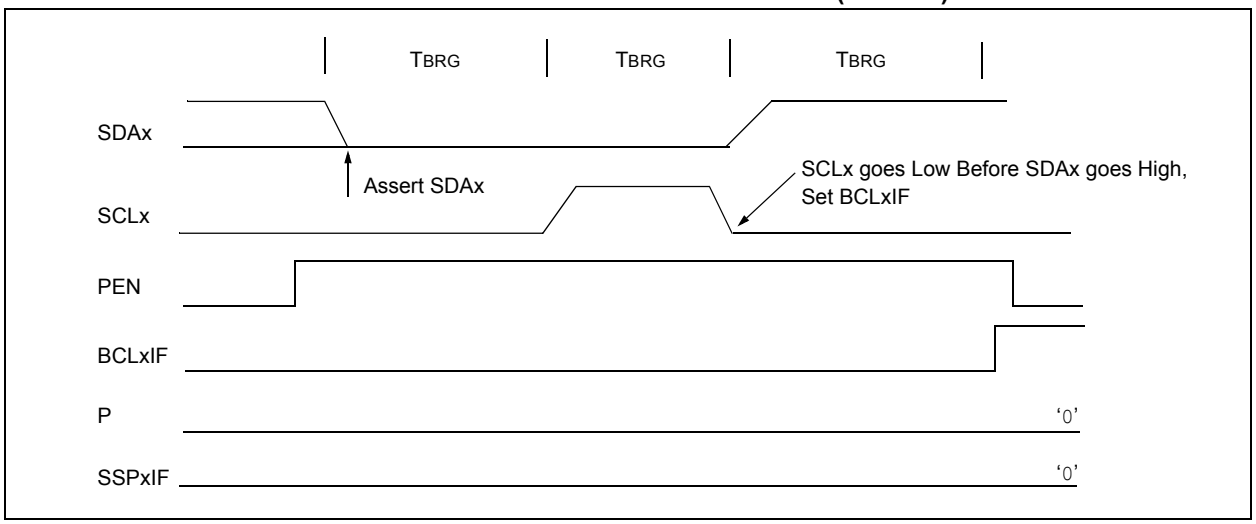



**FIGURE 21-34: BUS COLLISION DURING A STOP CONDITION (CASE 2)**





© 2009-2018 Microchip Technology Inc.

**TABLE 21-4: REGISTERS ASSOCIATED WITH I²C OPERATION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| PIR2 | OSCFIF | — | SSP2IF | BLC2IF | BCL1IF | HLVDIF | TMR3IF | TMR3GIF |
| PIE2 | OSCFIE | — | SSP2IE | BLC2IE | BCL1IE | HLVDIE | TMR3IE | TMR3GIE |
| IPR2 | OSCFIP | — | SSP2IP | BLC2IP | BCL1IP | HLVDIP | TMR3IP | TMR3GIP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 |
| SSP1BUF | MSSP1 Receive Buffer/Transmit Register | | | | | | | |
| SSP1ADD | MSSP1 Address Register (I²C Slave mode),<br>MSSP1 Baud Rate Reload Register (I²C Master mode) | | | | | | | |
| SSP1MSK(1) | MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0 |
| SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |
| SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| | GCEN | ACKSTAT | ADMSK5(2) | ADMSK4(2) | ADMSK3(2) | ADMSK2(2) | ADMSK1(2) | SEN |
| SSP1STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| SSP2BUF | MSSP2 Receive Buffer/Transmit Register | | | | | | | |
| SSP2ADD | MSSP2 Address Register (I²C Slave mode),<br>MSSP2 Baud Rate Reload Register (I²C Master mode) | | | | | | | |
| SSP2MSK(1) | MSK7 | MSK6 | MSK5 | MSK4 | MSK3 | MSK2 | MSK1 | MSK0 |
| SSP2CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |
| SSP2CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN |
| | GCEN | ACKSTAT | ADMSK5(2) | ADMSK4(2) | ADMSK3(2) | ADMSK2(2) | ADMSK1(2) | SEN |
| SSP2STAT | SMP | CKE | D/$\overline{A}$ | P | S | R/$\overline{W}$ | UA | BF |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**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 21.4.3.4 "7-Bit Address Masking Mode"** for more details.

**2:** Alternate bit definitions for use in I²C Slave mode operations only.

## 22.0 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 PIC18F87K22 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/AN19/C3OUT and RG2/RX2/DT2/AN18/C3INA), respectively. In order to configure these pins as an EUSART:

- For EUSART1:
  - Bit, SPEN (RCSTA1<7>), must be set (= 1)
  - Bit, TRISC<7>, must be set (= 1)
  - Bit, TRISC<6>, must be cleared (= 0) for Asynchronous and Synchronous Master modes
  - Bit, TRISC<6>, must be set (= 1) for Synchronous Slave mode
- For EUSART2:
  - Bit, SPEN (RCSTA2<7>), must be set (= 1)
  - Bit, TRISG<2>, must be set (= 1)
  - Bit TRISG<1> must be cleared (= 0) for Asynchronous and Synchronous Master modes
  - Bit, TRISC<6>, must be set (= 1) for Synchronous Slave mode

| **Note:** | The EUSART control will automatically reconfigure the pin from input to output as needed. |
|---|---|

The operation of each Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTAx)
- Receive Status and Control (RCSTAx)
- Baud Rate Control (BAUDCONx)

These are detailed on the following pages in Register 22-1, Register 22-2 and Register 22-3, 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. |
|---|---|


© 2009-2018 Microchip Technology Inc.

**REGISTER 22-1: TXSTAx: 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:** EUSART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send Sync Break on next transmission (cleared by hardware upon completion)
0 = Sync Break transmission 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
Can be address/data bit or a parity bit.

**Note 1:** SREN/CREN overrides TXEN in Sync mode.

**REGISTER 22-2: RCSTAx: 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 | OERR | 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 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 the 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 = 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 = 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.


© 2009-2018 Microchip Technology Inc.

**REGISTER 22-3: 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 ignored

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

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = EUSART will continue to sample the RXx pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RXx pin not monitored or rising edge detected
Synchronous mode:
Unused in this mode.

bit 0 **ABDEN:** Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Enable baud rate measurement on the next character. Requires reception of a Sync field (55h); cleared in hardware upon completion.
0 = Baud rate measurement is disabled or has completed
Synchronous mode:
Unused in this mode.

## 22.1 Baud Rate Generator (BRG)

The BRG is a dedicated, 8-bit or 16-bit generator that supports both the Asynchronous and Synchronous modes of the EUSART. By default, the BRG operates in 8-bit mode; setting the BRG16 bit (BAUDCONx<3>) selects 16-bit mode.

The SPBRGHx:SPBRGx register pair controls the period of a free-running timer. In Asynchronous mode, bits, BRGH (TXSTAx<2>) and BRG16 (BAUDCONx<3>), also control the baud rate. In Synchronous mode, BRGH is ignored. Table 22-1 shows the formula for computation of the baud rate for different EUSART modes which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRGHx:SPBRGx registers can be calculated using the formulas in Table 22-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 22-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 22-2. 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 the Synchronous mode, SPBRGH:SPBRG values of 0000h and 0001h are not supported. In the Asynchronous mode, all BRG values may be used.

### 22.1.1 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.

### 22.1.2 SAMPLING

The data on the RXx pin (either RC7/RX1/DT1 or RG2/RX2/DT2/AN18/C3INA) 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 22-1: BAUD RATE FORMULAS**

| Configuration Bits | | | BRG/EUSART 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


© 2009-2018 Microchip Technology Inc.

**EXAMPLE 22-1: 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 22-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH1 | EUSART1 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG1 | EUSART1 Baud Rate Generator Register | | | | | | | |
| TXSTA2 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| RCSTA2 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| BAUDCON2 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH2 | EUSART2 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG2 | EUSART2 Baud Rate Generator Register | | | | | | | |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

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

**TABLE 22-3: 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 | — | — | — | — | — | — |


© 2009-2018 Microchip Technology Inc.

**TABLE 22-3: 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 | — | — | — | — | — | — |

### 22.1.3 AUTO-BAUD RATE DETECT

The Enhanced USART module supports the automatic detection and calibration of baud rate. This feature is active only in Asynchronous mode and while the WUE bit is clear.

The automatic baud rate measurement sequence (Figure 22-1) 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 22-2).

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. The BRG clock will be configured by the BRG16 and BRGH bits. The BRG16 bit must be set to use both SPBRG1 and SPBRGH1 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 22-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the EUSART state machine is held in Idle. The RCxIF interrupt is set once the fifth rising edge on RXx is detected. The value in the RCREGx needs to be read to clear the RCxIF interrupt. The contents of RCREGx should be discarded.

**Note 1:** If the WUE bit is set with the ABDEN bit, Auto-Baud Rate Detection will occur on the byte *following* the Break character.

**2:** It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible due to bit error rates. Overall system timing and communication baud rates must be taken into consideration when using the Auto-Baud Rate Detection feature.

**3:** To maximize baud rate range, if that feature is used, it is recommended that the BRG16 bit (BAUDCONx<3>) be set.

**TABLE 22-4: BRG COUNTER CLOCK RATES**

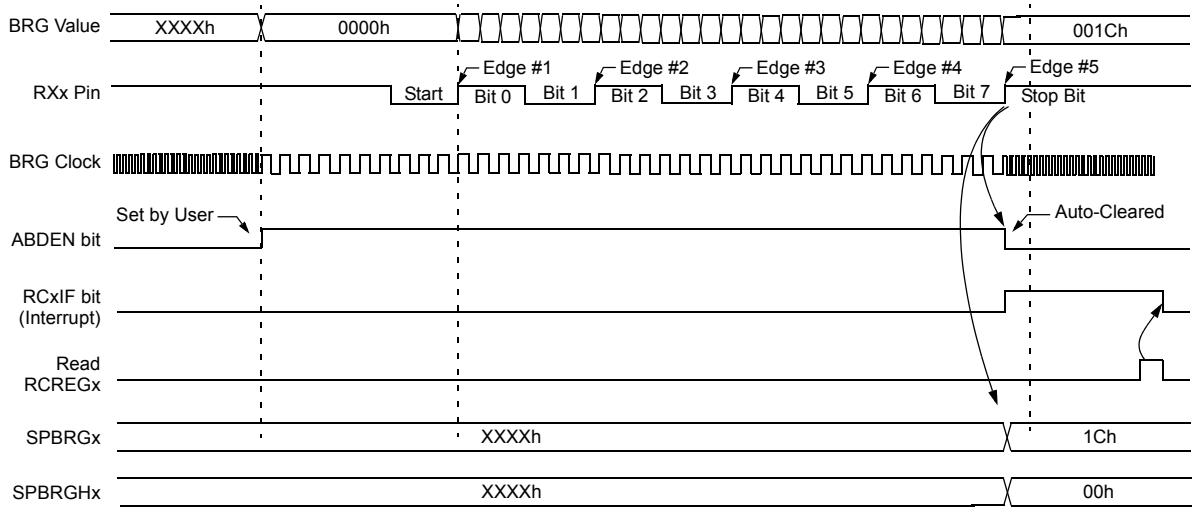
| BRG16 | BRGH | BRG Counter Clock |
|---|---|---|
| 0 | 0 | FOSC/512 |
| 0 | 1 | FOSC/128 |
| 1 | 0 | FOSC/128 |
| 1 | 1 | FOSC/32 |

#### 22.1.3.1 ABD and EUSART Transmission

Since the BRG clock is reversed during ABD acquisition, the EUSART transmitter cannot be used during ABD. This means that whenever the ABDEN bit is set, TXREGx cannot be written to. Users should also ensure that ABDEN does not become set during a transmit sequence. Failing to do this may result in unpredictable EUSART operation.

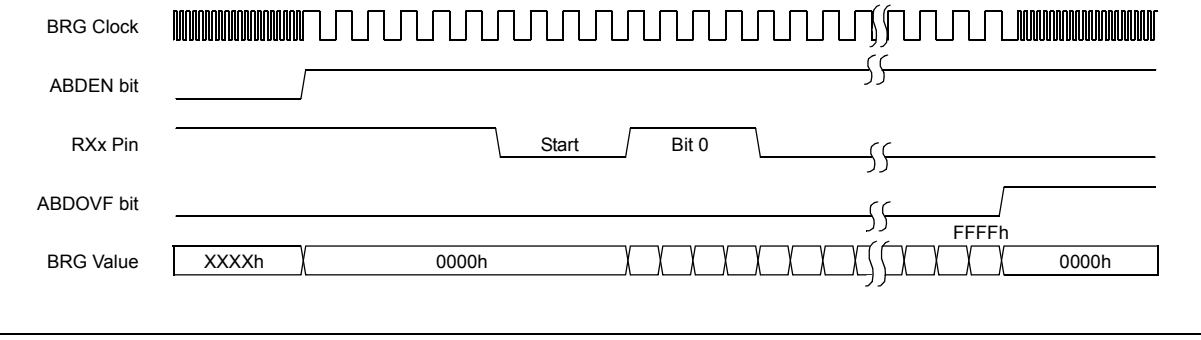

© 2009-2018 Microchip Technology Inc.

**FIGURE 22-1: AUTOMATIC BAUD RATE CALCULATION**




**Note:** The ABD sequence requires the EUSART module to be configured in Asynchronous mode and WUE = 0.

**FIGURE 22-2: BRG OVERFLOW SEQUENCE**



## 22.2 EUSART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTAx<4>). In this mode, the EUSART uses standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine data bits and one Stop bit). The most common data format is 8 bits. An on-chip, dedicated 8-bit/16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent but use the same data format and baud rate. The Baud Rate Generator produces a clock, either x16 or x64 of the bit shift rate, depending on the BRGH and BRG16 bits (TXSTAx<2> and BAUDCONx<3>). Parity is not supported by the hardware but can be implemented in software and stored as the 9th data bit.

When operating in Asynchronous mode, the EUSART module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver
- Auto-Wake-up on Sync Break Character
- 12-Bit Break Character Transmit
- Auto-Baud Rate Detection

### 22.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in Figure 22-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREGx. The TXREGx register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREGx register (if available).

Once the TXREGx register transfers the data to the TSR register (occurs in one TCY), the TXREGx register is empty and the 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 an 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.


© 2009-2018 Microchip Technology Inc.

**FIGURE 22-3: EUSART TRANSMIT BLOCK DIAGRAM**

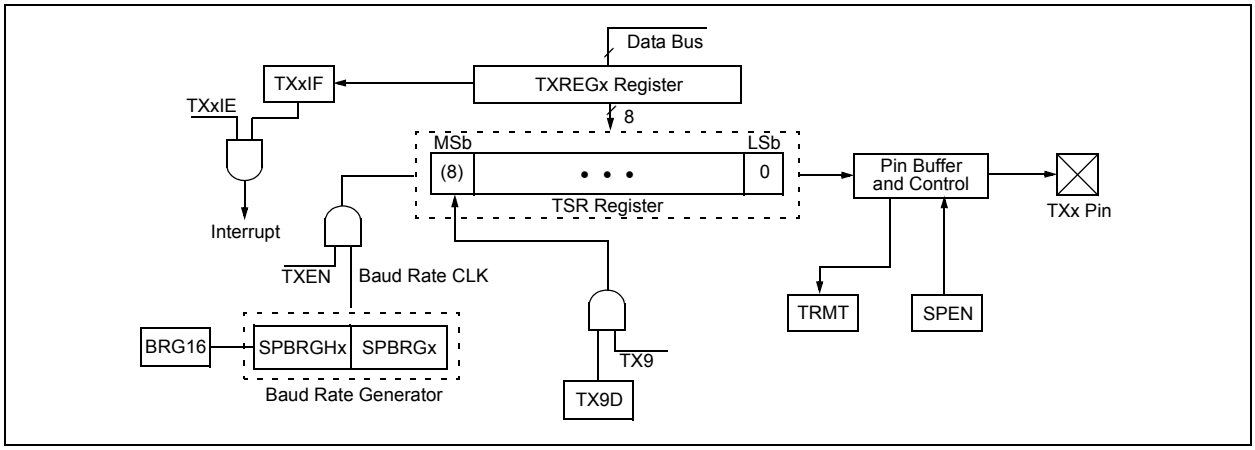



**FIGURE 22-4: ASYNCHRONOUS TRANSMISSION**

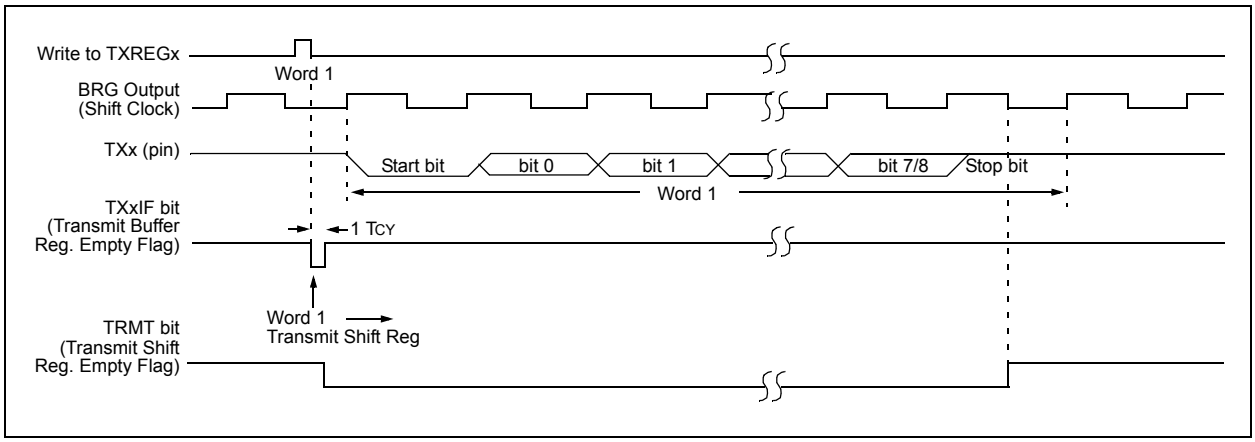
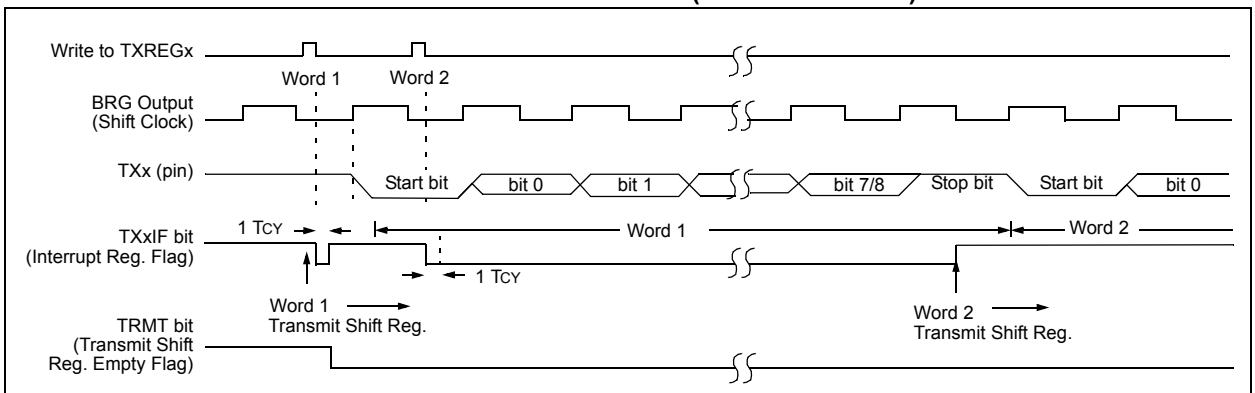



**FIGURE 22-5: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)**




**Note:** This timing diagram shows two consecutive transmissions.

**TABLE 22-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| TXREG1 | EUSART1 Transmit Register | | | | | | | |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH1 | EUSART1 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG1 | EUSART1 Baud Rate Generator Register | | | | | | | |
| RCSTA2 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| TXREG2 | EUSART2 Transmit Register | | | | | | | |
| TXSTA2 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON2 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH2 | EUSART2 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG2 | EUSART2 Baud Rate Generator Register | | | | | | | |
| ODCON3 | U2OD | U1OD | — | — | — | — | — | CTMUDS |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.


© 2009-2018 Microchip Technology Inc.

### 22.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 22-6. The data is received on the RXx pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

To set up an Asynchronous Reception:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RCxIE.
4. If 9-bit reception is desired, set bit, RX9.
5. Enable the reception by setting bit, CREN.
6. Flag bit, 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.

### 22.2.3 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 22-6: EUSART RECEIVE BLOCK DIAGRAM**

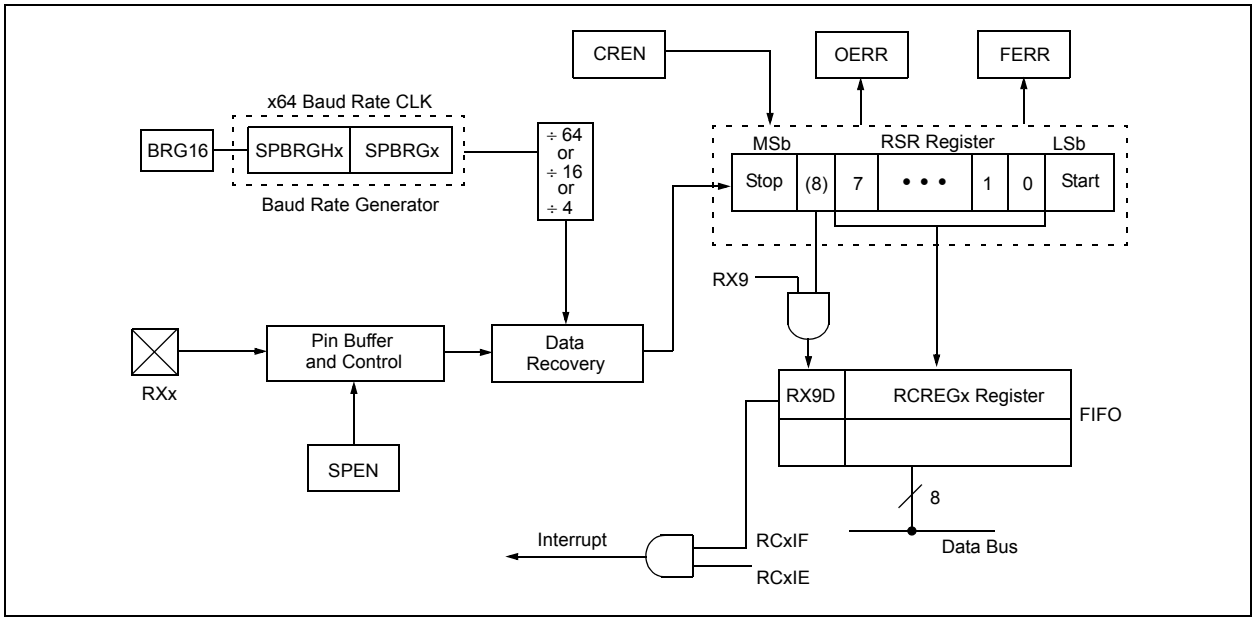
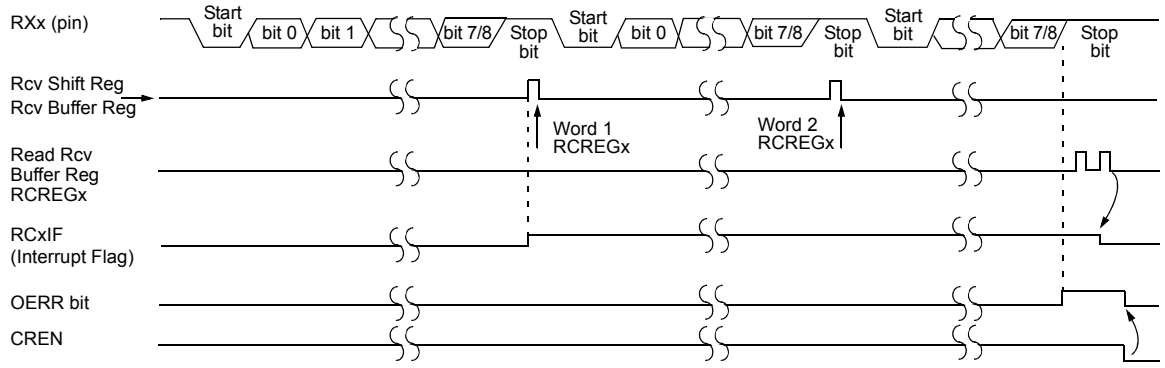


**FIGURE 22-7: ASYNCHRONOUS RECEPTION**




**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) bit to be set.

**TABLE 22-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| RCREG1 | EUSART1 Receive Register | | | | | | | |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH1 | EUSART1 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG1 | EUSART1 Baud Rate Generator Register | | | | | | | |
| RCSTA2 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| RCREG2 | EUSART2 Receive Register | | | | | | | |
| TXSTA2 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON2 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH2 | EUSART2 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG2 | EUSART2 Baud Rate Generator Register | | | | | | | |
| ODCON3 | U2OD | U1OD | — | — | — | — | — | CTMUDS |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**Legend:** — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.


© 2009-2018 Microchip Technology Inc.

### 22.2.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the Baud Rate Generator is inactive and a proper byte reception cannot be performed. The auto-wake-up feature allows the controller to wake up due to activity on the RXx/DTx line while the EUSART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCONx<1>). Once set, the typical receive sequence on RXx/DTx is disabled and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RXx/DTx line. (This coincides with the start of a Sync Break or a Wake-up Signal character for the LIN/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 22-8) and asynchronously if the device is in Sleep mode (Figure 22-9). The interrupt condition is cleared by reading the RCREGx register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RXx line following the wake-up event. At this point, the EUSART module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.

#### 22.2.4.1 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 EUSART.

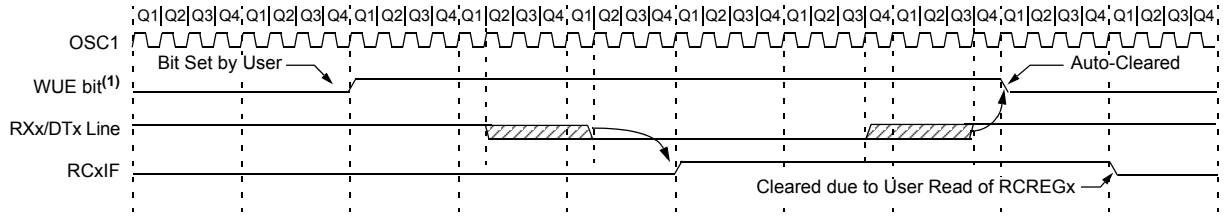
### 22.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCxIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the EUSART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCxIF bit. The WUE bit is cleared after this when a rising edge is seen on RXx/DTx. The interrupt condition is then cleared by reading the RCREGx register. Ordinarily, the data in RCREGx will be dummy data and should be discarded.

The fact that the WUE bit has been cleared (or is still set), and the RCxIF flag is set, should not be used as an indicator of the integrity of the data in RCREGx. Users should consider implementing a parallel method in firmware to verify received data integrity.

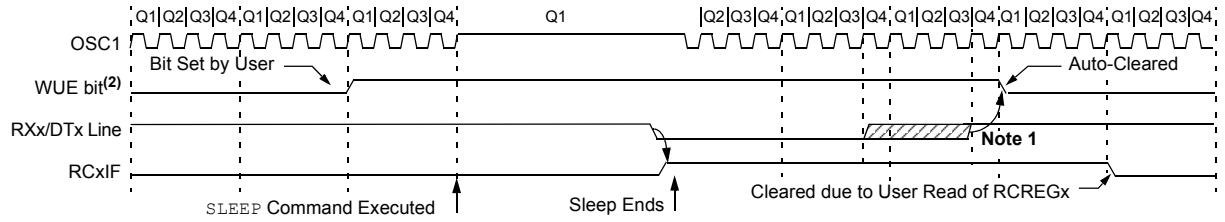
To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering Sleep mode.

**FIGURE 22-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION**




**Note 1:** The EUSART remains in Idle while the WUE bit is set.

**FIGURE 22-9: 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 EUSART remains in Idle while the WUE bit is set.


© 2009-2018 Microchip Technology Inc.

### 22.2.5 BREAK CHARACTER SEQUENCE

The EUSART 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>, respectively) 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 22-10 for the timing of the Break character sequence.

#### 22.2.5.1 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 EUSART for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.
3. Load the TXREGx with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREGx to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREGx becomes empty, as indicated by the TXxIF, the next data byte can be written to TXREGx.

### 22.2.6 RECEIVING A BREAK CHARACTER

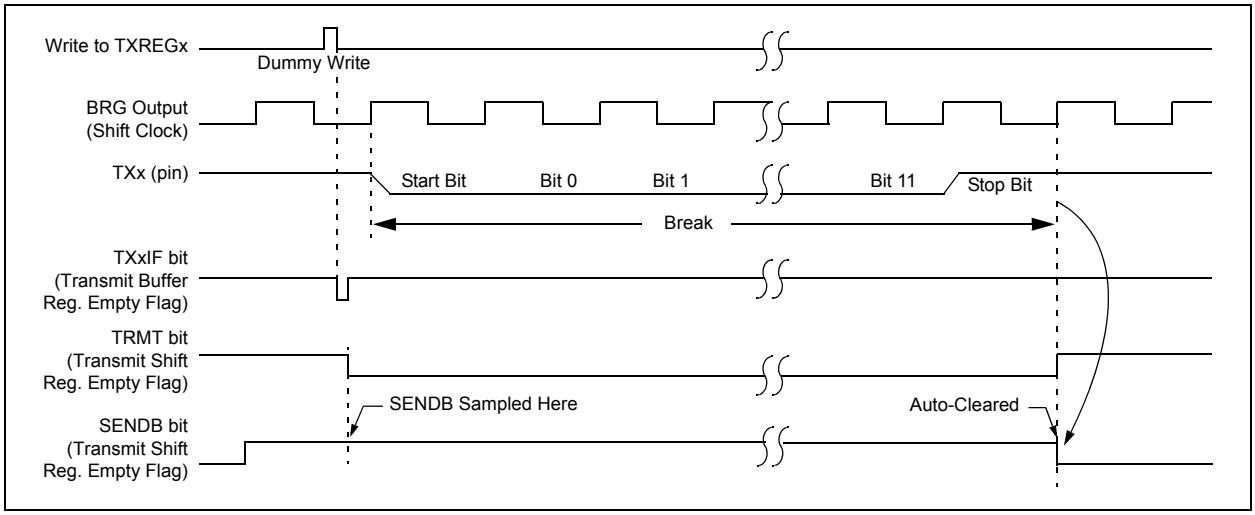
The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 22.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the EUSART will sample the next two transitions on RXx/DTx, cause an RCxIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABDEN bit once the TXxIF interrupt is observed.

**FIGURE 22-10: SEND BREAK CHARACTER SEQUENCE**



## 22.3 EUSART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTAx<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit, SYNC (TXSTAx<4>). In addition, enable bit, SPEN (RCSTAx<7>), is set in order to configure the TXx and RXx pins to CKx (clock) and DTx (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CKx line. Clock polarity is selected with the TXCKP bit (BAUDCONx<4>). Setting TXCKP sets the Idle state on CKx as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

### 22.3.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in Figure 22-3. 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).

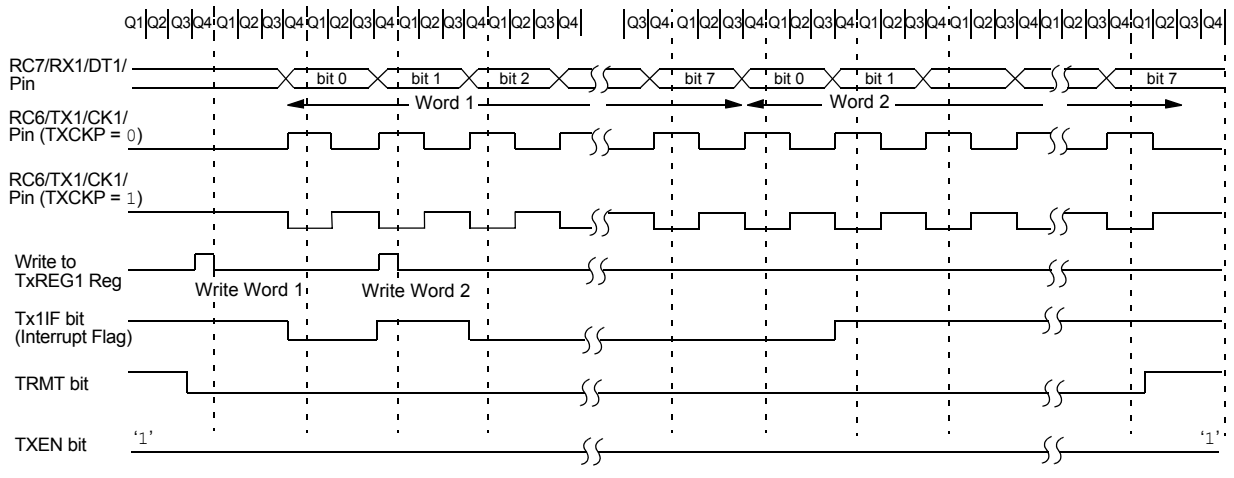
Once the TXREGx register transfers the data to the TSR register (occurs in one TCY), the TXREGx is empty and the TXxIF flag bit is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxIE. TXxIF is set regardless of the state of enable bit, TXxIE; it cannot 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.

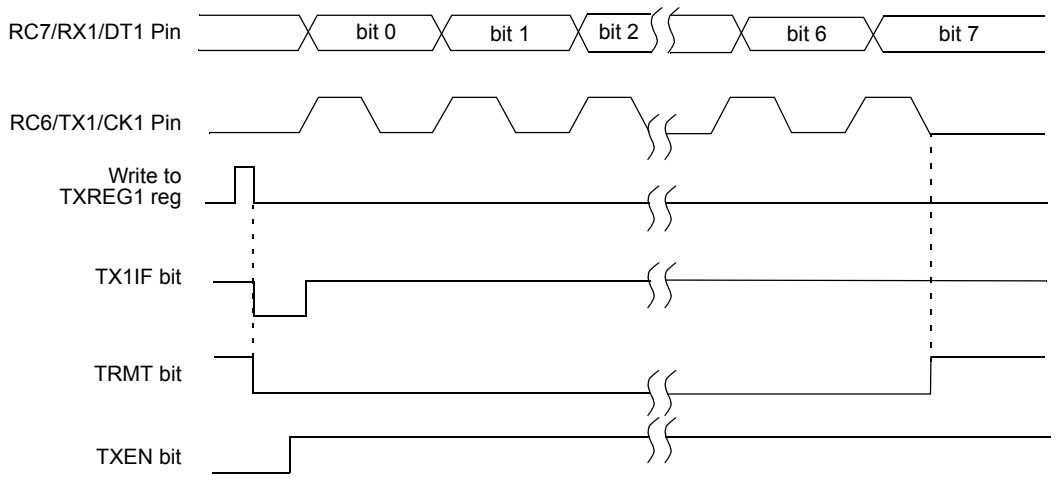
**FIGURE 22-11: SYNCHRONOUS TRANSMISSION**




**Note:** Sync Master mode, SPBRGx = 0, continuous transmission of two 8-bit words. This example is equally applicable to EUSART2 (RG1/TX2/CK2/AN19/C3OUT and RG2/RX2/DT2/AN18/C3INA).


© 2009-2018 Microchip Technology Inc.

**FIGURE 22-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)**




**Note:** This example is equally applicable to EUSART2 (RG1/TX2/CK2/AN19/C3OUT and RG2/RX2/DT2/AN18/C3INA).

**TABLE 22-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| TXREG1 | EUSART1 Transmit Register | | | | | | | |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH1 | EUSART1 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG1 | EUSART1 Baud Rate Generator Register | | | | | | | |
| RCSTA2 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| TXREG2 | EUSART2 Transmit Register | | | | | | | |
| TXSTA2 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON2 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH2 | EUSART2 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG2 | EUSART2 Baud Rate Generator Register | | | | | | | |
| ODCON3 | U2OD | U1OD | — | — | — | — | — | CTMUDS |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

### 22.3.2 EUSART SYNCHRONOUS MASTER RECEPTION

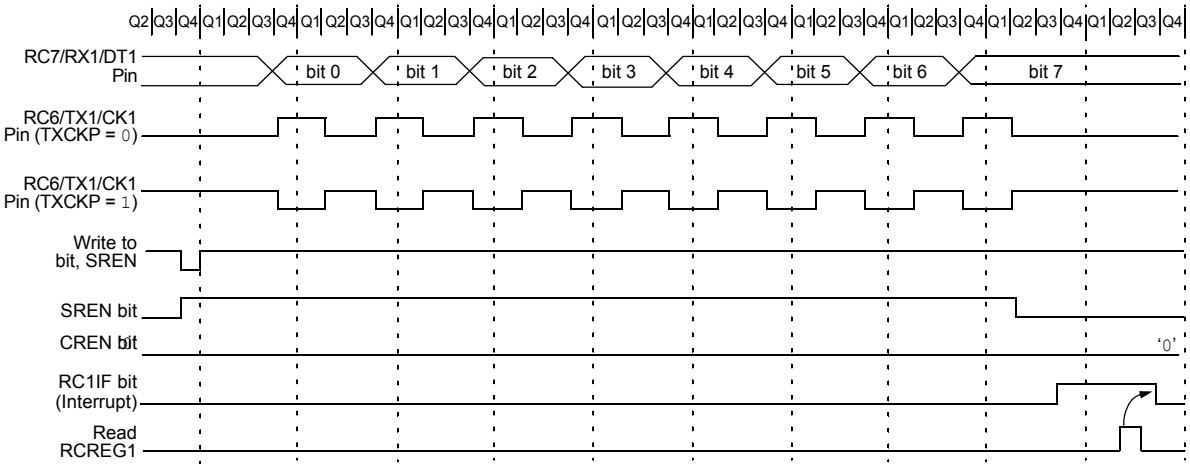
Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTAx<5>), or the Continuous Receive Enable bit, CREN (RCSTAx<4>). Data is sampled on the RXx pin on the falling edge of the clock.

If enable bit, SREN, is set, only a single word is received. If enable bit, CREN, is set, the reception is continuous until CREN is cleared. If both bits are set, then CREN takes precedence.

To set up a Synchronous Master Reception:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. Ensure bits, CREN and SREN, are clear.
4. If interrupts are desired, set enable bit, RCxIE.
5. If 9-bit reception is desired, set bit, RX9.
6. If a single reception is required, set bit, SREN. For continuous reception, set bit, CREN.
7. Interrupt flag bit, RCxIF, will be set when reception is complete and an interrupt will be generated if the enable bit, RCxIE, was set.
8. Read the RCSTAx register to get the 9th bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREGx register.
10. If any error occurred, clear the error by clearing bit CREN.
11. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**FIGURE 22-13: 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/AN19/C3OUT and RG2/RX2/DT2/AN18/C3INA).


© 2009-2018 Microchip Technology Inc.

**TABLE 22-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| RCREG1 | EUSART1 Receive Register | | | | | | | |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH1 | EUSART1 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG1 | EUSART1 Baud Rate Generator Register | | | | | | | |
| RCSTA2 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| RCREG2 | EUSART2 Receive Register | | | | | | | |
| TXSTA2 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON2 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH2 | EUSART2 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG2 | EUSART2 Baud Rate Generator Register | | | | | | | |
| ODCON3 | U2OD | U1OD | — | — | — | — | — | CTMUDS |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.

## 22.4 EUSART Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit, CSRC (TXSTAx<7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the CKx pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

### 22.4.1 EUSART SYNCHRONOUS SLAVE TRANSMISSION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep mode.

If two words are written to the TXREGx and then the SLEEP instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.
b) The second word will remain in the TXREGx register.
c) Flag bit, 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 22-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| TXREG1 | EUSART1 Transmit Register | | | | | | | |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH1 | EUSART1 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG1 | EUSART1 Baud Rate Generator Register | | | | | | | |
| RCSTA2 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| TXREG2 | EUSART2 Transmit Register | | | | | | | |
| TXSTA2 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON2 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH2 | EUSART2 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG2 | EUSART2 Baud Rate Generator Register | | | | | | | |
| ODCON3 | U2OD | U1OD | — | — | — | — | — | CTMUDS |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.


© 2009-2018 Microchip Technology Inc.

### 22.4.2 EUSART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical except in the case of Sleep, or any Idle mode, and bit, SREN, which is a “don’t care” in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep or any Idle mode, then a word may be received while in this low-power mode. Once the word is received, the RSR register will transfer the data to the RCREGx register. If the RCxIE enable bit is set, the interrupt generated will wake the chip from the low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a Synchronous Slave Reception:

1. Enable the synchronous master serial port by setting bits, SYNC and SPEN, and clearing bit, CSRC.
2. If interrupts are desired, set enable bit, RCxIE.
3. If 9-bit reception is desired, set bit, RX9.
4. To enable reception, set enable bit, CREN.
5. Flag bit, RCxIF, will be set when reception is complete. An interrupt will be generated if enable bit, RCxIE, was set.
6. Read the RCSTAx register to get the 9th bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREGx register.
8. If any error occurred, clear the error by clearing bit, CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

**TABLE 22-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| RCSTA1 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| RCREG1 | EUSART1 Receive Register | | | | | | | |
| TXSTA1 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON1 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH1 | EUSART1 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG1 | EUSART1 Baud Rate Generator Register | | | | | | | |
| RCSTA2 | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| RCREG2 | EUSART2 Receive Register | | | | | | | |
| TXSTA2 | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D |
| BAUDCON2 | ABDOVF | RCIDL | RXDTP | TXCKP | BRG16 | — | WUE | ABDEN |
| SPBRGH2 | EUSART2 Baud Rate Generator Register High Byte | | | | | | | |
| SPBRG2 | EUSART2 Baud Rate Generator Register | | | | | | | |
| ODCON3 | U2OD | U1OD | — | — | — | — | — | CTMUDS |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**Legend:** — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.

## 23.0 12-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module in the PIC18F87K22 family of devices has 16 inputs for the 64-pin devices and 24 inputs for the 80-pin devices. This module allows conversion of an analog input signal to a corresponding 12-bit digital number.

The module has these registers:

- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)
- A/D Port Configuration Register 0 (ANCON0)
- A/D Port Configuration Register 1 (ANCON1)
- A/D Port Configuration Register 2 (ANCON2)
- ADRESH (the upper, A/D Results register)
- ADRESL (the lower, A/D Results register)

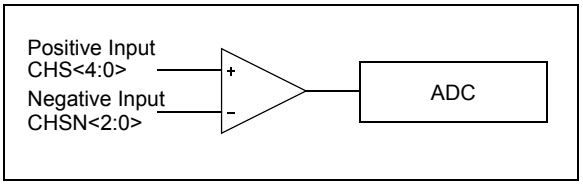
The ADCON0 register, shown in Register 23-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 23-2, configures the voltage reference and special trigger selection. The ADCON2 register, shown in Register 23-3, configures the A/D clock source and programmed acquisition time and justification.

### 23.1 Differential A/D Converter

The converter in PIC18F87K22 family devices is implemented as a differential A/D where the differential voltage between two channels is measured and converted to digital values (see Figure 23-1).

The converter can also be configured to measure a voltage from a single input by clearing the CHSN bits (ADCON1<2:0>). With this configuration, the negative channel input is connected internally to AVSS (see Figure 23-2).

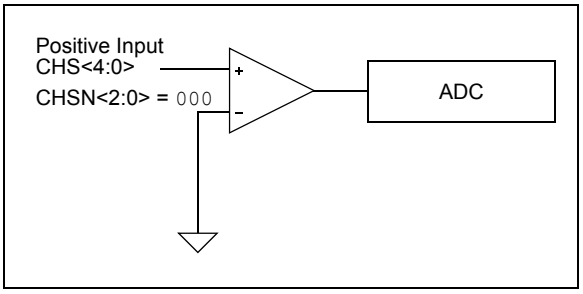
**FIGURE 23-1: DIFFERENTIAL CHANNEL MEASUREMENT**




Differential conversion feeds the two input channels to a unity gain differential amplifier. The positive channel input is selected using the CHS bits (ADCON0<6:2>) and the negative channel input is selected using the CHSN bits (ADCON1<2:0>).

The output from the amplifier is fed to the A/D Converter, as shown in Figure 23-1. The 12-bit result is available on the ADRESH and ADRESL registers. An additional bit indicates if the 12-bit result is a positive or negative value.

**FIGURE 23-2: SINGLE CHANNEL MEASUREMENT**




In the Single Channel Measurement mode, the negative input is connected to AVSS by clearing the CHSN bits (ADCON1<2:0>).


© 2009-2018 Microchip Technology Inc.

## 23.2 A/D Registers

### 23.2.1 A/D CONTROL REGISTERS

**REGISTER 23-1: ADCON0: A/D CONTROL REGISTER 0**

| U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| — | CHS4 | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | 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 **Unimplemented:** Read as '0'

bit 6-2 **CHS<4:0>:** Analog Channel Select bits

| | |
|---|---|
| 00000 = Channel 00 (AN0) | 10000 = Channel 16 (AN16) |
| 00001 = Channel 01 (AN1) | 10001 = Channel 17 (AN17) |
| 00010 = Channel 02 (AN2) | 10010 = Channel 18 (AN18) |
| 00011 = Channel 03 (AN3) | 10011 = Channel 19 (AN19) |
| 00100 = Channel 04 (AN4) | 10100 = Channel 20 (AN20)(1,2) |
| 00101 = Channel 05 (AN5) | 10101 = Channel 21 (AN21)(1,2) |
| 00110 = Channel 06 (AN6) | 10110 = Channel 22 (AN22)(1,2) |
| 00111 = Channel 07 (AN7) | 10111 = Channel 23 (AN23)(1,2) |
| 01000 = Channel 08 (AN8) | 11000 = (Reserved)(2) |
| 01001 = Channel 09 (AN9) | 11001 = (Reserved)(2) |
| 01010 = Channel 10 (AN10 | 11010 = (Reserved)(2) |
| 01011 = Channel 11 (AN11) | 11011 = (Reserved)(2) |
| 01100 = Channel 12 (AN12)(1,2) | 11100 = Channel 28 (Reserved CTMU) |
| 01101 = Channel 13 (AN13)(1,2) | 11101 = Channel 29 (Internal temperature diode) |
| 01110 = Channel 14 (AN14)(1,2) | 11110 = Channel 30 ($V_{DDCORE}$) |
| 01111 = Channel 15 (AN15)(1,2) | 11111 = Channel 31 (v1.024V band gap) |

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

1 = A/D (or calibration) cycle in progress. Setting this bit starts an A/D conversion cycle. The bit is cleared automatically by hardware when the A/D conversion is completed.

0 = A/D conversion has completed or is not in progress

bit 0 **ADON:** A/D On bit

1 = A/D Converter is operating

0 = A/D conversion module is shut off and consuming no operating current

**Note 1:** These channels are not implemented on 64-pin devices.

**2:** Performing a conversion on unimplemented channels will return random values.

**REGISTER 23-2: ADCON1: A/D CONTROL 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 |
|---|---|---|---|---|---|---|---|
| TRIGSEL1 | TRIGSEL0 | VCFG1 | VCFG0 | VNCFG | CHSN2 | CHSN1 | CHSN0 |
| 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 **TRIGSEL<1:0>:** Special Trigger Select bits
11 = Selects the special trigger from the RTCC
10 = Selects the special trigger from the Timer1
01 = Selects the special trigger from the CTMU
00 = Selects the special trigger from the ECCP2

bit 5-4 **VCFG<1:0>:** A/D $V_{REF}$+ Configuration bits
11 = Internal $V_{REF}$+ (4.096V)
10 = Internal $V_{REF}$+ (2.048V)
01 = External $V_{REF}$+
00 = $AV_{DD}$

bit 3 **VNCFG:** A/D $V_{REF}$- Configuration bit
1 = External $V_{REF}$
0 = $AV_{SS}$

bit 2-0 **CHSN<2:0>:** Analog Negative Channel Select bits
111 = Channel 07 (AN6)
110 = Channel 06 (AN5)
101 = Channel 05 (AN4)
100 = Channel 04 (AN3)
011 = Channel 03 (AN2)
010 = Channel 02 (AN1)
001 = Channel 01 (AN0)
000 = Channel 00 ($AV_{SS}$)


© 2009-2018 Microchip Technology Inc.

**REGISTER 23-3: ADCON2: A/D CONTROL REGISTER 2**

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| ADFM | — | 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 **Unimplemented:** Read as '0'

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(1)

bit 2-0 **ADCS<2:0>:** A/D Conversion Clock Select bits
111 = FRC (clock derived from A/D RC oscillator)(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC (clock derived from A/D RC oscillator)(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

**Note 1:** 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.

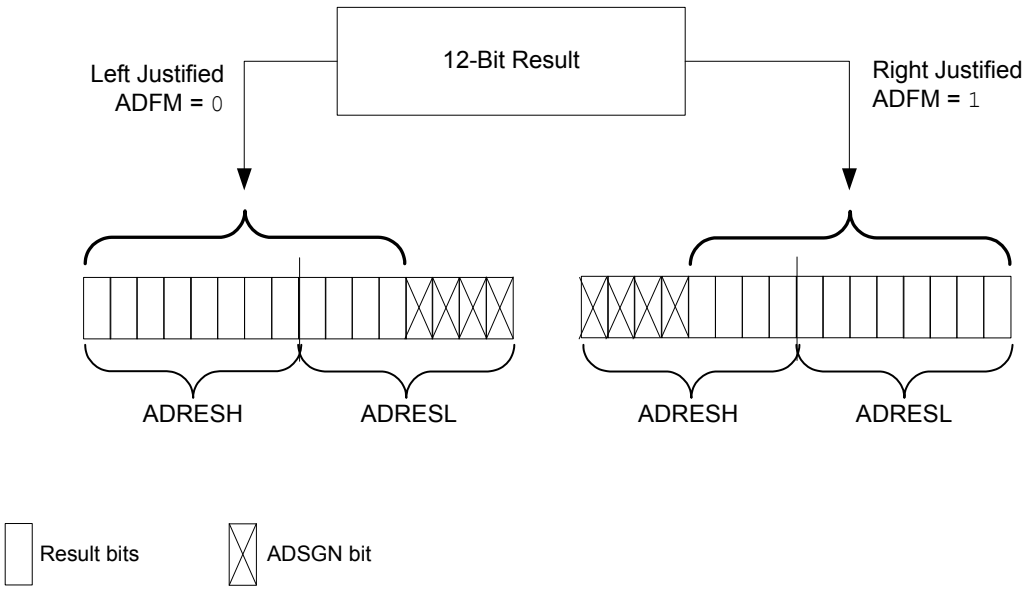
### 23.2.2 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is where the 12-bit A/D result and extended sign bits (ADSGN) are loaded at the completion of a conversion. This register pair is 16 bits wide. The A/D module gives the flexibility of left or right justifying the 12-bit result in the 16-Bit Result register. The A/D Format Select bit (ADFM) controls this justification.

Figure 23-3 shows the operation of the A/D result justification and location of the extended sign bits (ADSGN). The extended sign bits allow for easier 16-bit math to be performed on the result. The results are represented as a two's compliment binary value. This means that when sign bits and magnitude bits are considered together in right justification, the ADRESH and ADRESL registers can be read as a single signed integer value.

When the A/D Converter is disabled, these 8-bit registers can be used as two general purpose registers.

**FIGURE 23-3: A/D RESULT JUSTIFICATION**




**Two's Complement Example Results Number Line**

| Left Justified | | Right Justified | |
|---|---|---|---|
| **Hex** | **Decimal** | **Hex** | **Decimal** |
| 0xFFF0 | 4095 | 0x0FFF | 4095 |
| 0xFFE0 | 4094 | 0x0FFE | 4094 |
| ... | ... | ... | ... |
| 0x0020 | 2 | 0x0002 | 2 |
| 0x0010 | 1 | 0x0001 | 1 |
| 0x0000 | 0 | 0x0000 | 0 |
| 0xFFFF | -1 | 0xFFFF | -1 |
| 0xFFEF | -2 | 0xFFFE | -2 |
| ... | ... | ... | ... |
| 0x001F | -4095 | 0xF001 | -4095 |
| 0x000F | -4096 | 0xF000 | -4096 |


© 2009-2018 Microchip Technology Inc.

**REGISTER 23-4: ADRESH: A/D RESULT HIGH BYTE REGISTER, LEFT JUSTIFIED (ADFM = 0)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| ADRES11 | ADRES10 | ADRES9 | ADRES8 | ADRES7 | ADRES6 | ADRES5 | ADRES4 |
| 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 **ADRES<11:4>:** A/D Result High Byte bits

**REGISTER 23-5: ADRESL: A/D RESULT HIGH BYTE REGISTER, LEFT JUSTIFIED (ADFM = 0)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| ADRES3 | ADRES2 | ADRES1 | ADRES0 | ADSGN | ADSGN | ADSGN | ADSGN |
| 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 7-4 **ADRES<3:0>:** A/D Result Low Byte bits

bit 3-0 **ADSGN:** A/D Result Sign bit

1 = A/D result is negative

0 = A/D result is positive

**REGISTER 23-6: ADRESH: A/D RESULT HIGH BYTE REGISTER, RIGHT JUSTIFIED (ADFM = 1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| ADSGN | ADSGN | ADSGN | ADSGN | ADRES11 | ADRES10 | ADRES9 | ADRES8 |
| 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 7-4 **ADSGN:** A/D Result Sign bit
1 = A/D result is negative
0 = A/D result is positive

bit 3-0 **ADRESH<11:8>:** A/D Result High Byte bits

**REGISTER 23-7: ADRESL: A/D RESULT LOW BYTE REGISTER, RIGHT JUSTIFIED (ADFM = 1)**

| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
|---|---|---|---|---|---|---|---|
| ADRES7 | ADRES6 | ADRES5 | ADRES4 | ADRES3 | ADRES2 | ADRES1 | ADRES0 |
| 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 **ADRES<7:0>:** A/D Result Low Byte bits


© 2009-2018 Microchip Technology Inc.

The ANCONx registers are used to configure the operation of the I/O pin associated with each analog channel. Clearing an ANSELx bit configures the corresponding pin (ANx) to operate as a digital only I/O. Setting a bit configures the pin to operate as an analog input for either the A/D Converter or the comparator module, with all digital peripherals disabled and digital inputs read as '0'.

As a rule, I/O pins that are multiplexed with analog inputs default to analog operation on any device Reset.

**REGISTER 23-8: ANCON0: A/D PORT CONFIGURATION REGISTER 0**

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---|---|---|---|---|---|---|---|
| ANSEL7 | ANSEL6 | ANSEL5 | ANSEL4 | ANSEL3 | ANSEL2 | ANSEL1 | ANSEL0 |
| 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 **ANSEL<7:0>:** Analog Port Configuration bits (AN7 and AN0)
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'
0 = Pin is configured as a digital port

**REGISTER 23-9: ANCON1: A/D PORT CONFIGURATION 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 |
|---|---|---|---|---|---|---|---|
| ANSEL15(1) | ANSEL14(1) | ANSEL13(1) | ANSEL12(1) | ANSEL11 | ANSEL10 | ANSEL9 | ANSEL8 |
| 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 **ANSEL<15:8>:** Analog Port Configuration bits (AN15 through AN8)(1)
1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'
0 = Pin is configured as a digital port

**Note 1:** AN15 through AN12 and AN23 to AN20 are implemented only on 80-pin devices. For 64-pin devices, the corresponding ANSELx bits are still implemented for these channels, but have no effect.

**REGISTER 23-10: ANCON2: A/D PORT CONFIGURATION 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 |
|---|---|---|---|---|---|---|---|
| ANSEL23(1) | ANSEL22(1) | ANSEL21(1) | ANSEL20(1) | ANSEL19 | ANSEL18 | ANSEL17 | ANSEL16 |
| 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 **ANSEL<23:16>:** Analog Port Configuration bits (AN23 through AN16)(1)

1 = Pin is configured as an analog channel; digital input is disabled and any inputs read as '0'

0 = Pin is configured as a digital port

**Note 1:** AN15 through AN12 and AN23 through AN20 are implemented only on 80-pin devices. For 64-pin devices, the corresponding ANSELx 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 ($AV_{DD}$ and $AV_{SS}$) or the voltage level on the RA3/AN3/$V_{REF}$+ and RA2/AN2/$V_{REF}$- pins. $V_{REF}$+ has two additional Internal Reference Voltage selections: 2.048V and 4.096V.

The A/D Converter can uniquely operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D Converter's internal RC oscillator.

The output of the Sample-and-Hold (S/H) is the input into the converter, which generates the result via successive approximation.

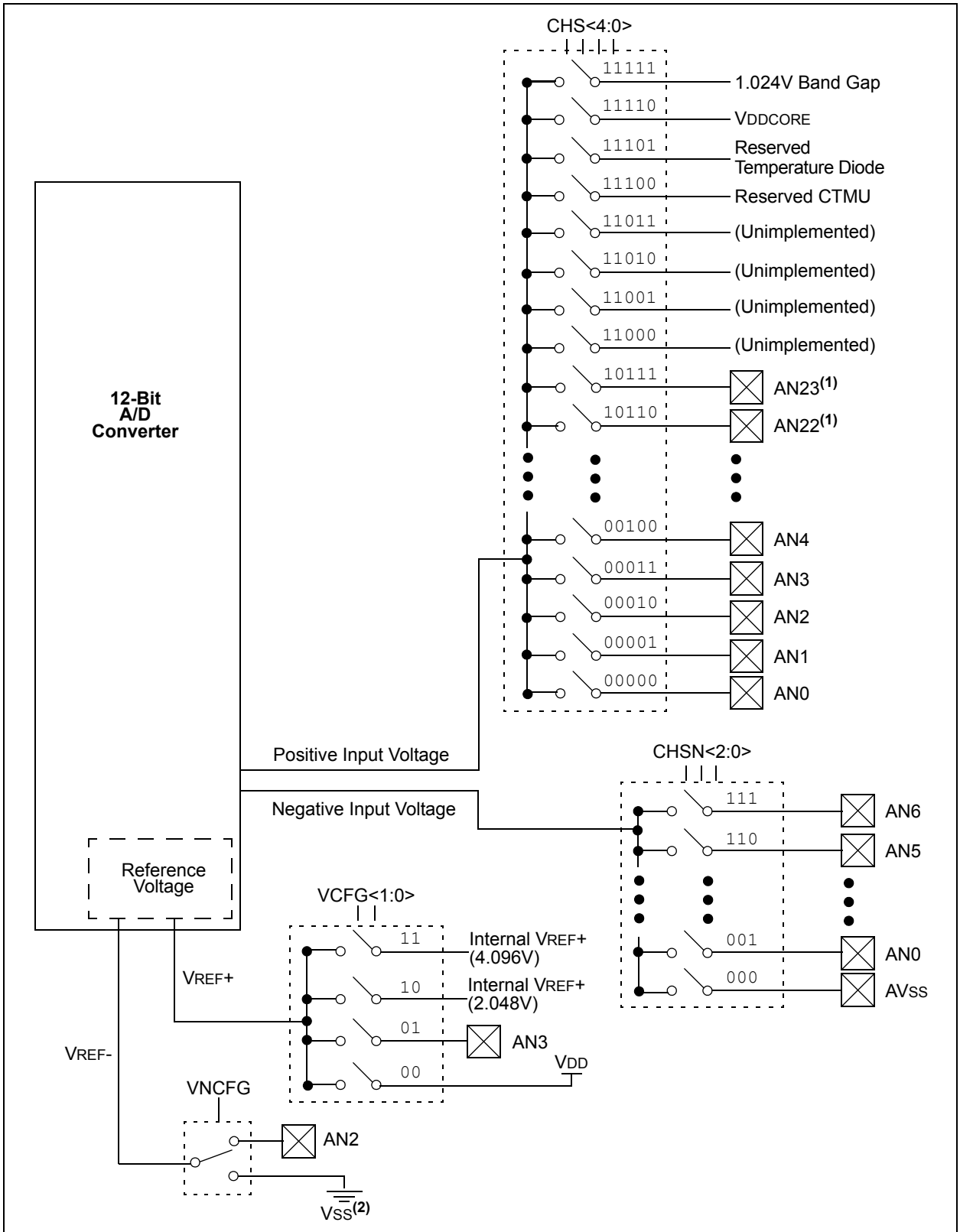
Each port pin associated with the A/D Converter can be configured as an analog input or a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH:ADRESL register pair, the GO/$\overline{\text{DONE}}$ bit (ADCON0<1>) is cleared and the A/D Interrupt Flag bit, ADIF (PIR1<6>), 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 23-4.


© 2009-2018 Microchip Technology Inc.

**FIGURE 23-4: A/D BLOCK DIAGRAM**




**Note 1:** Channels, AN15 through AN12, and AN20 through AN23, 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 can start. The analog input channels must have their corresponding TRIS bits selected as inputs. To determine acquisition time, see **Section 23.3 "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.

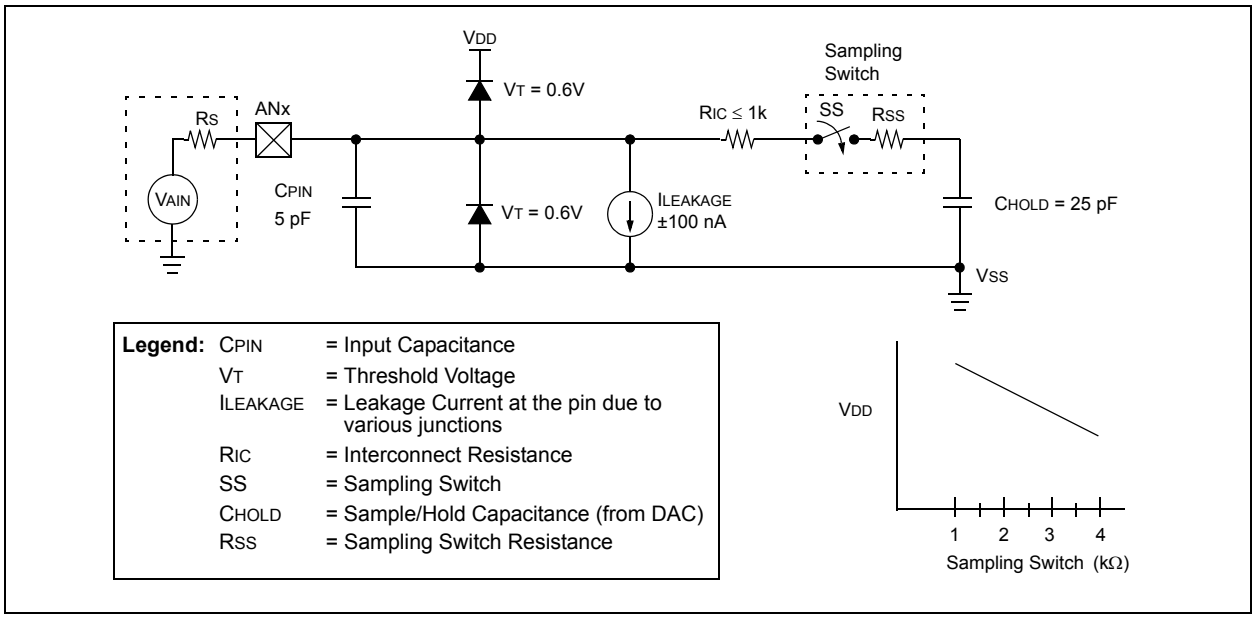
To do an A/D conversion, follow these steps:

1. Configure the A/D module:
   - Configure the required ADC pins as analog pins (ANCON0, ANCON1 and ANCON2)
   - Set the voltage reference (ADCON1)
   - Select the A/D positive and negative input channels (ADCON0 and ADCON1)
   - Select the A/D acquisition time (ADCON2)
   - Select the A/D conversion clock (ADCON2)
   - Turn on the A/D module (ADCON0)
2. Configure the A/D interrupt (if desired):
   - Clear the ADIF bit (PIR1<6>)
   - Set the ADIE bit (PIE1<6>)
   - Set the GIE bit (INTCON<7>)
3. Wait the required acquisition time (if required).
4. Start the conversion:
   - Set the 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) and, if required, clear bit, ADIF.
7. For the next conversion, begin with Step 1 or 2, as required.

   The A/D conversion time per bit is defined as $T_{AD}$. Before the next acquisition starts, a minimum Wait of 2 $T_{AD}$ is required.

**FIGURE 23-5: ANALOG INPUT MODEL**





© 2009-2018 Microchip Technology Inc.

## 23.3 A/D Acquisition Requirements

For the A/D Converter to meet its specified accuracy, the Charge Holding ($C_{HOLD}$) capacitor must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 23-5. 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 or 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 23-1 can be used. This equation assumes that 1/2 LSb error is used (1,024 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 23-3 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

**EQUATION 23-1: ACQUISITION TIME**

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$

**EQUATION 23-2: A/D MINIMUM CHARGING TIME**

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$
or
$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S) \ln(1/2048)$$

**EQUATION 23-3: 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.

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S) \ln(1/2048)\ \mu s$$
$$-(25\ pF)\ (1\ k\Omega + 2\ k\Omega + 2.5\ k\Omega) \ln(0.0004883)\ \mu s$$
$$1.05\ \mu s$$
$$T_{ACQ} = 0.2\ \mu s + 1.05\ \mu s + 1.2\ \mu s$$
$$2.45\ \mu s$$

© 2009-2018 Microchip Technology Inc.

## 23.4 Selecting and Configuring Automatic Acquisition Time

The ADCON2 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 (ADCON2<5:3>) remain in their Reset state ('000'), which 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.

## 23.5 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 14 TAD per 12-bit conversion. The source of the A/D conversion clock is software-selectable.

The possible options for TAD are:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Using the 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. (For more information, see Parameter 130 in Table 31-29.)

Table 23-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

**TABLE 23-1: TAD vs. DEVICE OPERATING FREQUENCIES**

| AD Clock Source (TAD) | | Maximum Device Frequency |
|---|---|---|
| Operation | ADCS<2:0> | |
| 2 TOSC | 000 | 2.50 MHz |
| 4 TOSC | 100 | 5.00 MHz |
| 8 TOSC | 001 | 10.00 MHz |
| 16 TOSC | 101 | 20.00 MHz |
| 32 TOSC | 010 | 40.00 MHz |
| 64 TOSC | 110 | 64.00 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.

## 23.6 Configuring Analog Port Pins

The ANCON0, ANCON1, ANCON2, TRISA, TRISF, TRISG and TRISH registers control the operation of the A/D port pins. The port pins needed as analog inputs must have their corresponding TRISx bits set (input). If the TRISx 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 TRISx 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.


© 2009-2018 Microchip Technology Inc.

## 23.7 A/D Conversions

Figure 23-6 shows the operation of the A/D Converter after the GO/$\overline{\text{DONE}}$ 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 23-7 shows the operation of the A/D Converter after the GO/$\overline{\text{DONE}}$ bit has been set, the ACQT<2:0> bits set to '010' and a 4 TAD acquisition time selected.

Clearing the GO/$\overline{\text{DONE}}$ 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/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D. |
|---|---|

**FIGURE 23-6: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)**

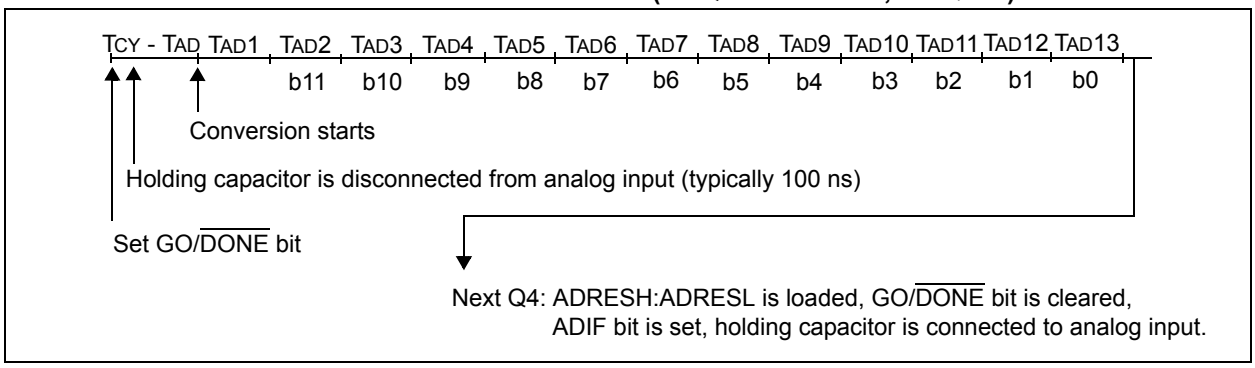
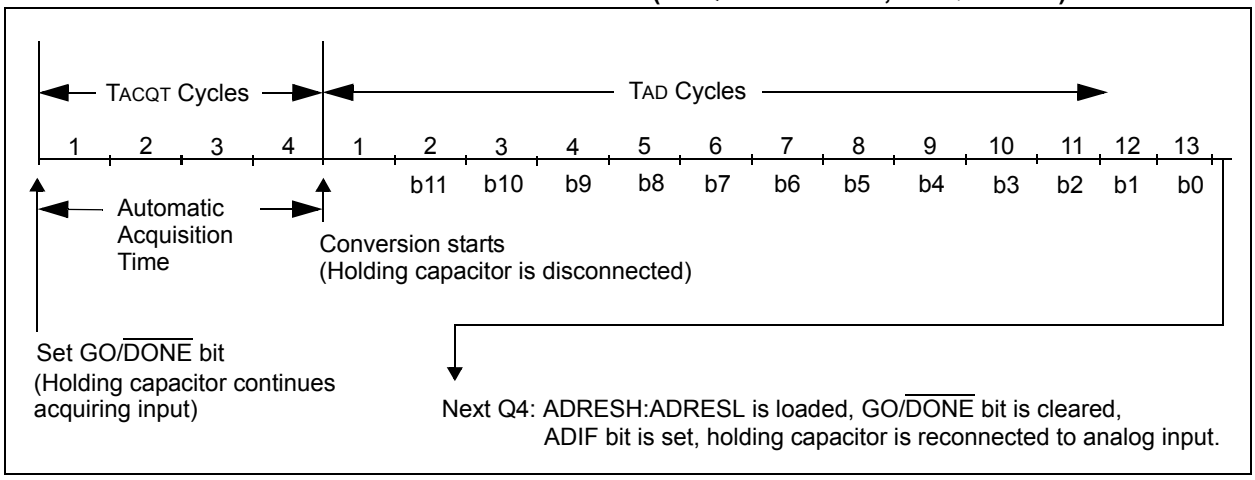



**FIGURE 23-7: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)**



## 23.8 Use of the Special Event Triggers

A/D conversion can be started by the Special Event Trigger of any of these modules:

- ECCP2 – Requires CCP2M<3:0> bits (CCP2CON<3:0>) set at '1011'
- CTMU – Requires the setting of the CTTRIG bit (CTMUCONH<0>)
- Timer1
- RTCC

To start an A/D conversion:

- The A/D module must be enabled (ADON = 1)
- The appropriate analog input channel is selected
- The minimum acquisition period is set one of these ways:
  - Timing provided by the user
  - Selection made of an appropriate TACQ time

With these conditions met, the trigger sets the GO/DONE bit and the A/D acquisition starts.

If the A/D module is not enabled (ADON = 0), the module ignores the Special Event Trigger.

| | |
|---|---|
| **Note:** | With an ECCP2 trigger, Timer1 or Timer 3 is cleared. The timers reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH:ADRESL to the desired location). If the A/D module is not enabled, the Special Event Trigger is ignored by the module, but the timer's counter resets. |

## 23.9 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 ADCON2 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 that the A/D RC clock 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 into Sleep mode. The IDLEN and SCS<1:0> bits in the OSCCON register must have already been cleared prior to starting the conversion.


© 2009-2018 Microchip Technology Inc.

**TABLE 23-2: REGISTERS ASSOCIATED WITH A/D MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR1 | PSPIF | ADIF | RC1IF | TX1IF | SSP1IF | TMR1GIF | TMR2IF | TMR1IF |
| PIE1 | PSPIE | ADIE | RC1IE | TX1IE | SSP1IE | TMR1GIE | TMR2IE | TMR1IE |
| IPR1 | PSPIP | ADIP | RC1IP | TX1IP | SSP1IP | TMR1GIP | TMR2IP | TMR1IP |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |
| ADRESH | A/D Result Register High Byte | | | | | | | |
| ADRESL | A/D Result Register Low Byte | | | | | | | |
| ADCON0 | — | CHS4 | CHS3 | CHS2 | CHS1 | CHS0 | GO/$\overline{\text{DONE}}$ | ADON |
| ADCON1 | TRIGSEL1 | TRIGSEL0 | VCFG1 | VCFG0 | VNCFG | CHSN2 | CHSN1 | CHSN0 |
| ADCON2 | ADFM | — | ACQT2 | ACQT1 | ACQT0 | ADCS2 | ADCS1 | ADCS0 |
| ANCON0 | ANSEL7 | ANSEL6 | ANSEL5 | ANSEL4 | ANSEL3 | ANSEL2 | ANSEL1 | ANSEL0 |
| ANCON1 | ANSEL15 | ANSEL14 | ANSEL13 | ANSEL12 | ANSEL11 | ANSEL10 | ANSEL9 | ANSEL8 |
| ANCON2 | ANSEL23 | ANSEL22 | ANSEL21 | ANSEL20 | ANSEL19 | ANSEL18 | ANSEL17 | ANSEL16 |
| CCP2CON | P2M1 | P2M0 | DC2B1 | DC2B0 | CCP2M3 | CCP2M2 | CCP2M1 | CCP2M0 |
| PORTA | RA7(2) | RA6(2) | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 |
| TRISA | TRISA7(2) | TRISA6(2) | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | — |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — |
| PORTG | — | — | RG5(3) | RG4 | RG3 | RG2 | RG1 | RG0 |
| TRISG | — | — | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 |
| PORTH(1) | RH7 | RH6 | RH5 | RH4 | RH3 | RH2 | RH1 | RH0 |
| TRISH(1) | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**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:** These bits are available only in certain oscillator modes, when the FOSC2 Configuration bit = 0. If that Configuration bit is cleared, this signal is not implemented.

**3:** This bit is available when Master Clear is disabled (MCLRE = 0). When MCLRE is set, the bit is unimplemented.

## 24.0 COMPARATOR MODULE

The analog comparator module contains three comparators that can be independently configured in a variety of ways. The inputs can be selected from the analog inputs and two Internal Reference Voltages. 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 24-1.

Key features of the module includes:

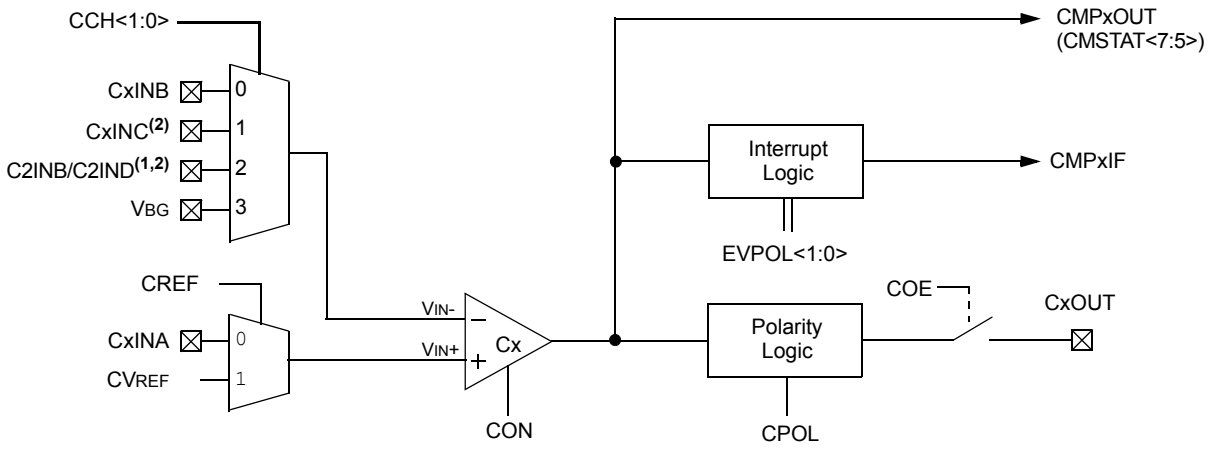
- Independent comparator control
- Programmable input configuration
- Output to both pin and register levels
- Programmable output polarity
- Independent interrupt generation for each comparator with configurable interrupt-on-change

## 24.1 Registers

The CMxCON registers (CM1CON, CM2CON and CM3CON) select the input and output configuration for each comparator, as well as the settings for interrupt generation (see Register 24-1).

The CMSTAT register (Register 24-2) provides the output results of the comparators. The bits in this register are read-only.

**FIGURE 24-1: COMPARATOR SIMPLIFIED BLOCK DIAGRAM**




**Note 1:** Comparators, 1 and 3, use C2INB as an input to the inverting terminal. Comparator 2 uses C2IND as an input to the inverted terminal.

**2:** C1INC, C2INC and C2IND are all unavailable for 64-pin devices (PIC18F6XK22).


© 2009-2018 Microchip Technology Inc.

**REGISTER 24-1: CMxCON: COMPARATOR CONTROL x 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 |
|---|---|---|---|---|---|---|---|
| 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 $CV_{REF}$ 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 $V_{BG}$
10 = Inverting input of comparator connects to C2INB or C2IND pin(2,3)
01 = Inverting input of comparator connects to CxINC pin
00 = Inverting input of comparator connects to CxINB pin

**Note 1:** The CMPxIF bit is automatically set any time this mode is selected and must be cleared by the application after the initial configuration.
**2:** Comparators, 1 and 3, use C2INB as an input to the inverting terminal. Comparator 2 uses C2IND.
**3:** C1INC, C2INC and C2IND are all unavailable for 64-pin devices (PIC18F6XK22).

**REGISTER 24-2: CMSTAT: COMPARATOR STATUS REGISTER**

| R-x | R-x | R-x | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| CMP3OUT | CMP2OUT | CMP1OUT | — | — | — | — | — |
| 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 **CMP<3:1>OUT:** Comparator x Status bits

If CPOL (CMxCON<5>)= 0 (noninverted polarity):

1 = Comparator x's VIN+ > VIN-

0 = Comparator x's VIN+ < VIN-

If CPOL = 1 (inverted polarity):

1 = Comparator x's VIN+ < VIN-

0 = Comparator x's VIN+ > VIN-

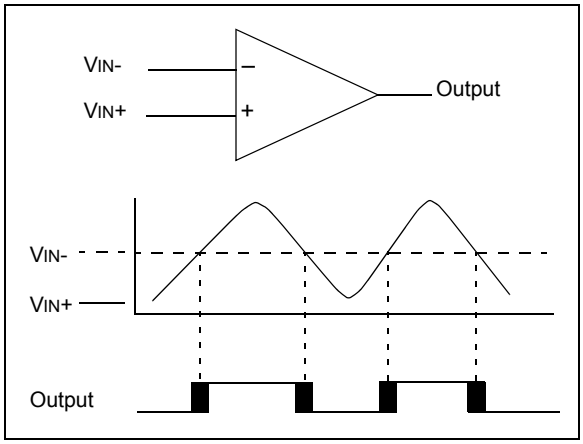
bit 4-0 **Unimplemented:** Read as '0'


© 2009-2018 Microchip Technology Inc.

## 24.2 Comparator Operation

A single comparator is shown in Figure 24-2, 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 24-2, represent the uncertainty due to input offsets and response time.

**FIGURE 24-2: SINGLE COMPARATOR**




## 24.3 Comparator Response Time

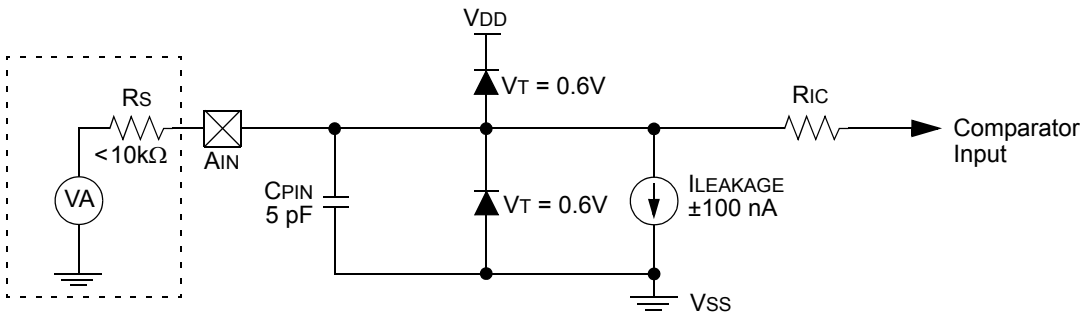
Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response to a comparator input change; otherwise, the maximum delay of the comparators should be used (see **Section 31.0 "Electrical Characteristics"**).

## 24.4 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 24-3. 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 24-3: 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 |

## 24.5 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 Reference Voltages.

All of the comparators allow a selection of the signal from pin, CxINA, or the voltage from the comparator reference ($CV_{REF}$) on the non-inverting channel. This is compared to either CxINB, CxINC, C2INB/C2IND or the microcontroller's fixed Internal Reference Voltage ($V_{BG}$, 1.024V nominal) on the inverting channel. The comparator inputs and outputs are tied to fixed I/O pins, defined in Table 24-1. The available comparator configurations and their corresponding bit settings are shown in Figure 24-4.

**TABLE 24-1: COMPARATOR INPUTS AND OUTPUTS**

| Comparator | Input or Output | I/O Pin |
|---|---|---|
| 1 | C1INA ($V_{IN+}$) | RF6 |
| | C1INB ($V_{IN-}$) | RF5 |
| | C1INC(1) ($V_{IN-}$) | RH6 |
| | C2INB ($V_{IN-}$) | RF3 |
| | C1OUT | RF2 |
| 2 | C2INA ($V_{IN+}$) | RF4 |
| | C2INB ($V_{IN-}$) | RF3 |
| | C2INC(1) ($V_{IN-}$) | RH4 |
| | C2IND(1) ($V_{IN-}$) | RH5 |
| | C2OUT | RF1 |
| 3 | C3INA ($V_{IN+}$) | RG2 |
| | C3INB ($V_{IN-}$) | RG3 |
| | C3INC ($V_{IN-}$) | RG4 |
| | C2INB ($V_{IN-}$) | RF3 |
| | C3OUT | RG1 |

**Note 1:** C1INC, C2INC and C2IND are all unavailable for 64-pin devices (PIC18F6XK22).

### 24.5.1 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 (VBG), to the comparator, $V_{IN-}$. Depending on the Comparator operating mode, either an external or Internal Reference Voltage may be used. For external analog pins that are unavailable in 64-pin devices (C1INC, C2INC and C2IND), the corresponding configurations that use them as inputs are unavailable.

The analog signal present at $V_{IN-}$ is compared to the signal at $V_{IN+}$ and the digital output of the comparator is adjusted accordingly.

The external reference is used when CREF = 0 (CMxCON<2>) and $V_{IN+}$ is connected to the CxINA pin. When external reference voltages are used, the comparator module can be configured to have the reference sources externally. The reference signal must be between $V_{SS}$ and $V_{DD}$, and can be applied to either pin of the comparator.

The comparator module also allows the selection of an internally generated reference voltage from the Comparator Voltage Reference ($CV_{REF}$) module. This module is described in more detail in **Section 25.0 "Comparator Voltage Reference Module"**. The reference from the comparator reference voltage module is only available when CREF = 1. In this mode, the Internal Reference Voltage is applied to the comparator's $V_{IN+}$ pin.

| **Note:** | The comparator input pin, selected by CCH<1:0>, must be configured as an input by setting both the corresponding TRISF, TRISG or TRISH bit and the corresponding ANSELx bit in the ANCONx register. |
|---|---|

### 24.5.2 COMPARATOR ENABLE AND OUTPUT SELECTION

The comparator outputs are read through the CMSTAT register. The CMSTAT<5> bit reads the Comparator 1 output, CMSTAT<6> reads Comparator 2 output and CMSTAT<7> reads Comparator 3 output. These bits are read-only.

The comparator outputs may also be directly output to the RF2, RF1 and RG1 I/O pins by setting the COE bit (CMxCON<6>). When enabled, multiplexers in the output path of the pins switch to the output of the comparator. While in this mode, the TRISF<2:1> and TRISG<1> bits still function as the digital output enable bits for the RF2, RF1 and RG1 pins.

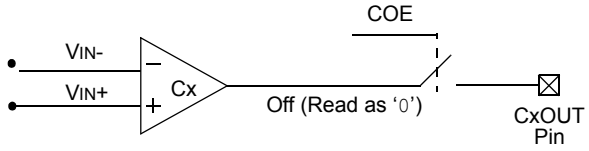
By default, the comparator's output is at logic high whenever the voltage on $V_{IN+}$ is greater than on $V_{IN-}$. The polarity of the comparator outputs can be inverted using the CPOL bit (CMxCON<5>).

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 24.2 "Comparator Operation"**.

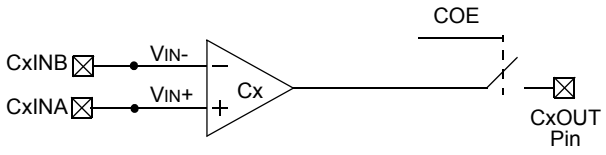

© 2009-2018 Microchip Technology Inc.

**FIGURE 24-4: COMPARATOR CONFIGURATIONS**

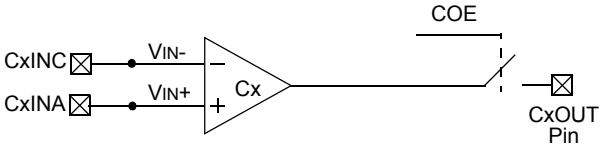
**Comparator Off**
**CON = 0, CREF = x, CCH<1:0> = xx**



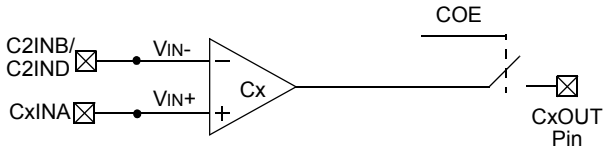

**Comparator CxINB > CxINA Compare**
**CON = 1, CREF = 0, CCH<1:0> = 00**



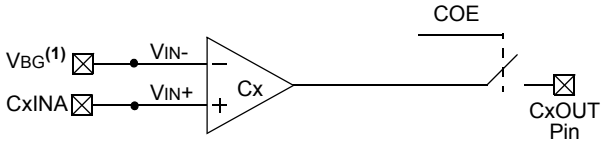

**Comparator CxINC > CxINA Compare(2,3)**
**CON = 1, CREF = 0, CCH<1:0> = 01**



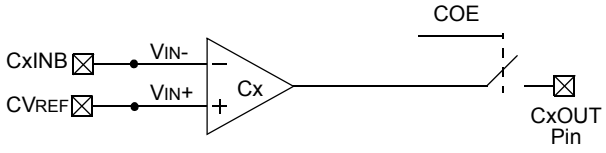

**Comparator CxIND > CxINA Compare(3)**
**CON = 1, CREF = 0, CCH<1:0> = 10**



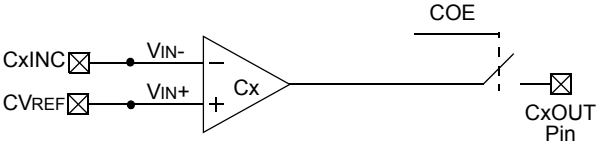

**Comparator VIRV > CxINA Compare**
**CON = 1, CREF = 0, CCH<1:0> = 11**



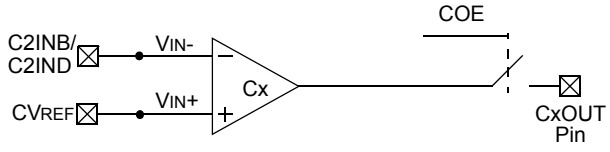

**Comparator CxINB > CVREF Compare**
**CON = 1, CREF = 1, CCH<1:0> = 00**



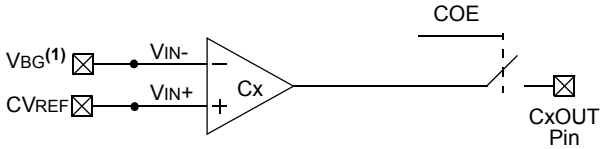

**Comparator CxINC > CVREF Compare(2,3)**
**CON = 1, CREF = 1, CCH<1:0> = 01**




**Comparator CxIND > CVREF Compare(3)**
**CON = 1, CREF = 1, CCH<1:0> = 10**




**Comparator VIRV > CVREF Compare**
**CON = 1, CREF = 1, CCH<1:0> = 11**




**Note 1:** VBG is the Internal Reference Voltage (1.024V nominal).
**2:** Configuration is unavailable for CM1CON on 64-pin devices (PIC18F6XK22).
**3:** Configuration is unavailable for CM2CON on 64-pin devices (PIC18F6XK22).

## 24.6 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. When 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<7:5>, to determine the actual change that occurred.

The CMPxIF bits (PIR6<2:0>) are the Comparator Interrupt Flags. The CMPxIF 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 24-2 shows the interrupt generation with respect to comparator input voltages and EVPOL bit settings.

Both the CMPxIE bits (PIE6<2:0>) 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 CMPxIF bits will still be set if an interrupt condition occurs.

A simplified diagram of the interrupt section is shown in Figure 24-3.

**Note:** The CMPxIF bits will not be set when EVPOL<1:0> = 00.

**TABLE 24-2: 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 |


© 2009-2018 Microchip Technology Inc.

## 24.7 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.

## 24.8 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 24-3: REGISTERS ASSOCIATED WITH COMPARATOR MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR6 | — | — | — | EEIF | — | CMP3IF | CMP2IF | CMP1IF |
| PIE6 | — | — | — | EEIE | — | CMP3IE | CMP2IE | CMP1IE |
| IPR6 | — | — | — | EEIP | — | CMP3IP | CMP2IP | CMP1IP |
| CM1CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 |
| CM2CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 |
| CM3CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 |
| CVRCON | CVREN | CVROE | CVRSS | CVR4 | CVR3 | CVR2 | CVR1 | CVR0 |
| CMSTAT | CMP3OUT | CMP2OUT | CMP1OUT | — | — | — | — | — |
| PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | — |
| LATF | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | — |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — |
| PORTG | — | — | RG5(1) | RG4 | RG3 | RG2 | RG1 | RG0 |
| LATG | — | — | — | LATG4 | LATG3 | LATG2 | LATG1 | LATG0 |
| TRISG | — | — | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 |
| PORTH | RH7 | RH6 | RH5 | RH4 | RH3 | RH2 | RH1 | RH0 |
| LATH | LATH7 | LATH6 | LATH5 | LATH4 | LATH3 | LATH2 | LATH1 | LATH0 |
| TRISH | TRISH7 | TRISH6 | TRISH5 | TRISH4 | TRISH3 | TRISH2 | TRISH1 | TRISH0 |
| ANCON0 | ANSEL7 | ANSEL6 | ANSEL5 | ANSEL4 | ANSEL3 | ANSEL2 | ANSEL1 | ANSEL0 |
| ANCON1 | ANSEL15 | ANSEL14 | ANSEL13 | ANSEL12 | ANSEL11 | ANSEL10 | ANSEL9 | ANSEL8 |
| ANCON2 | ANSEL23 | ANSEL22 | ANSEL21 | ANSEL20 | ANSEL19 | ANSEL18 | ANSEL17 | ANSEL16 |
| PMD0 | CCP3MD | CCP2MD | CCP1MD | UART2MD | UART1MD | SSP2MD | SSP1MD | ADCMD |

**Legend:** — = unimplemented, read as '0'.

**Note 1:** Bit is available when Master Clear is disabled (MCLRE = 0). When MCLRE is set, the bit is unimplemented.

## 25.0 COMPARATOR VOLTAGE REFERENCE MODULE

The comparator voltage reference is a 32-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 25-1. The resistor ladder is segmented to provide a range 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.

### 25.1 Configuring the Comparator Voltage Reference

The comparator voltage reference module is controlled through the CVRCON register (Register 25-1). The comparator voltage reference provides a range of output voltage with 32 levels.

The CVR<4:0> selection bits (CVRCON<4:0>) offer a range of output voltages. Equation 25-1 shows the how the comparator voltage reference is computed.

**EQUATION 25-1:**

If CVRSS = 1:

$$CVREF = (VREF-) + (CVR<4:0>/32) \bullet (VREF+ - VREF-)$$

If CVRSS = 0:

$$CVREF = (AVSS) + (CVR<4:0>/32) \bullet (AVDD - AVSS)$$

The comparator reference supply voltage can come from either VDD and VSS, or the external VREF+ and VREF- that are multiplexed with RA3 and RA2. The voltage source is selected by the CVRSS bit (CVRCON<5>).

The settling time of the comparator voltage reference must be considered when changing the CVREF output (see Table 31-2 in **Section 31.0 "Electrical Characteristics"**).

**REGISTER 25-1: 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 | CVRSS | CVR4 | 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

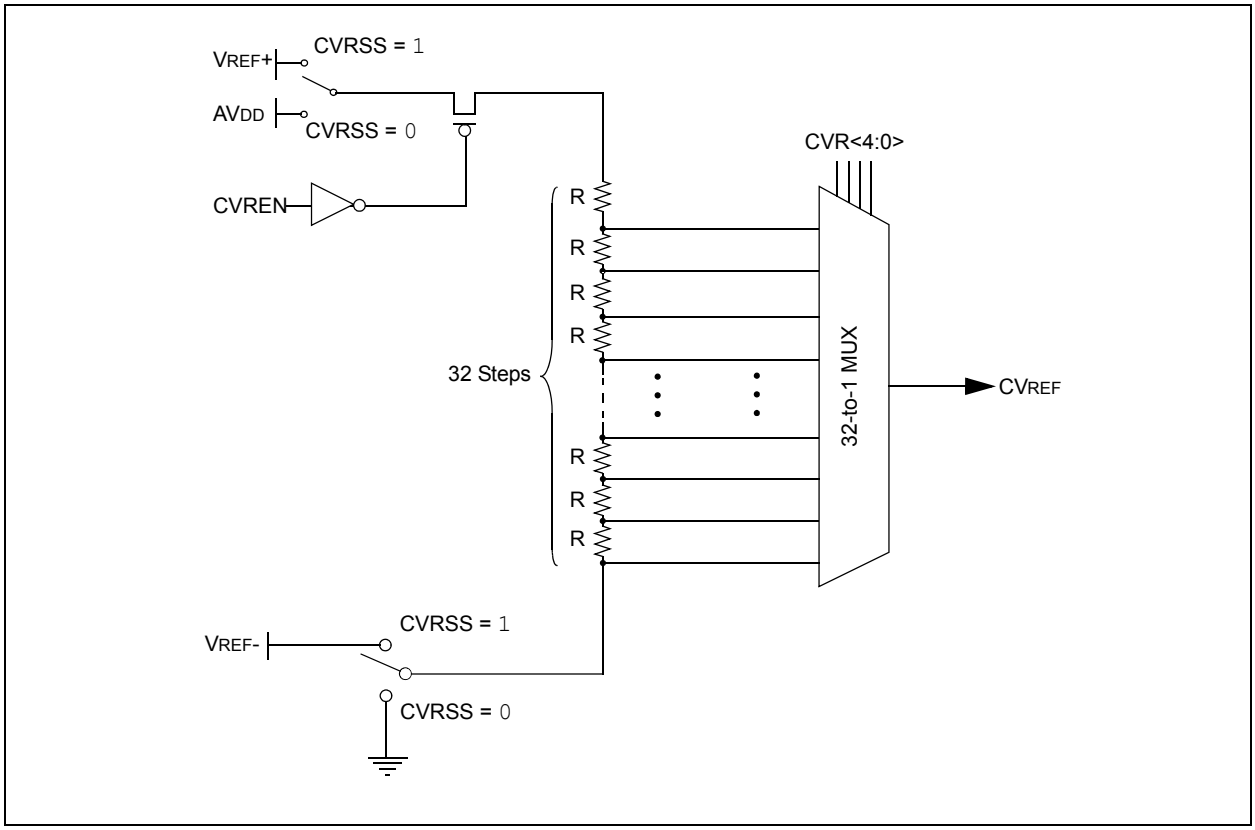
bit 6 **CVROE:** Comparator VREF Output Enable bit
1 = CVREF voltage level is output on CVREF pin
0 = CVREF voltage level is disconnected from CVREF pin

bit 5 **CVRSS:** Comparator VREF Source Selection bit
1 = Comparator reference source, CVRSRC = VREF+ – VREF-
0 = Comparator reference source, CVRSRC = AVDD – AVSS

bit 4-0 **CVR<4:0>:** Comparator VREF Value Selection 0 ≤ CVR<4:0> ≤ 31 bits
When CVRSS = 1:
CVREF = (VREF-) + (CVR<4:0>/32) • (VREF+ – VREF-)
When CVRSS = 0:
CVREF = (AVSS) + (CVR<4:0>/32) • (AVDD – AVSS)


© 2009-2018 Microchip Technology Inc.

**FIGURE 25-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM**




## 25.2 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 25-1) 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 31.0 "Electrical Characteristics"**.

## 25.3 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.

## 25.4 Effects of a Reset

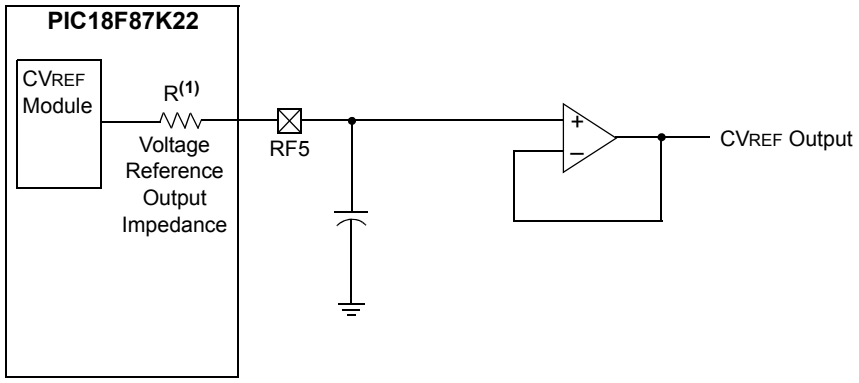
A device Reset disables the voltage reference by clearing bit, CVREN (CVRCON<7>). This Reset also disconnects the reference from the RF5 pin by clearing bit, CVROE (CVRCON<6>).

## 25.5 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 RF5 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 25-2 shows an example buffering technique.

**FIGURE 25-2: COMPARATOR VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE**




**Note 1:** R is dependent upon the Voltage Reference Configuration bits, CVRCON<3:0> and CVRCON<5>.

**TABLE 25-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| CVRCON | CVREN | CVROE | CVRSS | CVR4 | CVR3 | CVR2 | CVR1 | CVR0 |
| CM1CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 |
| CM2CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 |
| CM3CON | CON | COE | CPOL | EVPOL1 | EVPOL0 | CREF | CCH1 | CCH0 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | — |
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| ANCON0 | ANSEL7 | ANSEL6 | ANSEL5 | ANSEL4 | ANSEL3 | ANSEL2 | ANSEL1 | ANSEL0 |
| ANCON1 | ANSEL15 | ANSEL14 | ANSEL13 | ANSEL12 | ANSEL11 | ANSEL10 | ANSEL9 | ANSEL8 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are not used with the comparator voltage reference.


© 2009-2018 Microchip Technology Inc.

## 26.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

The PIC18F87K22 family of devices has a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that sets both a device voltage trip point and the direction of change from that point. If the device experiences an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution branches to the interrupt vector address and the software responds to the interrupt.

The High/Low-Voltage Detect Control register (Register 26-1) completely controls the operation of the HLVD module. This allows the circuitry to be "turned off" by the user under software control, which minimizes the current consumption for the device.

The module's block diagram is shown in Figure 26-1.

**REGISTER 26-1: HLVDCON: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER**

| R/W-0 | R-0 | R-0 | R/W-0 | R/W-0 | R/W-1 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| VDIRMAG | BGVST | IRVST | HLVDEN | HLVDL3(1) | HLVDL2(1) | HLVDL1(1) | HLVDL0(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 **VDIRMAG:** Voltage Direction Magnitude Select bit
1 = Event occurs when voltage equals or exceeds trip point (HLVDL<3:0>)
0 = Event occurs when voltage equals or falls below trip point (HLVDL<3:0>)

bit 6 **BGVST:** Band Gap Reference Voltages Stable Status Flag bit
1 = Internal band gap voltage references are stable
0 = Internal band gap voltage references are not stable

bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit
1 = Indicates that the voltage detect logic will generate the interrupt flag at the specified voltage range
0 = Indicates that the voltage detect logic will not generate the interrupt flag at the specified voltage range and the HLVD interrupt should not be enabled

bit 4 **HLVDEN:** High/Low-Voltage Detect Power Enable bit
1 = HLVD is enabled
0 = HLVD is disabled

bit 3-0 **HLVDL<3:0>:** Voltage Detection Limit bits(1)
1111 = External analog input is used (input comes from the HLVDIN pin)
1110 = Maximum setting
.
.
.
0000 = Minimum setting

**Note 1:** For the electrical specifications, see Parameter D420.

The module is enabled by setting the HLVDEN bit (HLVDCON<4>). Each time the HLVD module is enabled, the circuitry requires some time to stabilize. The IRVST bit (HLVDCON<5>) is a read-only bit used to indicate when the circuit is stable. The module can only generate an interrupt after the circuit is stable and IRVST is set.

The VDIRMAG bit (HLVDCON<7>) determines the overall operation of the module. When VDIRMAG is cleared, the module monitors for drops in VDD below a predetermined set point. When the bit is set, the module monitors for rises in VDD above the set point.

## 26.1 Operation

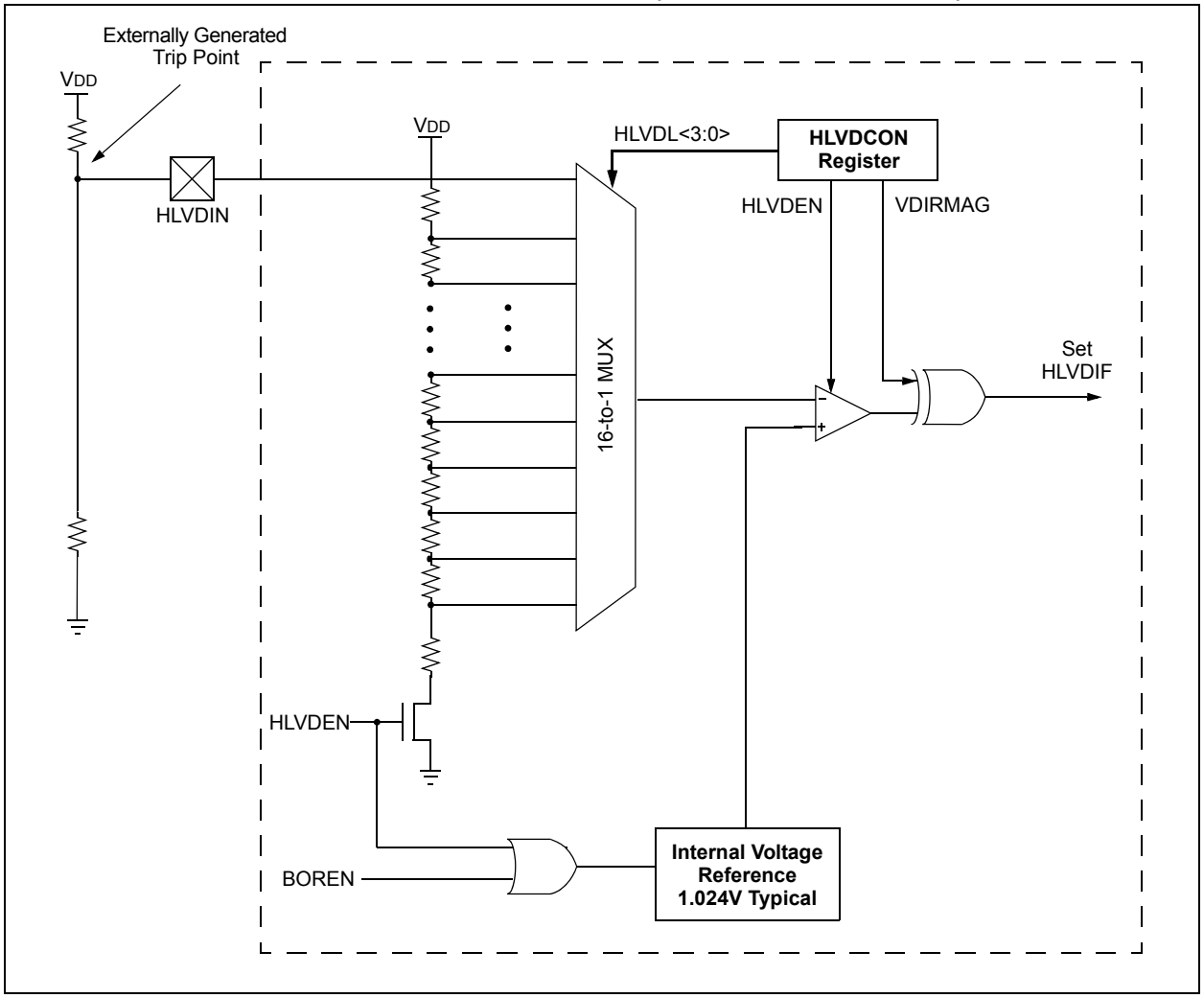
When the HLVD module is enabled, a comparator uses an internally generated voltage reference as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The "trip point" voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module.

When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable to any of 16 values. The trip point is selected by programming the HLVDL<3:0> bits (HLVDCON<3:0>).

The HLVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits, HLVDL<3:0>, are set to '1111'. In this state, the comparator input is multiplexed from the external input pin, HLVDIN. This gives users the flexibility of configuring the High/Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

**FIGURE 26-1: HLVD MODULE BLOCK DIAGRAM (WITH EXTERNAL INPUT)**





© 2009-2018 Microchip Technology Inc.

## 26.2 HLVD Setup

To set up the HLVD module:

1. Select the desired HLVD trip point by writing the value to the HLVDL<3:0> bits.
2. Set the VDIRMAG bit to detect high voltage (VDIRMAG = 1) or low voltage (VDIRMAG = 0).
3. Enable the HLVD module by setting the HLVDEN bit.
4. Clear the HLVD interrupt flag (PIR2<2>), which may have been set from a previous interrupt.
5. If interrupts are desired, enable the HLVD interrupt by setting the HLVDIE and GIE bits (PIE2<2> and INTCON<7>, respectively).

   An interrupt will not be generated until the IRVST bit is set.

| **Note:** | Before changing any module settings (VDIRMAG, LVDL<3:0>), first disable the module (LVDEN = 0), make the changes and re-enable the module. This prevents the generation of false HLVD events. |
|---|---|

## 26.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled and consume static current. The total current consumption, when enabled, is specified in electrical specification Parameter D022B (Table 31-14).

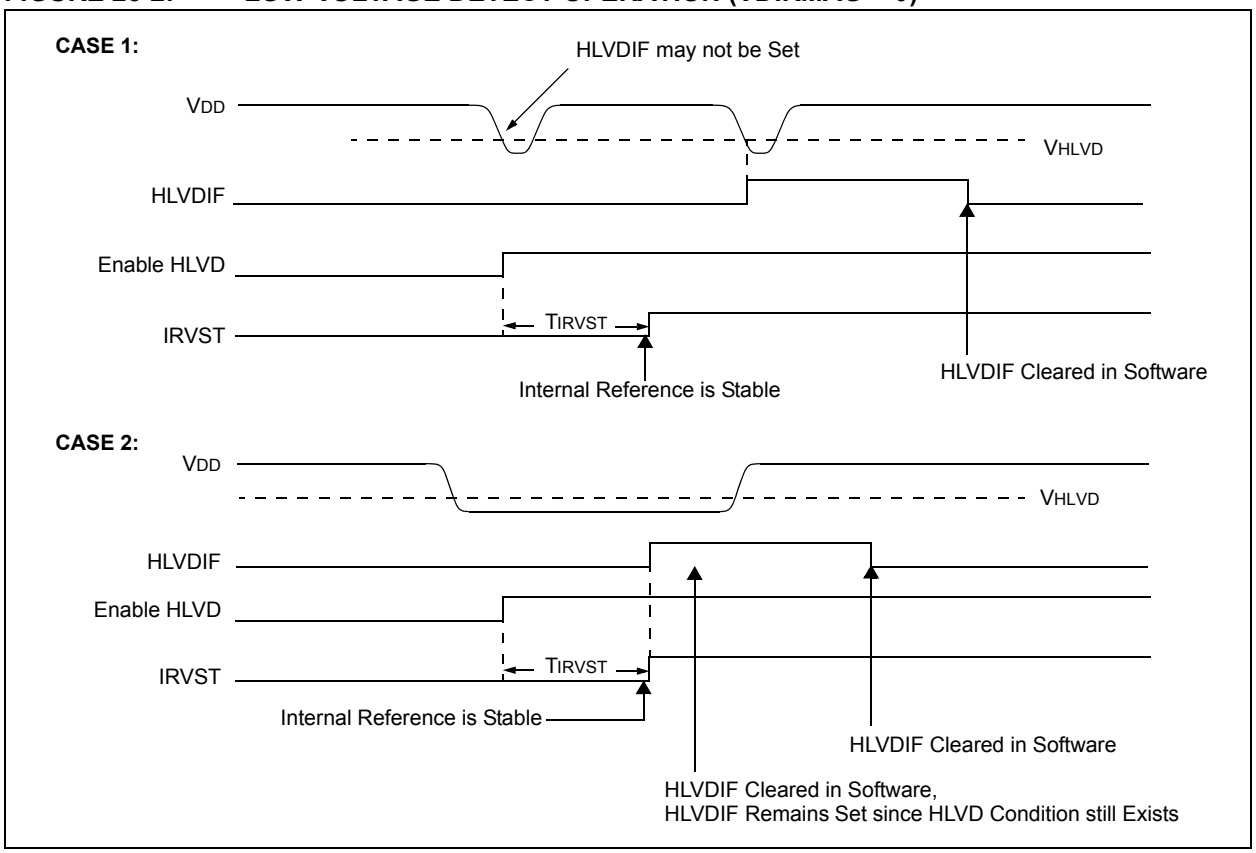
Depending on the application, the HLVD module does not need to operate constantly. To reduce current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After such a check, the module could be disabled.

## 26.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification Parameter 37 (**Section 31.0 "Electrical Characteristics"**), may be used by other internal circuitry, such as the programmable Brown-out Reset. If the HLVD or other circuits using the voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time, TIRVST, is an interval that is independent of device clock speed. It is specified in electrical specification Parameter 37 (Table 31-14).

The HLVD interrupt flag is not enabled until TIRVST has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval (see Figure 26-2 or Figure 26-3).

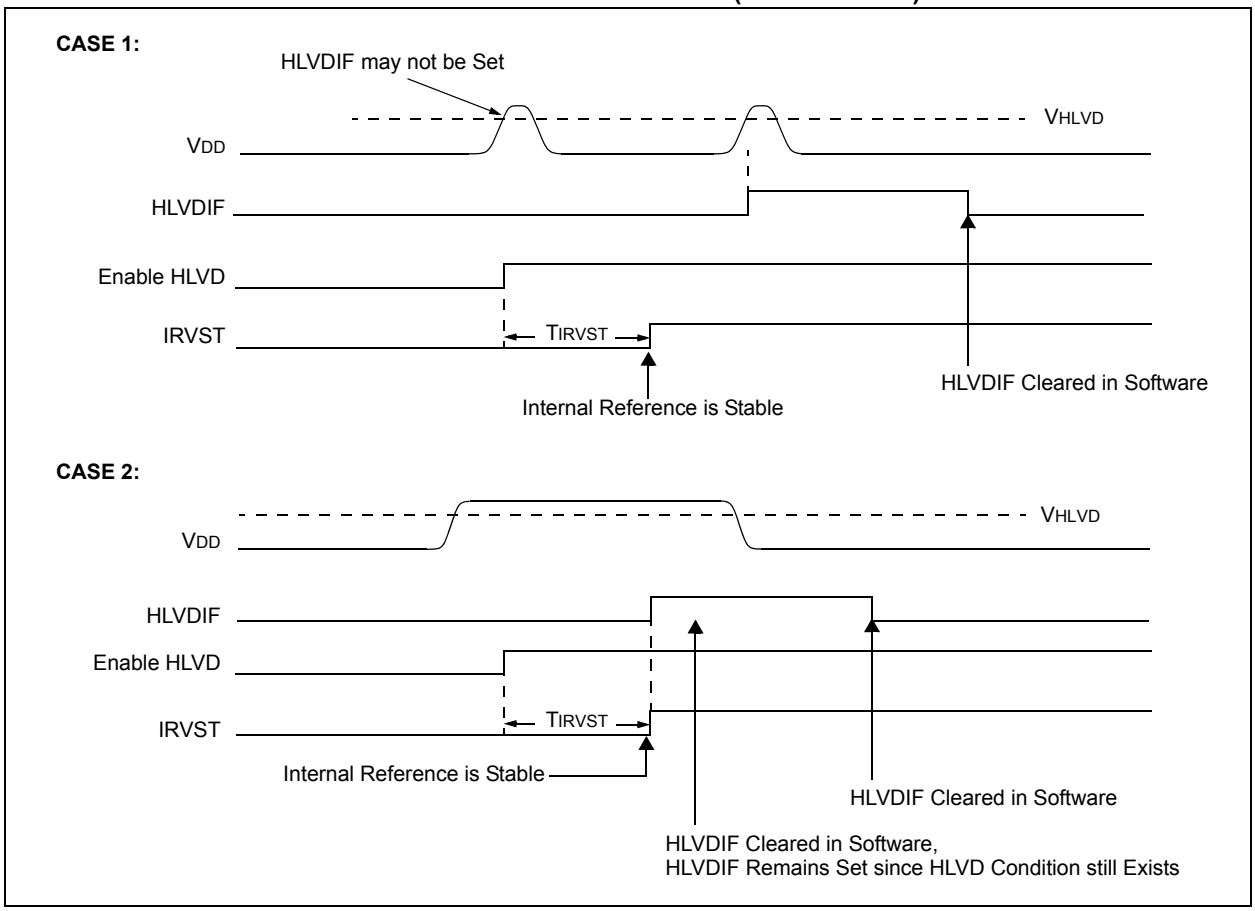
**FIGURE 26-2: LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 0)**





© 2009-2018 Microchip Technology Inc.

**FIGURE 26-3: HIGH-VOLTAGE DETECT OPERATION (VDIRMAG = 1)**

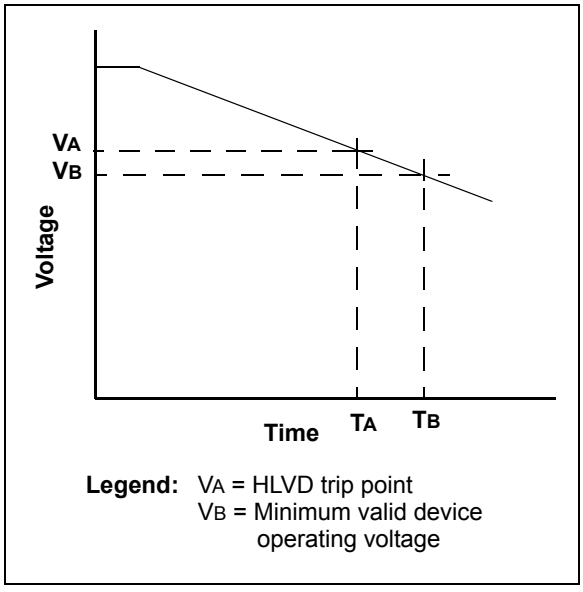



## 26.5 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. For example, the HLVD module could be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach would indicate a High-Voltage Detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 26-4 shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, VA, the HLVD logic generates an interrupt at time, TA. The interrupt could cause the execution of an Interrupt Service Routine (ISR), which would allow the application to perform “housekeeping tasks” and a controlled shutdown before the device voltage exits the valid operating range at TB. This would give the application a time window, represented by the difference between TA and TB, to safely exit.

**FIGURE 26-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION**



## 26.6 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the HLVDIF bit will be set and the device will wake up from Sleep. Device execution will continue from the interrupt vector address if interrupts have been globally enabled.

## 26.7 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

**TABLE 26-1: REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| HLVDCON | VDIRMAG | BGVST | IRVST | HLVDEN | HLVDL3 | HLVDL2 | HLVDL1 | HLVDL0 |
| INTCON | GIE/GIEH | PEIE/GIEL | TMR0IE | INT0IE | RBIE | TMR0IF | INT0IF | RBIF |
| PIR2 | OSCFIF | — | SSP2IF | BLC2IF | BCL1IF | HLVDIF | TMR3IF | TMR3GIF |
| PIE2 | OSCFIE | — | SSP2IE | BLC2IE | BCL1IE | HLVDIE | TMR3IE | TMR3GIE |
| IPR2 | OSCFIP | — | SSP2IP | BLC2IP | BCL1IP | HLVDIP | TMR3IP | TMR3GIP |
| TRISA | TRISA7(1) | TRISA6(1) | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| ANCON0 | ANSEL7 | ANSEL6 | ANSEL5 | ANSEL4 | ANSEL3 | ANSEL2 | ANSEL1 | ANSEL0 |

**Legend:** — = unimplemented, read as '0'. Shaded cells are unused by the HLVD module.

**Note 1:** PORTA<7:6> and their direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as '0'.


© 2009-2018 Microchip Technology Inc.

## 27.0 CHARGE TIME MEASUREMENT UNIT (CTMU)

The Charge Time Measurement Unit (CTMU) is a flexible analog module that provides accurate differential time measurement between pulse sources, as well as asynchronous pulse generation. By working with other on-chip analog modules, the CTMU can precisely measure time, capacitance and relative changes in capacitance or generate output pulses with a specific time delay. The CTMU is ideal for interfacing with capacitive-based sensors.

The module includes these key features:

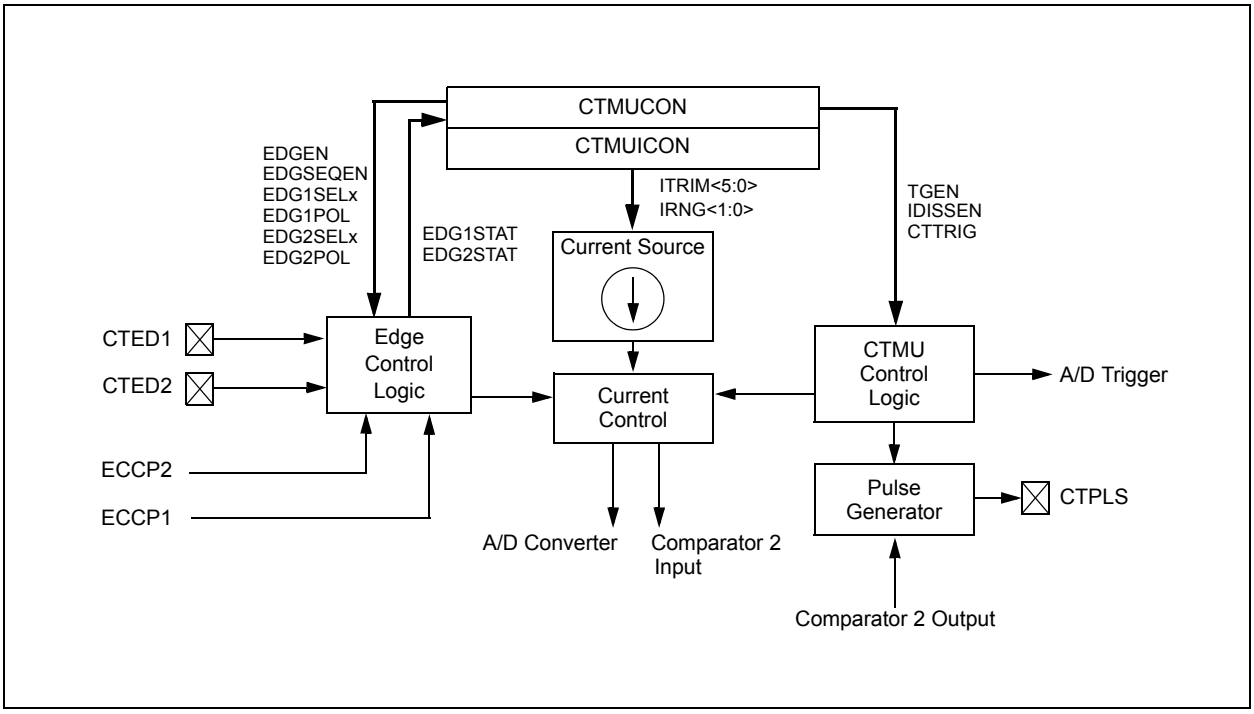
- Up to 24 channels available for capacitive or time measurement input
- On-chip precision current source
- Four-edge input trigger sources
- Polarity control for each edge source
- Control of edge sequence
- Control of response to edges
- Time measurement resolution of one nanosecond
- High-precision time measurement
- Time delay of external or internal signal asynchronous to system clock
- Accurate current source suitable for capacitive measurement

The CTMU works in conjunction with the A/D Converter to provide up to 24 channels for time or charge measurement, depending on the specific device and the number of A/D channels available. When configured for time delay, the CTMU is connected to one of the analog comparators. The level-sensitive input edge sources can be selected from four sources: two external inputs or the ECCP1/ECCP2 Special Event Triggers.

The CTMU special event can trigger the Analog-to-Digital Converter module.

Figure 27-1 provides a block diagram of the CTMU.

**FIGURE 27-1: CTMU BLOCK DIAGRAM**



## 27.1 CTMU Registers

The control registers for the CTMU are:

- CTMUCONH
- CTMUCONL
- CTMUICON

The CTMUCONH and CTMUCONL registers (Register 27-1 and Register 27-2) contain control bits for configuring the CTMU module edge source selection, edge source polarity selection, edge sequencing, A/D trigger, analog circuit capacitor discharge and enables. The CTMUICON register (Register 27-3) has bits for selecting the current source range and current source trim.

**REGISTER 27-1: CTMUCONH: CTMU CONTROL HIGH REGISTER**

| R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| CTMUEN | — | CTMUSIDL | TGEN | EDGEN | EDGSEQEN | IDISSEN | CTTRIG |
| 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 **CTMUEN:** CTMU Enable bit
1 = Module is enabled
0 = Module is disabled

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

bit 5 **CTMUSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 4 **TGEN:** Time Generation Enable bit
1 = Enables edge delay generation
0 = Disables edge delay generation

bit 3 **EDGEN:** Edge Enable bit
1 = Edges are not blocked
0 = Edges are blocked

bit 2 **EDGSEQEN:** Edge Sequence Enable bit
1 = Edge 1 event must occur before Edge 2 event can occur
0 = No edge sequence is needed

bit 1 **IDISSEN:** Analog Current Source Control bit
1 = Analog current source output is grounded
0 = Analog current source output is not grounded

bit 0 **CTTRIG:** Trigger Control bit
1 = Trigger output is enabled
0 = Trigger output is disabled


© 2009-2018 Microchip Technology Inc.

**REGISTER 27-2: CTMUCONL: CTMU CONTROL LOW 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 |
|---|---|---|---|---|---|---|---|
| EDG2POL | EDG2SEL1 | EDG2SEL0 | EDG1POL | EDG1SEL1 | EDG1SEL0 | EDG2STAT | EDG1STAT |
| 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 **EDG2POL:** Edge 2 Polarity Select bit
1 = Edge 2 is programmed for a positive edge response
0 = Edge 2 is programmed for a negative edge response

bit 6-5 **EDG2SEL<1:0>:** Edge 2 Source Select bits
11 = CTED1 pin
10 = CTED2 pin
01 = ECCP1 Special Event Trigger
00 = ECCP2 Special Event Trigger

bit 4 **EDG1POL:** Edge 1 Polarity Select bit
1 = Edge 1 is programmed for a positive edge response
0 = Edge 1 is programmed for a negative edge response

bit 3-2 **EDG1SEL<1:0>:** Edge 1 Source Select bits
11 = CTED1 pin
10 = CTED2 pin
01 = ECCP1 Special Event Trigger
00 = ECCP2 Special Event Trigger

bit 1 **EDG2STAT:** Edge 2 Status bit
1 = Edge 2 event has occurred
0 = Edge 2 event has not occurred

bit 0 **EDG1STAT:** Edge 1 Status bit
1 = Edge 1 event has occurred
0 = Edge 1 event has not occurred

**REGISTER 27-3: CTMUICON: CTMU CURRENT 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 |
|---|---|---|---|---|---|---|---|
| ITRIM5 | ITRIM4 | ITRIM3 | ITRIM2 | ITRIM1 | ITRIM0 | IRNG1 | IRNG0 |
| 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 **ITRIM<5:0>:** Current Source Trim bits

011111 = Maximum positive change from nominal current
011110
.
.
.
000001 = Minimum positive change from nominal current
000000 = Nominal current output specified by IRNG<1:0>
111111 = Minimum negative change from nominal current
.
.
.
100010
100001 = Maximum negative change from nominal current

bit 1-0 **IRNG<1:0>:** Current Source Range Select bits

11 = 100 x Base Current
10 = 10 x Base Current
01 = Base Current Level (0.55 μA nominal)
00 = Current Source Disabled


© 2009-2018 Microchip Technology Inc.

## 27.2 CTMU Operation

The CTMU works by using a fixed current source to charge a circuit. The type of circuit depends on the type of measurement being made.

In the case of charge measurement, the current is fixed and the amount of time the current is applied to the circuit is fixed. The amount of voltage read by the A/D becomes a measurement of the circuit's capacitance.

In the case of time measurement, the current, as well as the capacitance of the circuit, is fixed. In this case, the voltage read by the A/D is representative of the amount of time elapsed from the time the current source starts and stops charging the circuit.

If the CTMU is being used as a time delay, both capacitance and current source are fixed, as well as the voltage supplied to the comparator circuit. The delay of a signal is determined by the amount of time it takes the voltage to charge to the comparator threshold voltage.

### 27.2.1 THEORY OF OPERATION

The operation of the CTMU is based on the equation for charge:

$$C = I \bullet \frac{dV}{dT}$$

More simply, the amount of charge measured in coulombs in a circuit is defined as current in amperes (I) multiplied by the amount of time in seconds that the current flows (t). Charge is also defined as the capacitance in farads (C) multiplied by the voltage of the circuit (V). It follows that:

$$I \bullet t = C \bullet V$$

The CTMU module provides a constant, known current source. The A/D Converter is used to measure (V) in the equation, leaving two unknowns: capacitance (C) and time (t). The above equation can be used to calculate capacitance or time, by either the relationship using the known fixed capacitance of the circuit:

$$t = (C \bullet V)/I$$

or by:

$$C = (I \bullet t)/V$$

using a fixed time that the current source is applied to the circuit.

### 27.2.2 CURRENT SOURCE

At the heart of the CTMU is a precision current source, designed to provide a constant reference for measurements. The level of current is user-selectable across three ranges or a total of two orders of magnitude, with the ability to trim the output in ±2% increments (nominal). The current range is selected by the IRNG<1:0> bits (CTMUICON<1:0>), with a value of '00' representing the lowest range.

Current trim is provided by the ITRIM<5:0> bits (CTMUICON<7:2>). These six bits allow trimming of the current source in steps of approximately 2% per step. Half of the range adjusts the current source positively and the other half reduces the current source. A value of '000000' is the neutral position (no change). A value of '100000' is the maximum negative adjustment (approximately -62%) and '011111' is the maximum positive adjustment (approximately +62%).

### 27.2.3 EDGE SELECTION AND CONTROL

CTMU measurements are controlled by edge events occurring on the module's two input channels. Each channel, referred to as Edge 1 and Edge 2, can be configured to receive input pulses from one of the edge input pins (CTED1 and CTED2) or CCPx Special Event Triggers. The input channels are level-sensitive, responding to the instantaneous level on the channel rather than a transition between levels. The inputs are selected using the EDG1SEL and EDG2SEL bit pairs (CTMUCONL<3:2, 6:5>).

In addition to source, each channel can be configured for event polarity using the EDGE2POL and EDGE1POL bits (CTMUCONL<7,4>). The input channels can also be filtered for an edge event sequence (Edge 1 occurring before Edge 2) by setting the EDGSEQEN bit (CTMUCONH<2>).

### 27.2.4 EDGE STATUS

The CTMUCON register also contains two status bits: EDG2STAT and EDG1STAT (CTMUCONL<1:0>). Their primary function is to show if an edge response has occurred on the corresponding channel. The CTMU automatically sets a particular bit when an edge response is detected on its channel. The level-sensitive nature of the input channels also means that the status bits become set immediately if the channel's configuration is changed and matches the channel's current state.

The module uses the edge status bits to control the current source output to external analog modules (such as the A/D Converter). Current is only supplied to external modules when only one (not both) of the status bits is set. Current is shut off when both bits are either set or cleared. This allows the CTMU to measure current only during the interval between edges. After both status bits are set, it is necessary to clear them before another measurement is taken. Both bits should be cleared simultaneously, if possible, to avoid re-enabling the CTMU current source.

In addition to being set by the CTMU hardware, the edge status bits can also be set by software. This permits a user application to manually enable or disable the current source. Setting either (but not both) of the bits enables the current source. Setting or clearing both bits at once disables the source.

© 2009-2018 Microchip Technology Inc.

### 27.2.5 INTERRUPTS

The CTMU sets its interrupt flag (PIR3<3>) whenever the current source is enabled, then disabled. An interrupt is generated only if the corresponding interrupt enable bit (PIE3<3>) is also set. If edge sequencing is not enabled (i.e., Edge 1 must occur before Edge 2), it is necessary to monitor the edge status bits and determine which edge occurred last and caused the interrupt.

## 27.3 CTMU Module Initialization

The following sequence is a general guideline used to initialize the CTMU module:

1. Select the current source range using the IRNGx bits (CTMUICON<1:0>).
2. Adjust the current source trim using the ITRIMx bits (CTMUICON<7:2>).
3. Configure the edge input sources for Edge 1 and Edge 2 by setting the EDG1SEL and EDG2SEL bits (CTMUCONL<3:2> and <6:5>, respectively).
4. Configure the input polarities for the edge inputs using the EDG2POL and EDG1POL bits (CTMUCONL<7,4>).

   The default configuration is for negative edge polarity (high-to-low transitions).
5. Enable edge sequencing using the EDGSEQEN bit (CTMUCONH<2>).

   By default, edge sequencing is disabled.
6. Select the operating mode (Measurement or Time Delay) with the TGEN bit.

   The default mode is Time/Capacitance Measurement.
7. Configure the module to automatically trigger an A/D conversion when the second edge event has occurred, using the CTTRIG bit (CTMUCONH<0>).

   The conversion trigger is disabled by default.
8. Discharge the connected circuit by setting the IDISSEN bit (CTMUCONH<1>).
9. After waiting a sufficient time for the circuit to discharge, clear IDISSEN.
10. Disable the module by clearing the CTMUEN bit (CTMUCONH<7>).
11. Clear the Edge Status bits, EDG2STAT and EDG1STAT (CTMUCONL<1:0>).
12. Enable both edge inputs by setting the EDGEN bit (CTMUCONH<3>).
13. Enable the module by setting the CTMUEN bit.

Depending on the type of measurement or pulse generation being performed, one or more additional modules may also need to be initialized and configured with the CTMU module:

- Edge Source Generation: In addition to the external edge input pins, CCPx Special Event Triggers can be used as edge sources for the CTMU.
- Capacitance or Time Measurement: The CTMU module uses the A/D Converter to measure the voltage across a capacitor that is connected to one of the analog input channels.
- Pulse Generation: When generating system clock independent, output pulses, the CTMU module uses Comparator 2 and the associated comparator voltage reference.

## 27.4 Calibrating the CTMU Module

The CTMU requires calibration for precise measurements of capacitance and time, as well as for accurate time delay. If the application only requires measurement of a relative change in capacitance or time, calibration is usually not necessary. An example of a less precise application is a capacitive touch switch, in which the touch circuit has a baseline capacitance and the added capacitance of the human body changes the overall capacitance of a circuit.

If actual capacitance or time measurement is required, two hardware calibrations must take place:

- The current source needs calibration to set it to a precise current.
- The circuit being measured needs calibration to measure or nullify any capacitance other than that to be measured.

### 27.4.1 CURRENT SOURCE CALIBRATION

The current source on board the CTMU module has a range of ±60% nominal for each of three current ranges. For precise measurements, it is possible to measure and adjust this current source by placing a high-precision resistor, RCAL, onto an unused analog channel. An example circuit is shown in Figure 27-2.

To measure the current source:

1. Initialize the A/D Converter.
2. Initialize the CTMU.
3. Enable the current source by setting EDG1STAT (CTMUCONL<0>).
4. Issue the settling time delay.
5. Perform the A/D conversion.
6. Calculate the current source current using $I = V/R_{CAL}$, where RCAL is a high-precision resistance and V is measured by performing an A/D conversion.

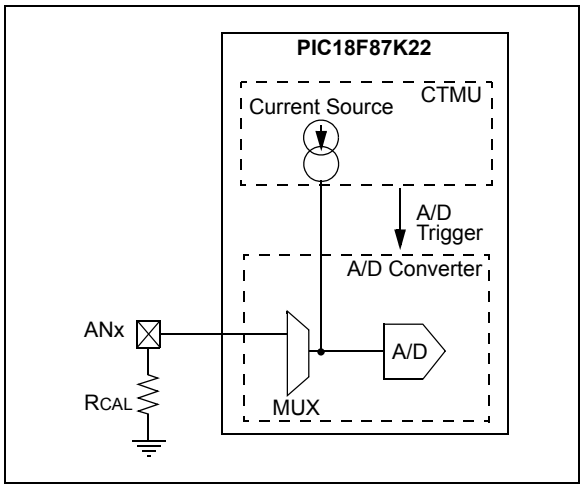

© 2009-2018 Microchip Technology Inc.

The CTMU current source may be trimmed with the trim bits in CTMUICON using an iterative process to get the exact current desired. Alternatively, the nominal value without adjustment may be used. That value may be stored by software for use in all subsequent capacitive or time measurements.

To calculate the value for RCAL, the nominal current must be chosen. Then, the resistance can be calculated.

For example, if the A/D Converter reference voltage is 3.3V, use 70% of full scale (or 2.31V) as the desired approximate voltage to be read by the A/D Converter. If the range of the CTMU current source is selected to be 0.55 μA, the resistor value needed is calculated as RCAL = 2.31V/0.55 μA, for a value of 4.2 MΩ. Similarly, if the current source is chosen to be 5.5 μA, RCAL would be 420,000Ω, and 42,000Ω if the current source is set to 55 μA.

**FIGURE 27-2: CTMU CURRENT SOURCE CALIBRATION CIRCUIT**




A value of 70% of full-scale voltage is chosen to make sure that the A/D Converter was in a range that is well above the noise floor. If an exact current is chosen to incorporate the trimming bits from CTMUICON, the resistor value of RCAL may need to be adjusted accordingly. RCAL may also be adjusted to allow for available resistor values. RCAL should be of the highest precision available, in light of the precision needed for the circuit that the CTMU will be measuring. A recommended minimum would be 0.1% tolerance.

The following examples show a typical method for performing a CTMU current calibration.

- Example 27-1 demonstrates how to initialize the A/D Converter and the CTMU.

  This routine is typical for applications using both modules.

- Example 27-2 demonstrates one method for the actual calibration routine.

  This method manually triggers the A/D Converter to demonstrate the entire step-wise process. It is also possible to automatically trigger the conversion by setting the CTMU's CTTRIG bit (CTMUCONH<0>).

**EXAMPLE 27-1: SETUP FOR CTMU CALIBRATION ROUTINES**

```
#include "p18cxxx.h"
/*****************************************************************************/
/*Setup CTMU *****************************************************************/
/*****************************************************************************/
void setup(void)

{ //CTMUCON - CTMU Control register

    CTMUCONH = 0x00;              //make sure CTMU is disabled
    CTMUCONL = 0X90;
    //CTMU continues to run when emulator is stopped,CTMU continues
    //to run in idle mode,Time Generation mode disabled, Edges are blocked
    //No edge sequence order, Analog current source not grounded, trigger
    //output disabled, Edge2 polarity = positive level, Edge2 source =
    //source 0, Edge1 polarity = positive level, Edge1 source = source 0,
    // Set Edge status bits to zero


    //CTMUICON - CTMU Current Control Register
    CTMUICON = 0x01;              //0.55uA, Nominal - No Adjustment

/*****************************************************************************/
//Setup AD converter;
/*****************************************************************************/

    TRISA=0x04;                   //set channel 2 as an input

    // Configured AN2 as an analog channel
    // ANCON0
    ANCON0 = 0X04;
    // ANCON1
    ANCON1 = 0XE0;

    // ADCON2
    ADCON2bits.ADFM=1;            // Resulst format 1= Right justified
    ADCON2bits.ACQT=1;            // Acquition time 7 = 20TAD 2 = 4TAD 1=2TAD
    ADCON2bits.ADCS=2;            // Clock conversion bits 6= FOSC/64 2=FOSC/32

    // ADCON0
    ADCON1bits.VCFG0 =0;          // Vref+ = AVdd
    ADCON0bits.VCFG1 =0;          // Vref+ = AVdd
    ADCON0bits.VCFG = 0;          // Vref- = AVss
    ADCON0bits.CHS=2;             // Select ADC channel

    ADCON0bits.ADON=1;            // Turn on ADC

}
```


© 2009-2018 Microchip Technology Inc.

**EXAMPLE 27-2: CURRENT CALIBRATION ROUTINE**

```
#include "p18cxxx.h"

#define COUNT 500                       //@ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define RCAL .027                       //R value is 4200000 (4.2M)
                                        //scaled so that result is in
                                        //1/100th of uA
#define ADSCALE 1023                    //for unsigned conversion 10 sig bits
#define ADREF 3.3                       //Vdd connected to A/D Vr+

int main(void)
{
    int i;
    int j = 0; //index for loop
    unsigned int Vread = 0;
    double VTot = 0;
    float Vavg=0, Vcal=0, CTMUISrc = 0;  //float values stored for calcs

//assume CTMU and A/D have been setup correctly
//see Example 25-1 for CTMU & A/D setup
setup();

CTMUCONHbits.CTMUEN = 1;                //Enable the CTMU
    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;       //drain charge on the circuit
        DELAY;                          //wait 125us
        CTMUCONHbits.IDISSEN = 0;       //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;      //Begin charging the circuit
                                        //using CTMU current source
        DELAY;                          //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;      //Stop charging circuit

        PIR1bits.ADIF = 0;              //make sure A/D Int not set
        ADCON0bits.GO=1;                //and begin A/D conv.
        while(!PIR1bits.ADIF);          //Wait for A/D convert complete

        Vread = ADRES;                  //Get the value from the A/D
        PIR1bits.ADIF = 0;              //Clear A/D Interrupt Flag
        VTot += Vread;                  //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);        //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);
    CTMUISrc = Vcal/RCAL;               //CTMUISrc is in 1/100ths of uA

}
```

### 27.4.2 CAPACITANCE CALIBRATION

There is a small amount of capacitance from the internal A/D Converter sample capacitor, as well as stray capacitance from the circuit board traces and pads that affect the precision of capacitance measurements. A measurement of the stray capacitance can be taken by making sure the desired capacitance to be measured has been removed.

After removing the capacitance to be measured:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT (= 1).
3. Wait for a fixed delay of time, t.
4. Clear EDG1STAT.
5. Perform an A/D conversion.
6. Calculate the stray and A/D sample capacitances:

$$C_{OFFSET} = C_{STRAY} + C_{AD} = (I \bullet t)/V$$

Where:

- I is known from the current source measurement step
- t is a fixed delay
- V is measured by performing an A/D conversion

This measured value is then stored and used for calculations of time measurement or subtracted for capacitance measurement. For calibration, it is expected that the capacitance of $C_{STRAY}$ + $C_{AD}$ is approximately known; $C_{AD}$ is approximately 4 pF.

An iterative process may be required to adjust the time, t, that the circuit is charged to obtain a reasonable voltage reading from the A/D Converter. The value of t may be determined by setting $C_{OFFSET}$ to a theoretical value and solving for t. For example, if $C_{STRAY}$ is theoretically calculated to be 11 pF, and V is expected to be 70% of $V_{DD}$ or 2.31V, t would be:

$$(4\ pF + 11\ pF) \bullet 2.31V/0.55\ \mu A$$

or 63 μs.

See Example 27-3 for a typical routine for CTMU capacitance calibration.


© 2009-2018 Microchip Technology Inc.

**EXAMPLE 27-3: CAPACITANCE CALIBRATION ROUTINE**

```
#include "p18cxxx.h"

#define COUNT 25                        //@ 8MHz INTFRC = 62.5 us.
#define ETIME COUNT*2.5                 //time in uS
#define DELAY for(i=0;i<COUNT;i++)
#define ADSCALE 1023                    //for unsigned conversion 10 sig bits
#define ADREF 3.3                       //Vdd connected to A/D Vr+
#define RCAL .027                       //R value is 4200000 (4.2M)
                                        //scaled so that result is in
                                        //1/100th of uA

int main(void)
{
    int i;
    int j = 0;                          //index for loop
    unsigned int Vread = 0;
    float CTMUISrc, CTMUCap, Vavg, VTot, Vcal;

//assume CTMU and A/D have been setup correctly
//see Example 25-1 for CTMU & A/D setup
setup();

CTMUCONHbits.CTMUEN = 1;                //Enable the CTMU
    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;       //drain charge on the circuit
        DELAY;                          //wait 125us
        CTMUCONHbits.IDISSEN = 0;       //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;      //Begin charging the circuit
                                        //using CTMU current source
        DELAY;                          //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;      //Stop charging circuit

        PIR1bits.ADIF = 0;              //make sure A/D Int not set
        ADCON0bits.GO=1;                //and begin A/D conv.
        while(!PIR1bits.ADIF);          //Wait for A/D convert complete


        Vread = ADRES;                  //Get the value from the A/D
        PIR1bits.ADIF = 0;              //Clear A/D Interrupt Flag
        VTot += Vread;                  //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);        //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);
    CTMUISrc = Vcal/RCAL;               //CTMUISrc is in 1/100ths of uA
    CTMUCap = (CTMUISrc*ETIME/Vcal)/100;
}
```

## 27.5 Measuring Capacitance with the CTMU

There are two ways to measure capacitance with the CTMU. The absolute method measures the actual capacitance value. The relative method only measures for any change in the capacitance.

### 27.5.1 ABSOLUTE CAPACITANCE MEASUREMENT

For absolute capacitance measurements, both the current and capacitance calibration steps found in **Section 27.4 "Calibrating the CTMU Module"** should be followed.

To perform these measurements:

1. Initialize the A/D Converter.
2. Initialize the CTMU.
3. Set EDG1STAT.
4. Wait for a fixed delay, T.
5. Clear EDG1STAT.
6. Perform an A/D conversion.
7. Calculate the total capacitance, $C_{TOTAL} = (I * T)/V$, where:
   - I is known from the current source measurement step (**Section 27.4.1 "Current Source Calibration"**)
   - T is a fixed delay
   - V is measured by performing an A/D conversion
8. Subtract the stray and A/D capacitance ($C_{OFFSET}$ from **Section 27.4.2 "Capacitance Calibration"**) from $C_{TOTAL}$ to determine the measured capacitance.

### 27.5.2 RELATIVE CHARGE MEASUREMENT

Not all applications require precise capacitance measurements. When detecting a valid press of a capacitance-based switch, only a relative change of capacitance needs to be detected.

In such an application, when the switch is open (or not touched), the total capacitance is the capacitance of the combination of the board traces, the A/D Converter and other elements. A larger voltage will be measured by the A/D Converter. When the switch is closed (or touched), the total capacitance is larger due to the addition of the capacitance of the human body to the above listed capacitances and a smaller voltage will be measured by the A/D Converter.

To detect capacitance changes simply:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT.
3. Wait for a fixed delay.
4. Clear EDG1STAT.
5. Perform an A/D conversion.

The voltage measured by performing the A/D conversion is an indication of the relative capacitance. In this case, no calibration of the current source or circuit capacitance measurement is needed. (For a sample software routine for a capacitive touch switch, see Example 27-4.)


© 2009-2018 Microchip Technology Inc.

**EXAMPLE 27-4: ROUTINE FOR CAPACITIVE TOUCH SWITCH**

```
#include "p18cxxx.h"

#define COUNT 500                       //@ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define OPENSW 1000                     //Un-pressed switch value
#define TRIP 300                        //Difference between pressed
                                        //and un-pressed switch
#define HYST 65                         //amount to change
                                        //from pressed to un-pressed
#define PRESSED 1
#define UNPRESSED 0

int main(void)
{
    unsigned int Vread;                 //storage for reading
    unsigned int switchState;
    int i;

    //assume CTMU and A/D have been setup correctly
    //see Example 25-1 for CTMU & A/D setup
    setup();

    CTMUCONHbits.CTMUEN = 1;            //Enable the CTMU

    CTMUCONHbits.IDISSEN = 1;           //drain charge on the circuit
    DELAY;                              //wait 125us
    CTMUCONHbits.IDISSEN = 0;           //end drain of circuit

    CTMUCONLbits.EDG1STAT = 1;          //Begin charging the circuit
                                        //using CTMU current source
    DELAY;                              //wait for 125us
    CTMUCONLbits.EDG1STAT = 0;          //Stop charging circuit

    PIR1bits.ADIF = 0;                  //make sure A/D Int not set
    ADCON0bits.GO=1;                    //and begin A/D conv.
    while(!PIR1bits.ADIF);              //Wait for A/D convert complete

    Vread = ADRES;                      //Get the value from the A/D

    if(Vread < OPENSW - TRIP)
    {
        switchState = PRESSED;
    }
    else if(Vread > OPENSW - TRIP + HYST)
    {
        switchState = UNPRESSED;
    }
}
```

## 27.6 Measuring Time with the CTMU Module

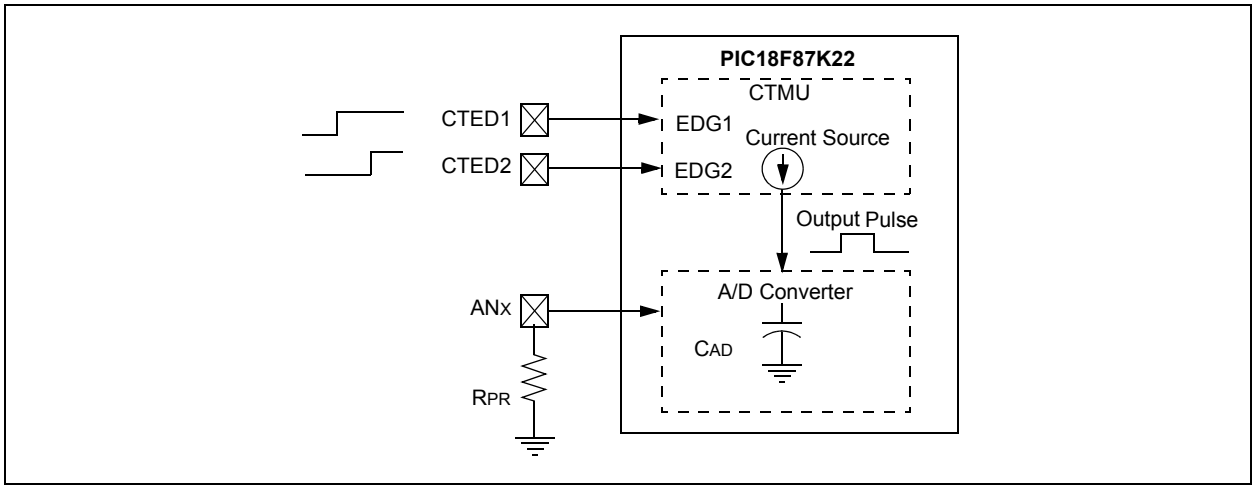
Time can be precisely measured after the ratio (C/I) is measured from the current and capacitance calibration step. To do that:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT.
3. Set EDG2STAT.
4. Perform an A/D conversion.
5. Calculate the time between edges as $T = (C/I) * V$, where:
   - I is calculated in the current calibration step (**Section 27.4.1 "Current Source Calibration"**)
   - C is calculated in the capacitance calibration step (**Section 27.4.2 "Capacitance Calibration"**)
   - V is measured by performing the A/D conversion

It is assumed that the time measured is small enough that the capacitance, $C_{OFFSET}$, provides a valid voltage to the A/D Converter. For the smallest time measurement, always set the A/D Channel Select register (AD1CHS) to an unused A/D channel, the corresponding pin for which is not connected to any circuit board trace. This minimizes added stray capacitance, keeping the total circuit capacitance close to that of the A/D Converter itself (25 pF).

To measure longer time intervals, an external capacitor may be connected to an A/D channel and that channel selected whenever making a time measurement.

**FIGURE 27-3: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR TIME MEASUREMENT**





© 2009-2018 Microchip Technology Inc.

## 27.7 Creating a Delay with the CTMU Module

A unique feature on board the CTMU module is its ability to generate system clock independent output pulses, based on either an external voltage or an external capacitor value. When using an external voltage, this is accomplished using the CTDIN input pin as a trigger for the pulse delay. When using an external capacitor value, this is accomplished using the internal comparator voltage reference module and Comparator 2 input pin.The pulse is output onto the CTPLS pin. To enable this mode, set the TGEN bit.

See Figure 27-4 for an example circuit. When CTMUDS (ODCON3<0>) is cleared, the pulse delay is determined by the output of Comparator 2, and when it is set, the pulse delay is determined by the input of CTDIN. CDELAY is chosen by the user to determine the output pulse width on CTPLS. The pulse width is calculated by $T = (C_{DELAY}/I) * V$, where I is known from the current source measurement step (**Section 27.4.1 "Current Source Calibration"**) and V is the Internal Reference Voltage (CVREF).

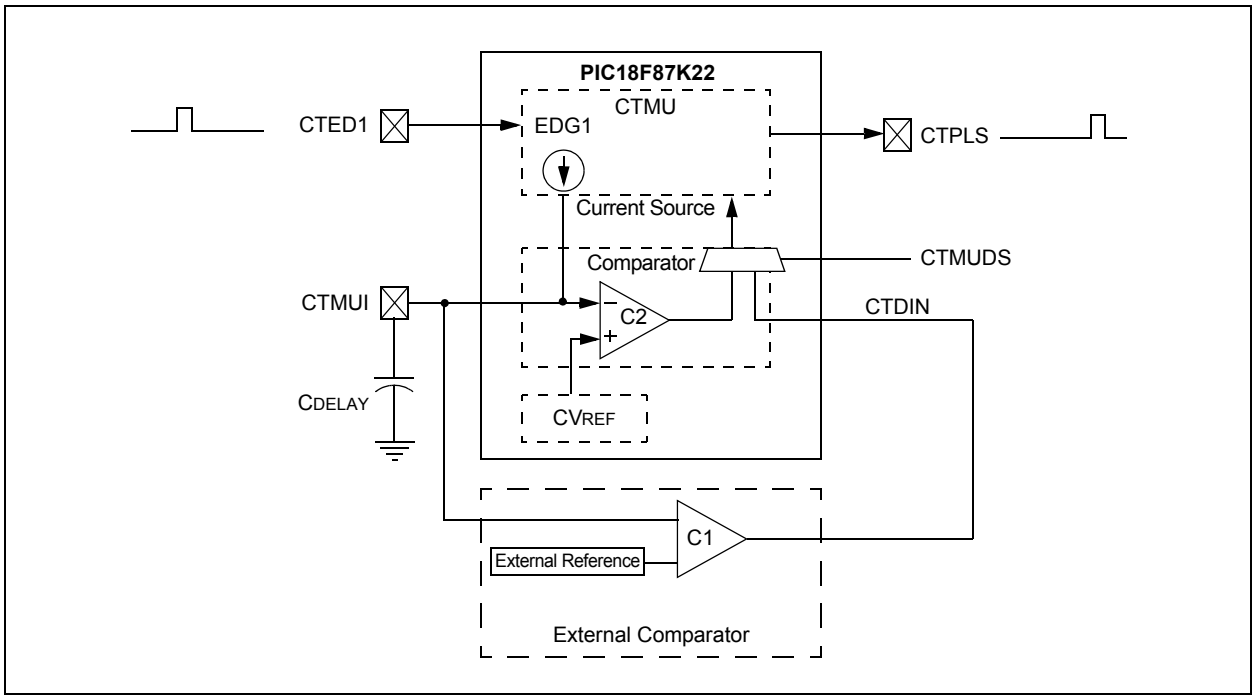
An example use of the external capacitor feature is interfacing with variable capacitive-based sensors, such as a humidity sensor. As the humidity varies, the pulse-width output on CTPLS will vary. An example use of the CTDIN feature is interfacing with a digital sensor. The CTPLS output pin can be connected to an input capture pin and the varying pulse width measured to determine the sensor's output in the application.

To use this feature:

1. If CTMUDS is cleared, initialize Comparator 2.
2. If CTMUDS is cleared, initialize the comparator voltage reference.
3. Initialize the CTMU and enable time delay generation by setting the TGEN bit.
4. Set EDG1STAT.

When CTMUDS is cleared, as soon as CDELAY charges to the value of the voltage reference trip point, an output pulse is generated on CTPLS. When CTMUDS is set, as soon as CTDIN is set, an output pulse is generated on CTPLS.

**FIGURE 27-4: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR PULSE DELAY GENERATION**



## 27.8 Measuring Temperature with the CTMU Module

The CTMU, along with an internal diode, can be used to measure the temperature. The ADC can be connected to the internal diode and the CTMU module can source the current to the diode. The ADC reading will reflect the temperature. With the increase, the ADC readings will go low. This can be used for low-cost temperature measurement applications.

**EXAMPLE 27-5: ROUTINE FOR TEMPERATURE MEASUREMENT USING INTERNAL DIODE**

```
// Initialize CTMU
CTMUICON = 0x03;
CTMUCONHbits.CTMUEN = 1;
CTMUCONLbits.EDG1STAT = 1;

// Initialize ADC
ADCON0 = 0xE5;                          // Enable ADC and connect to Internal diode
ADCON1 = 0x00;
ADCON2 = 0xBE;                          // Right Justified

ADCON0bits.GO = 1;                      // Start conversion
while(ADCON0bits.G0);
Temp = ADRES;                           // Read ADC results (inversely proportional to temperature)
```

**Note:** The temperature diode is not calibrated or standardized; the user must calibrate the diode to their application.


© 2009-2018 Microchip Technology Inc.

## 27.9 Operation During Sleep/Idle Modes

### 27.9.1 SLEEP MODE

When the device enters any Sleep mode, the CTMU module current source is always disabled. If the CTMU is performing an operation that depends on the current source when Sleep mode is invoked, the operation may not terminate correctly. Capacitance and time measurements may return erroneous values.

### 27.9.2 IDLE MODE

The behavior of the CTMU in Idle mode is determined by the CTMUSIDL bit (CTMUCONH<5>). If CTMUSIDL is cleared, the module will continue to operate in Idle mode. If CTMUSIDL is set, the module's current source is disabled when the device enters Idle mode. In this case, if the module is performing an operation when Idle mode is invoked, the results will be similar to those with Sleep mode.

## 27.10 Effects of a Reset on CTMU

Upon Reset, all registers of the CTMU are cleared. This disables the CTMU module, turns off its current source and returns all configuration options to their default settings. The module needs to be re-initialized following any Reset.

If the CTMU is in the process of taking a measurement at the time of Reset, the measurement will be lost. A partial charge may exist on the circuit that was being measured, which should be properly discharged before the CTMU makes subsequent attempts to make a measurement. The circuit is discharged by setting and clearing the IDISSEN bit (CTMUCONH<1>) while the A/D Converter is connected to the appropriate channel.

**TABLE 27-1: REGISTERS ASSOCIATED WITH CTMU MODULE**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| CTMUCONH | CTMUEN | — | CTMUSIDL | TGEN | EDGEN | EDGSEQEN | IDISSEN | CTTRIG |
| CTMUCONL | EDG2POL | EDG2SEL1 | EDG2SEL0 | EDG1POL | EDG1SEL1 | EDG1SEL0 | EDG2STAT | EDG1STAT |
| CTMUICON | ITRIM5 | ITRIM4 | ITRIM3 | ITRIM2 | ITRIM1 | ITRIM0 | IRNG1 | IRNG0 |
| PIR3 | TMR5GIF | — | RC2IF | TX2IF | CTMUIF | CCP2IF | CCP1IF | RTCCIF |
| PIE3 | TMR5GIE | — | RC2IE | TX2IE | CTMUIE | CCP2IE | CCP1IE | RTCCIE |
| IPR3 | TMR5GIP | — | RC2IP | TX2IP | CTMUIP | CCP2IP | CCP1IP | RTCCIP |

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

## 28.0 SPECIAL FEATURES OF THE CPU

The PIC18F87K22 family of devices includes several features intended to maximize reliability and minimize cost through elimination of external components. These include:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT) and On-chip Regulator
- Fail-Safe Clock Monitor
- Two-Speed Start-up
- Code Protection
- ID Locations
- In-Circuit Serial Programming™ (ICSP™)

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 PIC18F87K22 family of devices has a Watchdog Timer, which is either permanently enabled via the Configuration bits or software controlled (if configured as disabled).

The inclusion of an internal RC oscillator (LF-INTOSC) 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.

### 28.1 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.

The user will note that address, 300000h, is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h-3FFFFFh), which can only be accessed using table reads and table writes.

Software programming of the Configuration registers is done in a manner similar to programming the Flash memory. The WR bit in the EECON1 register starts a self-timed write to the Configuration register. In normal Operation mode, a `TBLWT` instruction, with the TBLPTR pointing to the Configuration register, sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a `TBLWT` instruction can write a '1' or a '0' into the cell. For additional details on Flash programming, refer to **Section 7.5 "Writing to Flash Program Memory"**.


© 2009-2018 Microchip Technology Inc.

**TABLE 28-1: 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 |
|---|---|---|---|---|---|---|---|---|---|---|
| 300000h | CONFIG1L | — | XINST | — | SOSCSEL1 | SOSCSEL0 | INTOSCSEL | — | $\overline{\text{RETEN}}$ | -1-1 1--1 |
| 300001h | CONFIG1H | IESO | FCMEN | — | PLLCFG | FOSC3 | FOSC2 | FOSC1 | FOSC0 | 00-0 1000 |
| 300002h | CONFIG2L | — | BORPWR1 | BORWPR0 | BORV1 | BORV0 | BOREN1 | BOREN0 | $\overline{\text{PWRTEN}}$ | -111 1111 |
| 300003h | CONFIG2H | — | WDTPS4 | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN1 | WDTEN0 | -111 1111 |
| 300004h | CONFIG3L | WAIT(2) | BW(2) | ABW1(2) | ABW0(2) | EASHFT(2) | — | — | RTCOSC | ---- ---1 |
| 300005h | CONFIG3H | MCLRE | — | — | — | MSSPMSK | — | ECCPMX(2) | CCP2MX | 1--- 1-11 |
| 300006h | CONFIG4L | $\overline{\text{DEBUG}}$ | — | — | BBSIZ0 | — | — | — | STVREN | 1--1 ---1 |
| 300008h | CONFIG5L | CP7(1) | CP6(1) | CP5(1) | CP4(1) | CP3 | CP2 | CP1 | CP0 | 1111 1111 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — | 11-- ---- |
| 30000Ah | CONFIG6L | WRT7(1) | WRT6(1) | WRT5(1) | WRT4(1) | WRT3 | WRT2 | WRT1 | WRT0 | 1111 1111 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC | — | — | — | — | — | 111- ---- |
| 30000Ch | CONFIG7L | EBTR7(1) | EBTR6(1) | EBTR5(1) | EBTR4(1) | EBTR3 | EBTR2 | EBTR1 | EBTR0 | 1111 1111 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — | -1-- ---- |
| 3FFFFEh | DEVID1(3) | DEV2 | DEV1 | DEV0 | REV4 | REV3 | REV2 | REV1 | REV0 | xxxx xxxx |
| 3FFFFFh | DEVID2(3) | DEV10 | DEV9 | DEV8 | DEV7 | DEV6 | DEV5 | DEV4 | DEV3 | xxxx xxxx |

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

**Note 1:** Implemented only on the PIC18F67K22 and PIC18F87K22 devices.

**2:** Implemented only on the 80-pin devices (PIC18F8XK22).

**3:** See Register 28-14 for DEVID1 values. DEVID registers are read-only and cannot be programmed by the user.

**REGISTER 28-1: CONFIG1L: CONFIGURATION REGISTER 1 LOW (BYTE ADDRESS 300000h)**

| U-0 | R/P-1 | U-0 | R/P-1 | R/P-1 | R/P-1 | U-0 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | XINST | — | SOSCSEL1 | SOSCSEL0 | INTOSCSEL | — | $\overline{\text{RETEN}}$ |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | P = Programmable 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 **Unimplemented:** Read as '0'

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 **Unimplemented:** Read as '0'

bit 4-3 **SOSCSEL<1:0>:** SOSC Power Selection and Mode Configuration bits
- 11 = High-power SOSC circuit is selected
- 10 = Digital (SCLKI) mode; I/O port functionality of RC0 and RC1 is enabled
- 01 = Low-power SOSC circuit is selected
- 00 = Reserved

bit 2 **INTOSCSEL:** LF-INTOSC Low-power Enable bit
- 1 = LF-INTOSC is in High-Power mode during Sleep
- 0 = LF-INTOSC is in Low-Power mode during Sleep

bit 1 **Unimplemented:** Read as '0'

bit 0 **$\overline{\text{RETEN}}$:** VREG Sleep Enable bit
- 1 = Regulator power while in Sleep mode is controlled by VREGSLP (WDTCON<7>)
- 0 = Regulator power while in Sleep mode is controlled by SRETEN (WDTCON<4>). Ultra low-power regulator is enabled.


© 2009-2018 Microchip Technology Inc.

**REGISTER 28-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)**

| R/P-0 | R/P-0 | U-0 | U-0 | R/P-1 | R/P-0 | R/P-0 | R/P-0 |
|---|---|---|---|---|---|---|---|
| IESO | FCMEN | — | PLLCFG(1) | FOSC3(2) | FOSC2(2) | FOSC1(2) | FOSC0(2) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | P = Programmable 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 **IESO:** Internal/External Oscillator Switchover 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 **Unimplemented:** Read as '0'

bit 4 **PLLCFG:** 4x PLL Enable bit(1)
1 = Oscillator is multiplied by 4
0 = Oscillator is used directly

bit 3-0 **FOSC<3:0>:** Oscillator Selection bits(2)
1101 = EC1, EC oscillator **(low power, DC-160 kHz)**
1100 = EC1IO, EC oscillator with CLKOUT function on RA6 **(low power, DC-160 kHz)**
1011 = EC2, EC oscillator **(medium power, 160 kHz-16 MHz)**
1010 = EC2IO, EC oscillator with CLKOUT function on RA6 **(medium power, DC-160 kHz-16 MHz)**
1001 = INTIO1, internal RC oscillator with CLKOUT function on RA6
1000 = INTIO2, internal RC oscillator
0111 = RC, external RC oscillator
0110 = RCIO, external RC oscillator with CKLOUT function on RA6
0101 = EC3, EC oscillator **(high power, 4 MHz-64 MHz)**
0100 = EC3IO, EC oscillator with CLKOUT function on RA6 **(high power, 4 MHz-64 MHz)**
0011 = HS1, HS oscillator **(medium power, 4 MHz-16 MHz)**
0010 = HS2, HS oscillator **(high power, 16 MHz-25 MHz)**
0001 = XT oscillator
0000 = LP oscillator

**Note 1:** Not valid for the INTIOx PLL mode.
**2:** INTIO + PLL can be enabled only by the PLLEN bit (OSCTUNE<6>). Other PLL modes can be enabled by either the PLLEN bit or the PLLCFG (CONFIG1H<4>) bit.

**REGISTER 28-3: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)**

| U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | BORPWR1(1) | BORPWR0(1) | BORV1(1) | BORV0(1) | BOREN1(2) | BOREN0(2) | $\overline{\text{PWRTEN}}$(2) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | P = Programmable 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 **Unimplemented:** Read as '0'

bit 6-5 **BORPWR<1:0>:** BORMV Power-Level bits(1)
- 11 = ZPBORVMV instead of BORMV is selected
- 10 = BORMV is set to a high-power level
- 01 = BORMV is set to a medium power level
- 00 = BORMV is set to a low-power level

bit 4-3 **BORV<1:0>:** Brown-out Reset Voltage bits(1)
- 11 = $V_{BORMV}$ is set to 1.8V
- 10 = $V_{BORMV}$ is set to 2.0V
- 01 = $V_{BORMV}$ is set to 2.7V
- 00 = $V_{BORMV}$ is set to 3.0V

bit 2-1 **BOREN<1:0>:** Brown-out Reset Enable bits(2)
- 11 = Brown-out Reset is enabled in hardware only (SBOREN is disabled)
- 10 = Brown-out Reset is enabled in hardware only and disabled in Sleep mode (SBOREN is disabled)
- 01 = Brown-out Reset is enabled and controlled by software (SBOREN is enabled)
- 00 = Brown-out Reset is disabled in hardware and software

bit 0 **$\overline{\text{PWRTEN}}$:** Power-up Timer Enable bit(2)
- 1 = PWRT is disabled
- 0 = PWRT is enabled

**Note 1:** For the specifications, see **Section 31.1 "DC Characteristics: Supply Voltage PIC18F87K22 Family (Industrial/Extended)"**.

**2:** The Power-up Timer is decoupled from Brown-out Reset, allowing these features to be independently controlled.


© 2009-2018 Microchip Technology Inc.

**REGISTER 28-4: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)**

| U-0 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| — | WDTPS4 | WDTPS3 | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN1 | WDTEN0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | P = Programmable 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 **Unimplemented:** Read as '0'

bit 6-2 **WDTPS<4:0>:** Watchdog Timer Postscale Select bits

10101-11111 = Reserved
10100 = 1:1,048,576 (4,194.304s)
10011 = 1:524,288 (2,097.152s)
10010 = 1:262,144 (1,048.576s)
10001 = 1:131,072 (524.288s)
10000 = 1:65,536 (262.144s)
01111 = 1:32,768 (131.072s)
01110 = 1:16,384 (65.536s)
01101 = 1:8,192 (32.768s)
01100 = 1:4,096 (16.384s)
01011 = 1:2,048 (8.192s)
01010 = 1:1,024 (4.096s)
01001 = 1:512 (2.048s)
01000 = 1:256 (1.024s)
00111 = 1:128 (512 ms)
00110 = 1:64 (256 ms)
00101 = 1:32 (128 ms)
00100 = 1:16 (64 ms)
00011 = 1:8 (32 ms)
00010 = 1:4 (16 ms)
00001 = 1:2 (8 ms)
00000 = 1:1 (4 ms)

bit 1-0 **WDTEN<1:0>:** Watchdog Timer Enable bits

11 = WDT is enabled in hardware; SWDTEN bit is disabled
10 = WDT is controlled by the SWDTEN bit setting
01 = WDT is enabled only while the device is active and disabled in Sleep mode; SWDTEN bit is disabled
00 = WDT is disabled in hardware; SWDTEN bit is disabled

**REGISTER 28-5: CONFIG3L: CONFIGURATION REGISTER 3 LOW (BYTE ADDRESS 300004h)**

| U-1 | U-1 | U-1 | U-1 | U-1 | U-0 | U-0 | R/P-1 |
|---|---|---|---|---|---|---|---|
| WAIT(1) | BW(1) | ABW1(1) | ABW0(1) | EASHFT(1) | — | — | RTCOSC |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | P = Programmable 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 **ABW<1:0>:** External Memory Bus Configuration bits(1)
11 = 8-Bit Address mode (Microcontroller mode)
10 = 12-Bit Address mode
01 = 16-Bit Address mode
00 = 20-Bit Address mode

bit 3 **EASHFT:** External Address Bus Shift Enable bit(1)
1 = Address shifting is enabled; external address is shifted to start at 000000h
0 = Address shifting is disabled; external address bus reflects the PC value

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **RTCOSC:** RTCC Reference Clock Select bit
1 = RTCC uses SOSC as the reference clock
0 = RTCC uses LF-INTOSC as the reference clock

**Note 1:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.


© 2009-2018 Microchip Technology Inc.

**REGISTER 28-6: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)**

| R/P-1 | U-0 | U-0 | U-0 | R/P-1 | U-0 | R/P-1 | R/P-1 |
|---|---|---|---|---|---|---|---|
| MCLRE | — | — | — | MSSPMSK | — | ECCPMX(1) | CCP2MX |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | P = Programmable 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 **MCLRE:** $\overline{\text{MCLR}}$ Pin Enable bit
1 = $\overline{\text{MCLR}}$ pin is enabled; RG5 input pin is disabled
0 = RG5 input pin is enabled; $\overline{\text{MCLR}}$ is disabled

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **MSSPMSK:** MSSP V3 7-Bit Address Masking Mode Enable bit
1 = 7-Bit Address Masking mode is enabled
0 = 5-Bit Address Masking mode is enabled

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

bit 1 **ECCPMX:** ECCP MUX bit(1)
1 =
- ECCP1 (P1B/P1C) is multiplexed onto RE6 and RE5, CCP6 onto RE6, and CCP7 onto RE5
- ECCP3 (P3B/P3C) is multiplexed onto RE4 and RE3, CCP8 onto RE4, and CCP9 onto RE3

0 =
- ECCP1 (P1B/P1C) is multiplexed onto RH7 and RH6, CCP6 onto RH7, and CCP7(2) onto RH6
- ECCP3 (P3B/P3C) is multiplexed onto RH5 and RH4, CCP8 onto RH5, and CCP9(2) onto RH4

bit 0 **CCP2MX:** ECCP2 MUX bit
1 = ECCP2 is multiplexed with RC1
0 = ECCP2 is multiplexed with RB3 in Extended Microcontroller mode; ECCP2 is multiplexed with RE7 in Microcontroller mode

**Note 1:** Unimplemented on 64-pin devices (PIC18F6XK22), read as '0'.
**2:** Not implemented on 32K devices (PIC18F65K22 and PIC18F85K22).

**REGISTER 28-7: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)**

| R/P-1 | U-0 | U-0 | R/P-0 | U-0 | U-0 | U-0 | R/P-1 |
|---|---|---|---|---|---|---|---|
| $\overline{\text{DEBUG}}$ | — | — | BBSIZ0 | — | — | — | STVREN |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | P = Programmable 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 **$\overline{\text{DEBUG}}$:** 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 6-5 **Unimplemented:** Read as '0'

bit 4 **BBSIZ0:** Boot Block Size Select bit
1 = 2 kW boot block size
0 = 1 kW boot block size

bit 3-1 **Unimplemented:** Read as '0'

bit 0 **STVREN:** Stack Full/Underflow Reset Enable bit
1 = Stack full/underflow will cause Reset
0 = Stack full/underflow will not cause Reset


© 2009-2018 Microchip Technology Inc.

**REGISTER 28-8: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)**

| R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|---|---|---|---|---|---|---|---|
| CP7(1) | CP6(1) | CP5(1) | CP4(1) | CP3 | CP2 | CP1 | CP0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | C = Clearable 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 **CP7:** Code Protection bit(1)
1 = Block 7 is not code-protected(2)
0 = Block 7 is code-protected(2)

bit 6 **CP6:** Code Protection bit(1)
1 = Block 6 is not code-protected(2)
0 = Block 6 is code-protected(2)

bit 5 **CP5:** Code Protection bit(1)
1 = Block 5 is not code-protected(2)
0 = Block 5 is code-protected(2)

bit 4 **CP4:** Code Protection bit(1)
1 = Block 4 is not code-protected(2)
0 = Block 4 is code-protected(2)

bit 3 **CP3:** Code Protection bit
1 = Block 3 is not code-protected(2)
0 = Block 3 is code-protected(2)

bit 2 **CP2:** Code Protection bit
1 = Block 2 is not code-protected(2)
0 = Block 2 is code-protected(2)

bit 1 **CP1:** Code Protection bit
1 = Block 1 is not code-protected(2)
0 = Block 1 is code-protected(2)

bit 0 **CP0:** Code Protection bit
1 = Block 0 is not code-protected(2)
0 = Block 0 is code-protected(2)

**Note 1:** This bit is available only on PIC18F67K22 and PIC18F87K22 devices.
**2:** For the memory size of the blocks, see Figure 28-6.

**REGISTER 28-9: CONFIG5H: CONFIGURATION REGISTER 5 HIGH (BYTE ADDRESS 300009h)**

| R/C-1 | R/C-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| CPD | CPB | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | C = Clearable 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 **CPD:** Data EEPROM Code Protection bit
1 = Data EEPROM is not code-protected
0 = Data EEPROM is code-protected

bit 6 **CPB:** Boot Block Code Protection bit
1 = Boot block is not code-protected(1)
0 = Boot block is code-protected(1)

bit 5-0 **Unimplemented:** Read as '0'

**Note 1:** For the memory size of the blocks, see Figure 28-6. The boot block size changes with BBSIZ0.


© 2009-2018 Microchip Technology Inc.

**REGISTER 28-10: CONFIG6L: CONFIGURATION REGISTER 6 LOW (BYTE ADDRESS 30000Ah)**

| R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|---|---|---|---|---|---|---|---|
| WRT7(1) | WRT6(1) | WRT5(1) | WRT4(1) | WRT3 | WRT2 | WRT1 | WRT0 |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | C = Clearable 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 **WRT7:** Write Protection bit(1)
1 = Block 7 is not write-protected(2)
0 = Block 7 is write-protected(2)

bit 6 **WRT6:** Write Protection bit(1)
1 = Block 6 is not write-protected(2)
0 = Block 6 is write-protected(2)

bit 5 **WRT5:** Write Protection bit(1)
1 = Block 5 is not write-protected(2)
0 = Block 5 is write-protected(2)

bit 4 **WRT4:** Write Protection bit(1)
1 = Block 4 is not write-protected(2)
0 = Block 4 is write-protected(2)

bit 3 **WRT3:** Write Protection bit
1 = Block 3 is not write-protected(2)
0 = Block 3 is write-protected(2)

bit 2 **WRT2:** Write Protection bit
1 = Block 2 is not write-protected(2)
0 = Block 2 is write-protected(2)

bit 1 **WRT1:** Write Protection bit
1 = Block 1 is not write-protected(2)
0 = Block 1 is write-protected(2)

bit 0 **WRT0:** Write Protection bit
1 = Block 0 is not write-protected(2)
0 = Block 0 is write-protected(2)

**Note 1:** This bit is available only on PIC18F67K22 and PIC18F87K22 devices.
**2:** For the memory size of the blocks, see Figure 28-6.

**REGISTER 28-11: CONFIG6H: CONFIGURATION REGISTER 6 HIGH (BYTE ADDRESS 30000Bh)**

| R/C-1 | R/C-1 | R-1 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| WRTD | WRTB | WRTC(1) | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | C = Clearable 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 **WRTD:** Data EEPROM Write Protection bit
1 = Data EEPROM is not write-protected
0 = Data EEPROM is write-protected

bit 6 **WRTB:** Boot Block Write Protection bit
1 = Boot block is not write-protected(2)
0 = Boot block is write-protected(2)

bit 5 **WRTC:** Configuration Register Write Protection bit(1)
1 = Configuration registers are not write-protected(2)
0 = Configuration registers are write-protected(2)

bit 4-0 **Unimplemented:** Read as '0'

**Note 1:** This bit is read-only in normal Execution mode; it can be written only in Program mode.
**2:** For the memory size of the blocks, see Figure 28-6.


© 2009-2018 Microchip Technology Inc.

**REGISTER 28-12: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 30000Ch)**

| R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 | R/C-1 |
|---|---|---|---|---|---|---|---|
| EBTR7(1,3) | EBTR6(1,3) | EBTR5(1,3) | EBTR4(1,3) | EBTR3(3) | EBTR2(3) | EBTR1(3) | EBTR0(3) |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | C = Clearable 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 **EBTR7:** Table Read Protection bit(1,3)
1 = Block 7 is not protected from table reads executed in other blocks(2)
0 = Block 7 is protected from table reads executed in other blocks(2)

bit 6 **EBTR6:** Table Read Protection bit(1,3)
1 = Block 6 is not protected from table reads executed in other blocks(2)
0 = Block 6 is protected from table reads executed in other blocks(2)

bit 5 **EBTR5:** Table Read Protection bit(1,3)
1 = Block 5 is not protected from table reads executed in other blocks(2)
0 = Block 5 is protected from table reads executed in other blocks(2)

bit 4 **EBTR4:** Table Read Protection bit(1,3)
1 = Block 4 is not protected from table reads executed in other blocks(2)
0 = Block 4 is protected from table reads executed in other blocks(2)

bit 3 **EBTR3:** Table Read Protection bit(3)
1 = Block 3 is not protected from table reads executed in other blocks(2)
0 = Block 3 is protected from table reads executed in other blocks(2)

bit 2 **EBTR2**: Table Read Protection bit(3)
1 = Block 2 is not protected from table reads executed in other blocks(2)
0 = Block 2 is protected from table reads executed in other blocks(2)

bit 1 **EBTR1:** Table Read Protection bit(3)
1 = Block 1 is not protected from table reads executed in other blocks(2)
0 = Block 1 is protected from table reads executed in other blocks(2)

bit 0 **EBTR0:** Table Read Protection bit(3)
1 = Block 0 is not protected from table reads executed in other blocks(2)
0 = Block 0 is protected from table reads executed in other blocks(2)

**Note 1:** This bit is available only on PIC18F67K22 and PIC18F87K22 devices.
**2:** For the memory size of the blocks, see Figure 28-6
**3:** Enable the corresponding CPx bit to protect the block from external read operations.

**REGISTER 28-13: CONFIG7H: CONFIGURATION REGISTER 7 HIGH (BYTE ADDRESS 30000Dh)**

| U-0 | R/C-1 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
|---|---|---|---|---|---|---|---|
| — | EBTRB(2) | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

| **Legend:** | C = Clearable 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 **Unimplemented:** Read as '0'

bit 6 **EBTRB:** Boot Block Table Read Protection bit(2)

1 = Boot block is not protected from table reads executed in other blocks(1)
0 = Boot block is protected from table reads executed in other blocks(1)

bit 5-0 **Unimplemented:** Read as '0'

**Note 1:** For the memory size of the blocks, see Figure 28-6.

**2:** Enable the corresponding CPx bit to protect the block from external read operations.


© 2009-2018 Microchip Technology Inc.

**REGISTER 28-14: DEVID1: DEVICE ID REGISTER 1 FOR THE PIC18F87K22 FAMILY**

| 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

Devices with DEV<10:3> of '0101 0010' (see DEVID2):
010 = PIC18F65K22
000 = PIC18F66K22
101 = PIC18F85K22
011 = PIC18F86K22

Devices with DEV<10:3> of '0101 0001':
000 = PIC18F67K22
010 = PIC18F87K22

bit 4-0 **REV<4:0>:** Revision ID bits

These bits are used to indicate the device revision.

**REGISTER 28-15: DEVID2: DEVICE ID REGISTER 2 FOR THE PIC18F87K22 FAMILY**

| R | R | R | R | R | R | R | R |
|---|---|---|---|---|---|---|---|
| DEV10(1) | DEV9(1) | DEV8(1) | DEV7(1) | DEV6(1) | DEV5(1) | DEV4(1) | DEV3(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 7-0 **DEV<10:3>:** Device ID bits(1)

These bits are used with the DEV<2:0> bits in the Device ID Register 1 to identify the part number.
0101 0010 = PIC18F65K22, PIC18F66K22, PIC18F85K22 and PIC18F86K22
0101 0001 = PIC18F67K22 and PIC18F87K22

**Note 1:** These values for DEV<10:3> may be shared with other devices. The specific device is always identified by using the entire DEV<10:0> bit sequence.

## 28.2 Watchdog Timer (WDT)

For the PIC18F87K22 family of devices, the WDT is driven by the LF-INTOSC source. 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 LF-INTOSC 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 multiplexer, controlled by bits in Configuration Register 2H. Available periods range from 4 ms to 4,194 seconds (about one hour). The WDT and postscaler are cleared when any of the following events occur: a SLEEP or CLRWDT instruction is executed, the IRCF bits (OSCCON<6:4>) are changed or a clock failure has occurred.

The WDT can be operated in one of four modes as determined by the WDTEN<1:0> (CONFIG2H<1:0> bits. The four modes are:

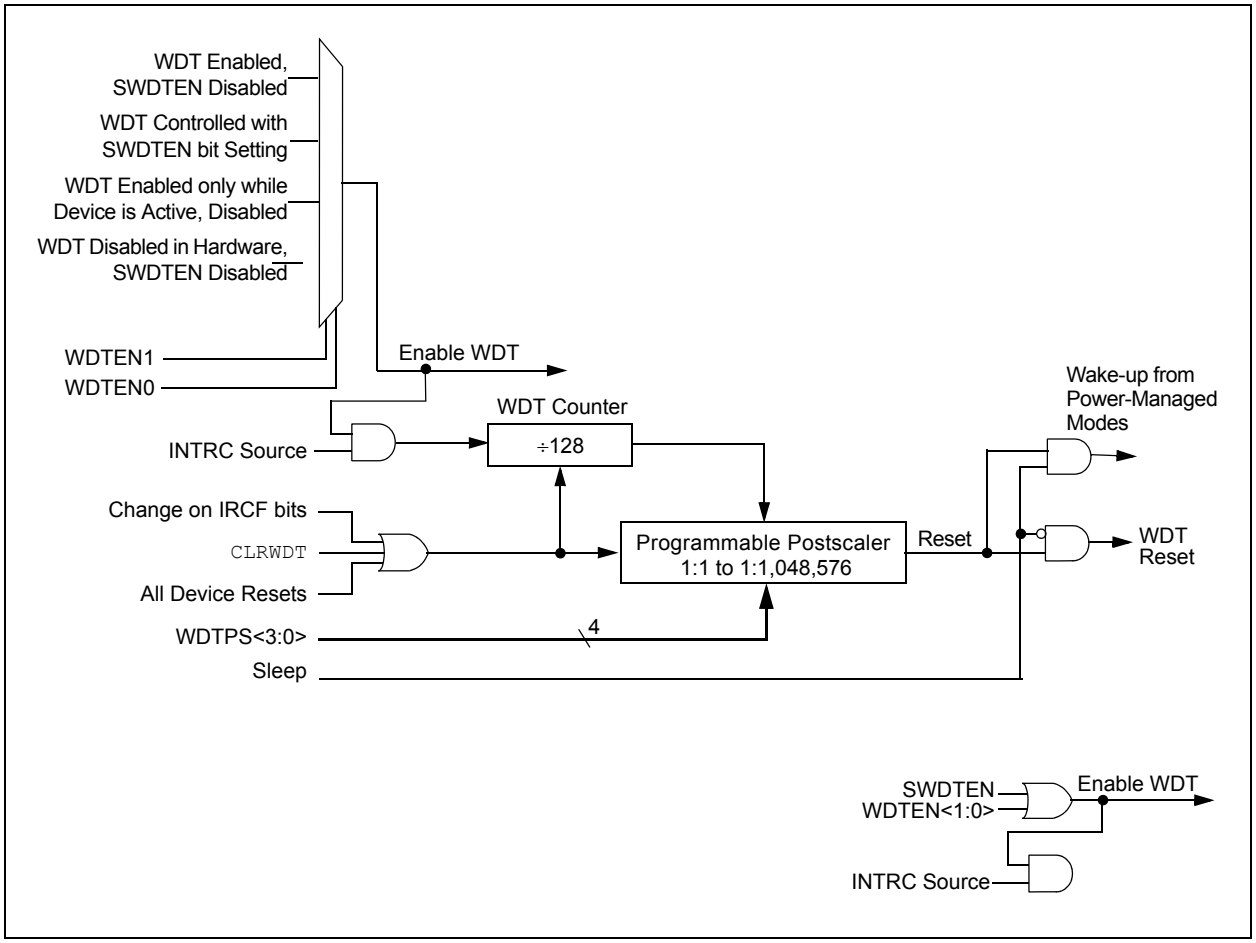
- WDT Enabled
- WDT Disabled
- WDT under Software Control, SWDTEN (WDTCON<0>)
- WDT
  - Enabled during normal operation
  - Disabled during Sleep

**Note 1:** The CLRWDT and SLEEP instructions clear the WDT and postscaler counts when executed.

**2:** Changing the setting of the IRCF bits (OSCCON<6:4>) clears the WDT and postscaler counts.

**3:** When a CLRWDT instruction is executed, the postscaler count will be cleared.

**FIGURE 28-1: WDT BLOCK DIAGRAM**





© 2009-2018 Microchip Technology Inc.

### 28.2.1 CONTROL REGISTER

Register 28-16 shows the WDTCON register. This is a readable and writable register which contains a control bit that allows software to override the WDT Enable Configuration bit, but only if the Configuration bit has disabled the WDT.

**REGISTER 28-16: WDTCON: WATCHDOG TIMER CONTROL REGISTER**

| R/W-0 | U-0 | R-x | R/W-0 | U-0 | R/W-0 | R/W-0 | R/W-0 |
|---|---|---|---|---|---|---|---|
| REGSLP | — | ULPLVL | SRETEN(2) | — | ULPEN | ULPSINK | 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:** Regulator Voltage Sleep Enable bit
1 = Regulator goes into Low-Power mode when device's Sleep mode is enabled
0 = Regulator stays in normal Operation mode when device's Sleep mode is activated

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

bit 5 **ULPLVL:** Ultra Low-Power Wake-up Output bit
Not valid unless ULPEN = 1.
1 = Voltage on RA0 pin > ~ 0.5V
0 = Voltage on RA0 pin < ~ 0.5V.

bit 4 **SRETEN:** Regulator Voltage Sleep Disable bit(2)
1 = If $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 0 and the regulator is enabled, the device goes into Ultra Low-Power mode in Sleep
0 = The regulator is on when the device's Sleep mode is enabled and the Low-Power mode is controlled by REGSLP

bit 3 **Unimplemented**: Read as '0'

bit 2 **ULPEN:** Ultra Low-Power Wake-up Module Enable bit
1 = Ultra Low-Power Wake-up module is enabled; ULPLVL bit indicates the comparator output
0 = Ultra Low-Power Wake-up module is disabled

bit 1 **ULPSINK:** Ultra Low-Power Wake-up Current Sink Enable bit
Not valid unless ULPEN = 1.
1 = Ultra Low-Power Wake-up current sink is enabled
0 = Ultra Low-Power Wake-up current sink is disabled

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 bits, WDTEN<1:0>, are enabled.
**2:** This bit is available only when ENVREG = 1 and $\overline{\text{RETEN}}$ = 0.

**TABLE 28-2: SUMMARY OF WATCHDOG TIMER REGISTERS**

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| RCON | IPEN | SBOREN | $\overline{\text{CM}}$ | $\overline{\text{RI}}$ | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |
| WDTCON | REGSLP | — | ULPLVL | SRETEN | — | ULPEN | ULPSINK | SWDTEN |

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

© 2009-2018 Microchip Technology Inc.

## 28.3 On-Chip Voltage Regulator

All of the PIC18F87K22 family devices power their core digital logic at a nominal 3.3V. For designs that are required to operate at a higher typical voltage, such as 5V, all family devices incorporate two on-chip regulators that allow the device to run its core logic from VDD. Those regulators are:

- Normal On-Chip Regulator
- Ultra Low-Power On-Chip Regulator

The hardware configuration of these regulators is the same and is explained in **Section 28.3.1 "Regulator Enable/Disable By Hardware"**. The regulators' only differences relate to when the device enters Sleep, as explained in **Section 28.3.2 "Operation of Regulator in Sleep"**.

### 28.3.1 REGULATOR ENABLE/DISABLE BY HARDWARE

The regulator can be enabled or disabled only by hardware. The regulator is controlled by the ENVREG pin and the VDDCORE/VCAP pin.

#### 28.3.1.1 Regulator Enable Mode

Tying VDD to the pin enables the regulator, which in turn, provides power to the core from the other VDD pins.

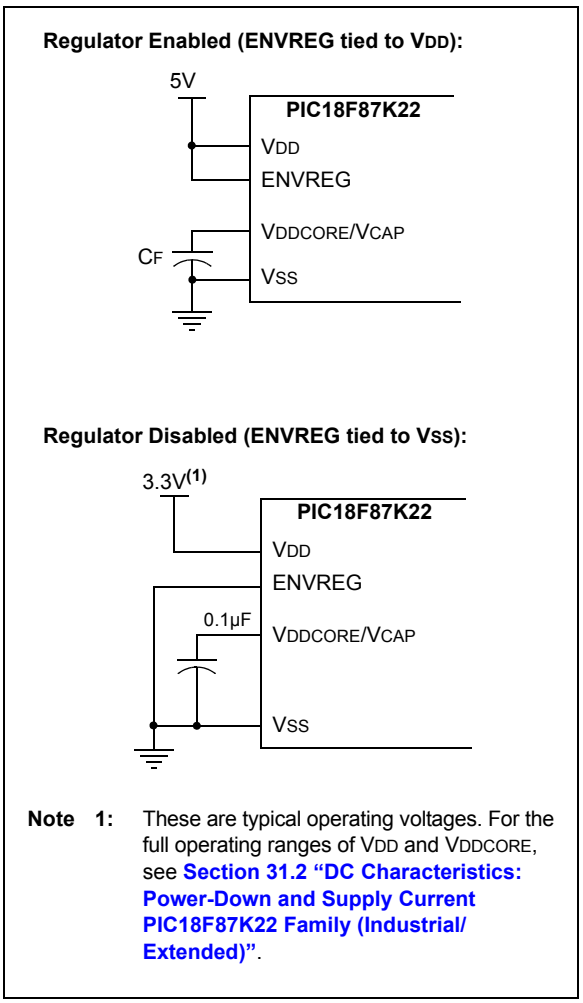
When the regulator is enabled, a low-ESR filter capacitor must be connected to the VDDCORE/VCAP pin (see Figure 28-2). This helps maintain the regulator's stability. The recommended value for the filter capacitor is given in **Section 31.2 "DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended)"**.

#### 28.3.1.2 Regulator Disable Mode

If the regulator is disabled by connecting VSS to the ENVREG pin, the power to the core is supplied directly by VDD. The voltage levels for VDD must not exceed the specified VDDCORE levels. In Regulator Disabled mode, a 0.1 µF capacitor should be connected to the VDDCORE/VCAP pin (see Figure 28-2).

When the regulator is being used, the overall voltage budget is very tight. The regulator should operate the device down to 1.8V. When VDD drops below 3.3V, the regulator no longer regulates, but the output voltage follows the input until VDD reaches 1.8V. Below this voltage, the output of the regulator output may drop to 0V.

**FIGURE 28-2: CONNECTIONS FOR THE ON-CHIP REGULATOR**




**Note 1:** These are typical operating voltages. For the full operating ranges of VDD and VDDCORE, see **Section 31.2 "DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended)"**.


© 2009-2018 Microchip Technology Inc.

### 28.3.2 OPERATION OF REGULATOR IN SLEEP

The difference in the two regulators' operation arises with Sleep mode. The ultra low-power regulator gives the device the lowest current in the Regulator Enabled mode.

The on-chip regulator can go into a lower power mode, when the device goes to Sleep, by setting the REGSLP bit (WDTCON<7>). This puts the regulator in a Standby mode so that the device consumes much less current.

The on-chip regulator can also go into the Ultra Low-Power mode, which consumes the lowest current possible with the regulator enabled. This mode is controlled by the $\overline{\text{RETEN}}$ bit (CONFIG1L<0>) and SRETEN bit (WDTCON<4>).

The various modes of regulator operation are shown in Table 28-3.

When the ultra low-power regulator is in Sleep mode, the internal reference voltages in the chip will be shut off and any interrupts referring to the internal reference will not wake up the device. If the BOR or LVD is enabled, the regulator will keep the internal references on and the lowest possible current will not be achieved.

When using the ultra low-power regulator in Sleep mode, the device will take about 250 μs, typical, to start executing the code after it wakes up.

**TABLE 28-3: SLEEP MODE REGULATOR SETTINGS(1)**

| Regulator | Power Mode | VREGSLP WDTCON<7> | SRETEN WDTCON<4> | $\overline{\text{RETEN}}$ CONFIG1L<0> |
|---|---|---|---|---|
| Enabled | Normal Operation (Sleep) | 0 | x | 1 |
| Enabled | Low-Power mode (Sleep) | 1 | x | 1 |
| Enabled | Normal Operation (Sleep) | 0 | 0 | x |
| Enabled | Low-Power mode (Sleep) | 1 | 0 | x |
| Enabled | Ultra Low-Power mode (Sleep) | x | 1 | 0 |

**Note 1:** x = Indicates that VIT status is invalid.

## 28.4 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 INTOSC (LF-INTOSC, MF-INTOSC, HF-INTOSC) 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 LP, XT or HS (Crystal-Based modes). Other sources do not require an OST start-up delay; for these, 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.

To use a higher clock speed on wake-up, the INTOSC or postscaler clock sources can be selected to provide a higher clock speed by setting bits, IRCF<2:0>, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF<2:0> bits prior to entering Sleep mode.

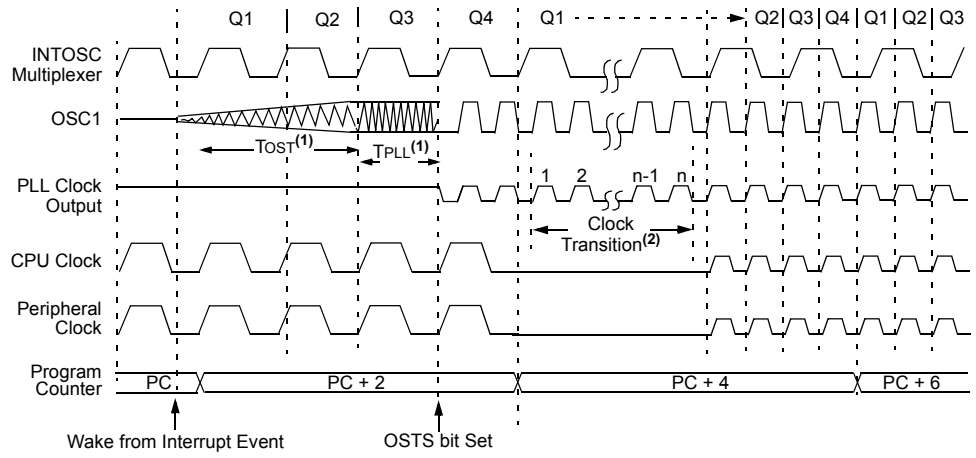
In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

### 28.4.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTOSC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power-managed modes, including multiple `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.

**FIGURE 28-3: TIMING TRANSITION FOR TWO-SPEED START-UP (INTOSC TO HSPLL)**




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

**2:** Clock transition typically occurs within 2-4 TOSC.

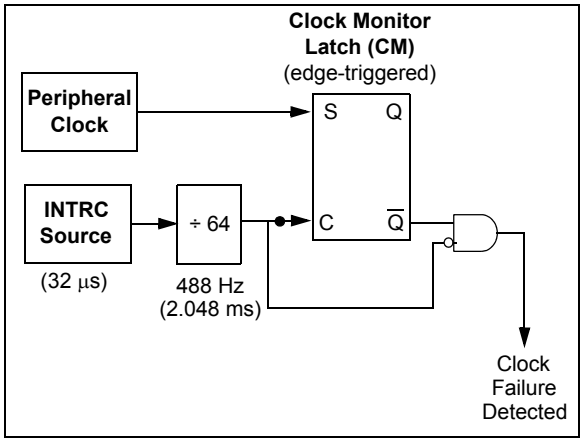

© 2009-2018 Microchip Technology Inc.

## 28.5 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 LF-INTOSC 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 28-4) is accomplished by creating a sample clock signal, which is the output from the LF-INTOSC, 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 28-4: 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 28-5). This causes the following:

- The FSCM generates an oscillator fail interrupt by setting bit, OSCFIF (PIR2<7>)
- The device clock source switches 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 shut-down. See **Section 4.1.4 "Multiple Sleep Commands"** and **Section 28.4.1 "Special Considerations for Using Two-Speed Start-up"** for more details.

To use a higher clock speed on wake-up, the INTOSC or postscaler clock sources can be selected to provide a higher clock speed by setting bits, IRCF<2:0>, immediately after Reset. For wake-ups from Sleep, the INTOSC or postscaler clock sources can be selected by setting the IRCF<2:0> bits prior to entering Sleep mode.

The FSCM will detect only failures of the primary or secondary clock sources. If the internal oscillator block fails, no failure would be detected nor would any action be possible.

### 28.5.1 FSCM AND THE WATCHDOG TIMER

Both the FSCM and the WDT are clocked by the INTOSC oscillator. Since the WDT operates with a separate divider and counter, disabling the WDT has no effect on the operation of the INTOSC oscillator when the FSCM is enabled.

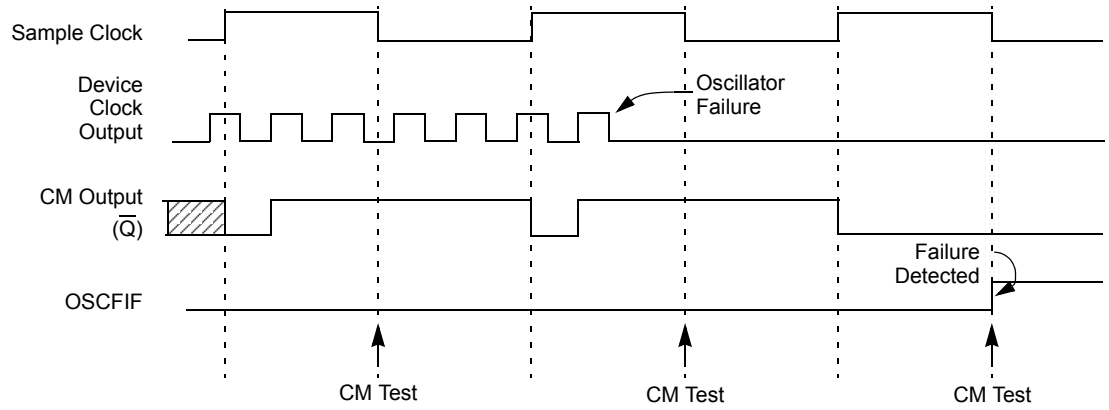
As already noted, the clock source is switched to the INTOSC clock when a clock failure is detected. Depending on the frequency selected by the IRCF<2:0> bits, this may mean a substantial change in the speed of code execution. If the WDT is enabled with a small prescale value, a decrease in clock speed 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.

### 28.5.2 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 1H (with any required start-up delays that are required for the Oscillator mode, such as the OST or PLL timer). The INTOSC multiplexer 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 INTOSC multiplexer. The OSCCON register will remain in its Reset state until a power-managed mode is entered.

**FIGURE 28-5: 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.

### 28.5.3 FSCM INTERRUPTS IN POWER-MANAGED MODES

By entering a power-managed mode, the clock multiplexer selects the clock source selected by the OSCCON register. Fail-Safe Clock Monitoring of the power-managed clock source resumes in the power-managed mode.

If an oscillator failure occurs during power-managed operation, the subsequent events depend on whether or not the Oscillator Failure Interrupt Flag bit is enabled (OSCFIF = 1). If enabled, code execution will be clocked by the INTOSC multiplexer. An automatic transition back to the failed clock source will not occur.

If the interrupt is disabled, subsequent interrupts while in Idle mode will cause the CPU to begin executing instructions while being clocked by the INTOSC source.

### 28.5.4 POR OR WAKE 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 EC, RC or INTRC modes, monitoring can begin immediately following these events.

For Oscillator modes involving a crystal or resonator (HS, HSPLL, LP or XT), the situation is somewhat different. Since the oscillator may require a start-up time considerably longer than the FCSM 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 (when the OST and PLL timers have timed out).

This is identical to Two-Speed Start-up mode. Once the primary clock is stable, the INTOSC returns to its role as the FSCM source.

**Note:** The same logic that prevents false oscillator failure interrupts on POR, or wake from Sleep, also prevents 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 28.4.1 "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.


© 2009-2018 Microchip Technology Inc.

## 28.6 Program Verification and Code Protection

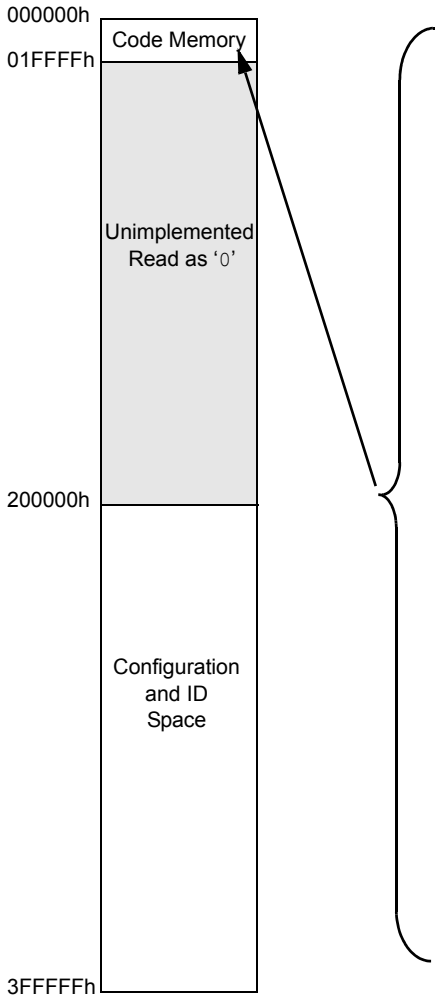
The user program memory is divided into four blocks for the PIC18FX5K22 and PIC18FX6K22 devices, and eight blocks for PIC18FX7K22 devices. One of these is a boot block of 1 or 2 Kbytes. The remainder of the memory is divided into blocks on binary boundaries.

Each of the blocks has three code protection bits associated with them. They are:

- Code-Protect bit (CPx)
- Write-Protect bit (WRTx)
- External Block Table Read bit (EBTRx)

Figure 28-6 shows the program memory organization for 48, 64, 96 and 128 Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 28-4.

**FIGURE 28-6: CODE-PROTECTED PROGRAM MEMORY FOR THE PIC18F87K22 FAMILY(1)**




**Device/Memory Size(2)**

| PIC18FX7K22 | | PIC18FX6K22 | | PIC18FX5K22 | | |
|---|---|---|---|---|---|---|
| BBSIZ = 1 | BBSIZ = 0 | BBSIZ = 1 | BBSIZ = 0 | BBSIZ = 1 | BBSIZ = 0 | Address |
| Boot Block 2 kW | Boot Block | Boot Block 2 kW | Boot Block | Boot Block 2 kW | Boot Block | 0000h |
| | Block 0 7 kW | | Block 0 7 kW | | Block 0 3 kW | 0800h |
| Block 0 6 kW | | Block 0 6 kW | | Block 0 2 kW | | 1000h 17FFh |
| | | | | Block 1 4 kW | Block 1 4 kW | 1800 3FFF |
| Block 1 8 kW | Block 1 8 kW | Block 1 8 kW | Block 1 8 kW | Block 2 4 kW | Block 2 4 kW | 4000h 5FFFh |
| | | | | Block 3 4 kW | Block 3 4 kW | 6000h 7FFF |
| Block 2 8 kW | Block 2 8 kW | Block 2 8 kW | Block 2 8 kW | | | 8000h BFFFh |
| Block 3 8 kW | Block 3 8 kW | Block 3 8 kW | Block 3 8 kW | | | C000h FFFFh |
| Block 4 8 kW | Block 4 8 kW | | | | | 10000h 13FFFh |
| Block 5 8 kW | Block 5 8 kW | | | | | 14000h 17FFFh |
| Block 6 8 kW | Block 6 8 kW | | | | | 18000h 1BFFFh |
| Block 7 8 kW | Block 7 8 kW | | | | | 1C000h 1FFFFh |

**Note 1:** Sizes of memory areas are not to scale.

**2:** Boot block size is determined by the BBSIZ0 bit (CONFIG4L<4>).

**TABLE 28-4: SUMMARY OF CODE PROTECTION REGISTERS**

| File Name | | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 300008h | CONFIG5L | CP7(1) | CP6(1) | CP5(1) | CP4(1) | CP3 | CP2 | CP1 | CP0 |
| 300009h | CONFIG5H | CPD | CPB | — | — | — | — | — | — |
| 30000Ah | CONFIG6L | WRT7(1) | WRT6(1) | WRT5(1) | WRT4(1) | WRT3 | WRT2 | WRT1 | WRT0 |
| 30000Bh | CONFIG6H | WRTD | WRTB | WRTC | — | — | — | — | — |
| 30000Ch | CONFIG7L | EBTR7(1) | EBTR6(1) | EBTR5(1) | EBTR4(1) | EBTR3 | EBTR2 | EBTR1 | EBTR0 |
| 30000Dh | CONFIG7H | — | EBTRB | — | — | — | — | — | — |

**Legend:** Shaded cells are unimplemented.

**Note 1:** This bit is available only on the PIC18F67K22 and PIC18F87K22 devices.

### 28.6.1 PROGRAM MEMORY CODE PROTECTION

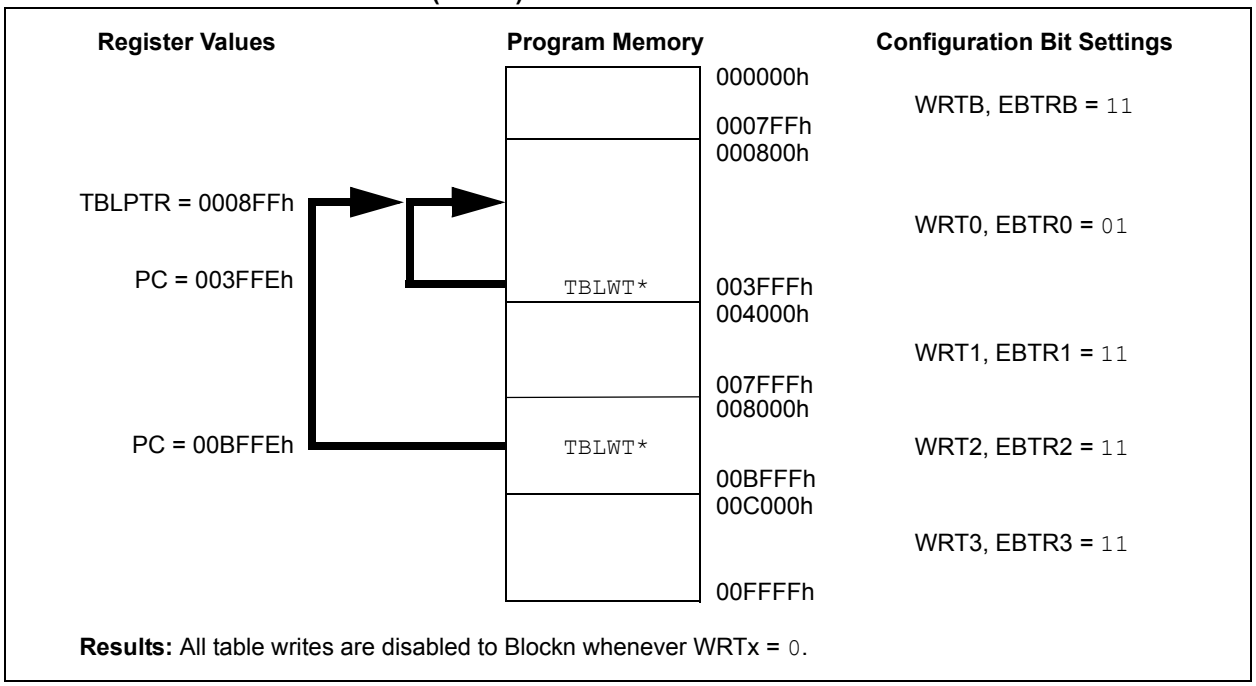
The program memory may be read to, or written from, any location using the table read and table write instructions. The Device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In normal Execution mode, the CPx bits have no direct effect. CPx bits inhibit external reads and writes. A block of user memory may be protected from table writes if the WRTx Configuration bit is '0'.

The EBTRx bits control table reads. For a block of user memory, with the EBTRx bit set to '0', a table read instruction that executes from within that block is allowed to read. A table read instruction that executes from a location outside of that block is not allowed to read and will result in reading '0's. Figures 28-7 through 28-9 illustrate table write and table read protection.

**Note:** Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full chip erase or block erase function. The full chip erase and block erase functions can only be initiated via ICSP or an external programmer. Refer to the device programming specification for more information.

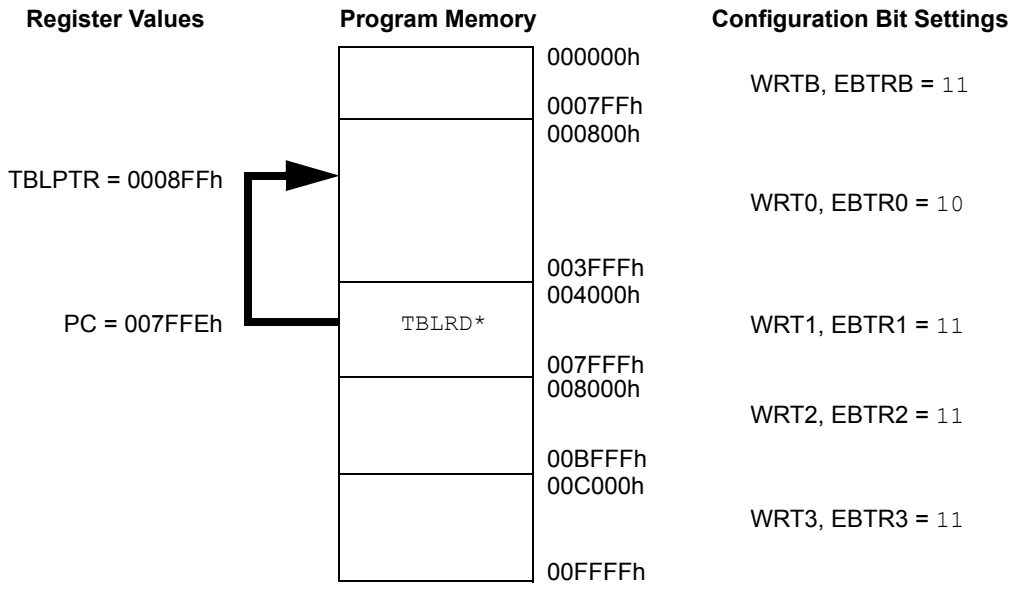
**FIGURE 28-7: TABLE WRITE (WRTx) DISALLOWED**




**Results:** All table writes are disabled to Blockn whenever WRTx = 0.

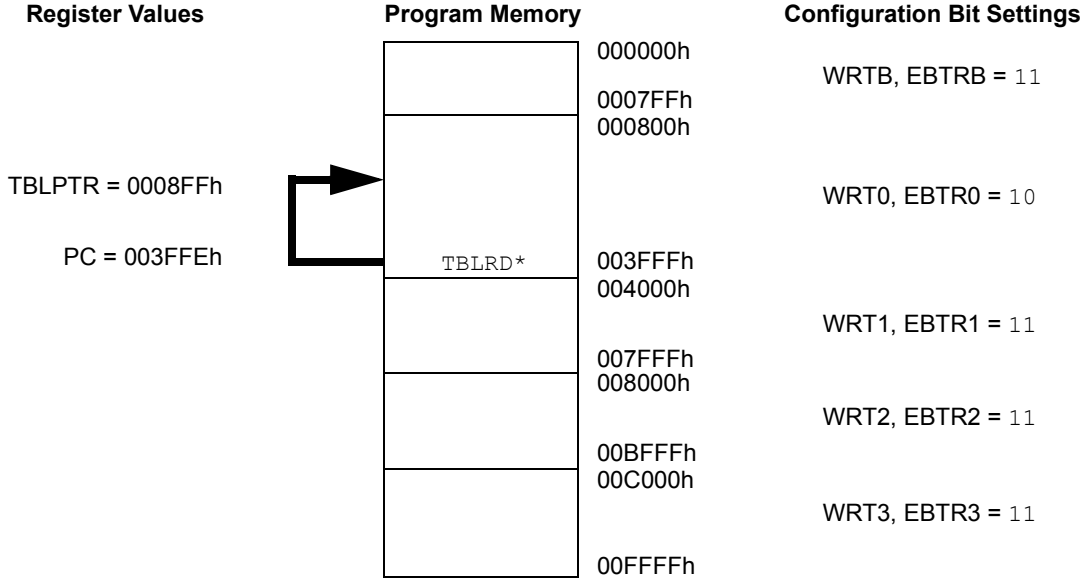

© 2009-2018 Microchip Technology Inc.

**FIGURE 28-8: EXTERNAL BLOCK TABLE READ (EBTRx) DISALLOWED**




**Results:** All table reads from external blocks to Blockn are disabled whenever EBTRx = 0. The TABLAT register returns a value of '0'.

**FIGURE 28-9: EXTERNAL BLOCK TABLE READ (EBTRx) ALLOWED**




**Results:** Table reads are permitted within Blockn, even when EBTRBx = 0.
The TABLAT register returns the value of the data at the location, TBLPTR.

### 28.6.2 DATA EEPROM CODE PROTECTION

The entire data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of data EEPROM. WRTD inhibits internal and external writes to data EEPROM. The CPU can always read data EEPROM under normal operation, regardless of the protection bit settings.

### 28.6.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In normal Execution mode, the WRTC bit is readable only. WRTC can only be written via ICSP or an external programmer.

## 28.7 ID Locations

Eight memory locations (200000h-200007h) are designated as ID locations, where the user can store checksum or other code identification numbers. These locations are both readable and writable, during normal execution, through the `TBLRD` and `TBLWT` instructions or during program/verify. The ID locations can be read when the device is code-protected.

## 28.8 In-Circuit Serial Programming

The PIC18F87K22 family of devices 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.

For the various Programming modes, see the device programming specification.

## 28.9 In-Circuit Debugger

When the $\overline{\text{DEBUG}}$ Configuration bit is programmed to a '0', the In-Circuit Debugger (ICD) 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 28-5 shows which resources are required by the background debugger.

**TABLE 28-5: DEBUGGER RESOURCES**

| | |
|---|---|
| **I/O Pins:** | RB6, RB7 |
| **Stack:** | Two levels |
| **Program Memory:** | 512 bytes |
| **Data Memory:** | 10 bytes |

To use the In-Circuit Debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to $\overline{\text{MCLR}}$/RG5/$V_{PP}$, $V_{DD}$, $V_{SS}$, RB7 and RB6. This will interface to the In-Circuit Debugger module, available from Microchip or one of the third-party development tool companies.


© 2009-2018 Microchip Technology Inc.

## 29.0 INSTRUCTION SET SUMMARY

The PIC18F87K22 family of devices incorporates 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.

### 29.1 Standard Instruction Set

The standard PIC18 MCU instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC MCU 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 29-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 29-1 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 zero, the result is placed in the WREG register. If ‘d’ is one, 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 29-1 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 29-2, lists the standard instructions recognized by the Microchip MPASM™ Assembler.

**Section 29.1.1 “Standard Instruction Set”** provides a description of each instruction.

**TABLE 29-1: OPCODE FIELD DESCRIPTIONS**

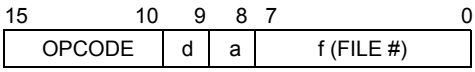
| Field | Description |
|---|---|
| a | RAM access bit:<br>a = 0: RAM location in Access RAM (BSR register is ignored)<br>a = 1: RAM bank is specified by BSR register |
| bbb | Bit address within an 8-bit file register (0 to 7). |
| BSR | Bank Select Register. Used to select the current RAM bank. |
| 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<br><br>*<br>*+<br>*–<br>+* | The mode of the TBLPTR register for the table read and table write instructions.<br>Only used with table read and table write instructions:<br>No Change to register (such as TBLPTR with table reads and writes)<br>Post-Increment register (such as TBLPTR with table reads and writes)<br>Post-Decrement register (such as TBLPTR with table reads and writes)<br>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 an Indexed Address. |
| (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). |


© 2009-2018 Microchip Technology Inc.

**FIGURE 29-1: GENERAL FORMAT FOR INSTRUCTIONS**

**Byte-oriented** file register operations

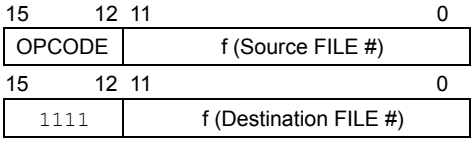
**Example Instruction**




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

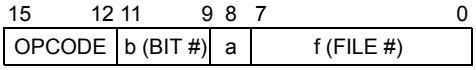
**Byte to Byte** move operations (2-word)




MOVFF MYREG1, MYREG2

f = 12-bit file register address

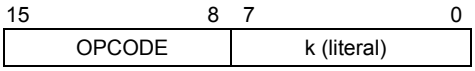
**Bit-oriented** file register operations




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

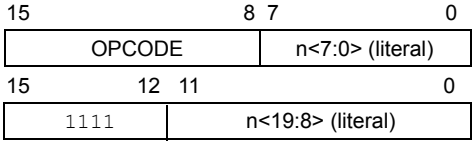



MOVLW 7Fh

k = 8-bit immediate value

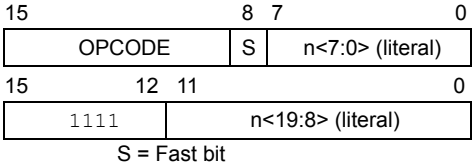
**Control** operations

**CALL, GOTO and Branch** operations



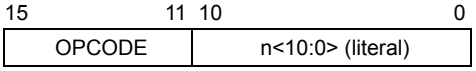

GOTO Label

n = 20-bit immediate value

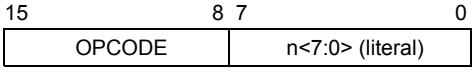



CALL MYFUNC

S = Fast bit




BRA MYFUNC




BC MYFUNC

**TABLE 29-2: PIC18F87K22 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<br>$f_d$ (destination) 2nd word | 2 | 1100 ffff ffff ffff<br>1111 ffff ffff ffff | None | |
| 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.


© 2009-2018 Microchip Technology Inc.

**TABLE 29-2: PIC18F87K22 FAMILY INSTRUCTION SET (CONTINUED)**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word MSb LSb | Status Affected | Notes |
|---|---|---|---|---|---|---|
| **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 29-2: PIC18F87K22 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.


© 2009-2018 Microchip Technology Inc.

### 29.1.1 STANDARD INSTRUCTION SET

**ADDLW** **ADD Literal to W**

Syntax: ADDLW k

Operands: $0 \le k \le 255$

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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

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 $f \le 95$ (5Fh). See **Section 29.2.3 "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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) + (f) + (C) → dest

Status Affected: N,OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0010 | 00da | ffff | ffff |

Description: Add W, the Carry flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 "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: $0 \le k \le 255$

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


© 2009-2018 Microchip Technology Inc.

**ANDWF** **AND W with f**

Syntax: ANDWF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

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'.

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 29.2.3 "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: $-128 \le n \le 127$

Operation: if Carry bit is '1',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0010 | nnnn | nnnn |
|---|---|---|---|

Description: If the Carry bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be 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 29.2.3 "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)


© 2009-2018 Microchip Technology Inc.

**BNC** **Branch if Not Carry**

Syntax: BNC n

Operands: -128 ≤ n ≤ 127

Operation: if Carry bit is '0',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0011 | nnnn | nnnn |
|---|---|---|---|

Description: If the Carry bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | 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',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| 1110 | 0111 | nnnn | nnnn |
|---|---|---|---|

Description: If the Negative bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | 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',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0101 | nnnn | nnnn |

Description: If the Overflow bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| 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',
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 | 0001 | nnnn | nnnn |

Description: If the Zero bit is '0', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

| 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)


© 2009-2018 Microchip Technology Inc.

**BRA** **Unconditional Branch**

Syntax: BRA n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 + 2n → PC

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1101 | 0nnn | nnnn | nnnn |

Description: Add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

| 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 ≤ f ≤ 255
0 ≤ b ≤ 7
a ∈ [0,1]

Operation: 1 → f<b>

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1000 | bbba | ffff | ffff |

Description: Bit 'b' in register 'f' is set.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 29.2.3 "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: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

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 $f \le 95$ (5Fh). See **Section 29.2.3 "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: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

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 $f \le 95$ (5Fh). See **Section 29.2.3 "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)


© 2009-2018 Microchip Technology Inc.

**BTG** **Bit Toggle f**

Syntax: BTG f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b < 7$
$a \in [0,1]$

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.

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 29.2.3 “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’,
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| 1110 | 0100 | nnnn | nnnn |
|---|---|---|---|

Description: If the Overflow bit is ‘1’, then the program will branch.

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 ≤ n ≤ 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 ≤ k ≤ 1048575
s ∈ [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


© 2009-2018 Microchip Technology Inc.

**CLRF** **Clear f**

Syntax: CLRF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: 000h → f,
1 → Z

Status Affected: Z

Encoding:

| 0110 | 101a | ffff | ffff |
|---|---|---|---|

Description: Clears the contents of the specified register.

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 29.2.3 "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,
000h → WDT postscaler,
1 → $\overline{TO}$,
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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $\bar{f} \rightarrow$ 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'.

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 29.2.3 "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: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0110 | 001a | ffff | ffff |

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.

If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 "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    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)


© 2009-2018 Microchip Technology Inc.

**CPFSGT** **Compare f with W, Skip if f > W**

Syntax: CPFSGT f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) > (W)
(unsigned comparison)

Status Affected: None

Encoding:

| 0110 | 010a | ffff | ffff |
|---|---|---|---|

Description: Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction.

If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 "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      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: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) < (W)
(unsigned comparison)

Status Affected: None

Encoding:

| 0110 | 000a | ffff | ffff |
|---|---|---|---|

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.

If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

Words: 1

Cycles: 1(2)
**Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read 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
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>

If [W<7:4> > 9] or [C = 1], then
(W<7:4>) + 6 → W<7:4>;
C = 1;
else
(W<7:4>) → W<7:4>

Status Affected: C

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0000 | 0111 |

Description: DAW adjusts the 8-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 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [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'.

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 29.2.3 "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


© 2009-2018 Microchip Technology Inc.

**DECFSZ** **Decrement f, Skip if 0**

Syntax: DECFSZ f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
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'.

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 ≤ 95 (5Fh). See **Section 29.2.3 "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    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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest,
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'.

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 ≤ 95 (5Fh). See **Section 29.2.3 "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    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)

© 2009-2018 Microchip Technology Inc.

**GOTO** **Unconditional Branch**

Syntax: GOTO k

Operands: $0 \le k \le 1048575$

Operation: $k \rightarrow PC<20:1>$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (k<7:0>) | 1110 | $k_7kkk$ | $kkkk_0$ |
| 2nd word(k<19:8>) | 1111 | $k_{19}kkk$ | $kkkk_8$ |

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 29.2.3 "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


© 2009-2018 Microchip Technology Inc.

**INCFSZ** **Increment f, Skip if 0**

Syntax: INCFSZ f {,d {,a}}

Operands: 0 ≤ f ≤ 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 ≤ 95 (5Fh). See **Section 29.2.3 "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 ≤ f ≤ 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 ≤ 95 (5Fh). See **Section 29.2.3 "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: $0 \le k \le 255$

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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

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'.

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 29.2.3 "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


© 2009-2018 Microchip Technology Inc.

**LFSR** **Load FSR**

Syntax: LFSR f, k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1110 1111 | 1110 0000 | 00ff $k_7$kkk | $k_{11}$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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $f \rightarrow dest$

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 $f \le 95$ (5Fh). See **Section 29.2.3 "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: $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 1st word (source) | 1100 | ffff | ffff$_s$ |
| 2nd word (destin.) | 1111 | 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


© 2009-2018 Microchip Technology Inc.

**MOVLW** **Move Literal to W**

Syntax: MOVLW k

Operands: $0 \le k \le 255$

Operation: $k \rightarrow W$

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: $0 \le f \le 255$
$a \in [0,1]$

Operation: $(W) \rightarrow f$

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 $f \le 95$ (5Fh). See **Section 29.2.3 “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

© 2009-2018 Microchip Technology Inc.

**MULLW** **Multiply Literal with W**

Syntax: MULLW k

Operands: $0 \le k \le 255$

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 the 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: $0 \le f \le 255$
$a \in [0,1]$

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 $f \le 95$ (5Fh). See **Section 29.2.3 "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


© 2009-2018 Microchip Technology Inc.

**NEGF** **Negate f**

Syntax: NEGF f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

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'.

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 29.2.3 "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 1111 | 0000 xxxx | 0000 xxxx | 0000 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


© 2009-2018 Microchip Technology Inc.

**RCALL** **Relative Call**

Syntax: RCALL n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 → TOS,
(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 $\overline{\text{MCLR}}$ Reset.

Status Affected: All

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 1111 | 1111 |

Description: This instruction provides a way to execute a $\overline{\text{MCLR}}$ 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,
1 → GIE/GIEH or PEIE/GIEL;
if s = 1,
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 0000 | 0001 | 000s |

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low-priority Global Interrupt Enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers W, STATUS and BSR. If 's' = 0, no update of these registers occurs.

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,
(TOS) → PC,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 0000 | 1100 | kkkk | kkkk |

Description: W is loaded with the 8-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


© 2009-2018 Microchip Technology Inc.

**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

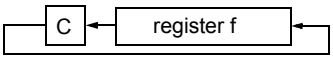
Encoding:

| | | | |
|---|---|---|---|
| 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 29.2.3 "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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<n>) → dest<n + 1>,
(f<7>) → dest<0>

Status Affected: N, Z

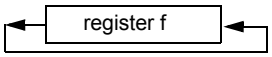
Encoding:

| | | | |
|---|---|---|---|
| 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 29.2.3 "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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<n>) → dest<n – 1>,
(f<0>) → C,
(C) → dest<7>

Status Affected: C, N, Z

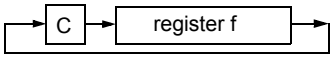
Encoding:

| | | | |
|---|---|---|---|
| 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 29.2.3 "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


© 2009-2018 Microchip Technology Inc.

**RRNCF** **Rotate Right f (No Carry)**

Syntax: RRNCF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

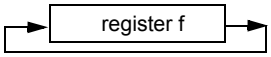
Encoding:

| | | | |
|---|---|---|---|
| 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 ≤ 95 (5Fh). See **Section 29.2.3 "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: $0 \le f \le 255$
$a \in [0,1]$

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 ≤ 95 (5Fh). See **Section 29.2.3 "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 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [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 ≤ 95 (5Fh). See **Section 29.2.3 "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


© 2009-2018 Microchip Technology Inc.

**SUBLW** **Subtract W from Literal**

Syntax: SUBLW k

Operands: $0 \le k \le 255$

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: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $(f) - (W) \rightarrow dest$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|---|---|---|---|
| 0101 | 11da | ffff | ffff |

Description: Subtract W from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

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 29.2.3 "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

© 2009-2018 Microchip Technology Inc.

**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) – ($\overline{C}$) → 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 ≤ 95 (5Fh). See **Section 29.2.3 "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>) → dest<7:4>,
(f<7:4>) → 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 ≤ 95 (5Fh). See **Section 29.2.3 "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
```


© 2009-2018 Microchip Technology Inc.

**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*,
(TABLAT) → Holding Register;
TBLPTR – No Change
if TBLWT*+,
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR
if TBLWT*-,
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register

Status Affected: None

Encoding:

| | | | |
|---|---|---|---|
| 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


© 2009-2018 Microchip Technology Inc.

**TSTFSZ** **Test f, Skip if 0**

Syntax: TSTFSZ f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

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.

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 29.2.3 "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    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: $0 \le k \le 255$

Operation: (W) .XOR. k → W

Status Affected: N, Z

Encoding:

| 0000 | 1010 | kkkk | kkkk |
|---|---|---|---|

Description: The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

| 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 ≤ f ≤ 255<br>d ∈ [0,1]<br>a ∈ [0,1] |
| Operation: | (W) .XOR. (f) → dest |
| Status Affected: | N, Z |

Encoding:

| | | | |
|---|---|---|---|
| 0001 | 10da | ffff | ffff |

Description: Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f'.

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 29.2.3 "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


© 2009-2018 Microchip Technology Inc.

## 29.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, the PIC18F87K22 family of devices also provides 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 29-3. Detailed descriptions are provided in **Section 29.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 29-1 (page 420) 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.

### 29.2.1 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 29.2.3.1 "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 29-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected |
|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | |
| 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 |

### 29.2.2 EXTENDED INSTRUCTION SET

**ADDFSR** **Add Literal to FSR**

Syntax: ADDFSR f, k

Operands: $0 \le k \le 63$
$f \in [0, 1, 2]$

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: $0 \le k \le 63$

Operation: FSR2 + k → FSR2,
(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.

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 `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).


© 2009-2018 Microchip Technology Inc.

**CALLW** **Subroutine Call Using WREG**

Syntax: CALLW

Operands: None

Operation: (PC + 2) → TOS,
(W) → PCL,
(PCLATH) → PCH,
(PCLATU) → 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.

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: $0 \le z_s \le 127$
$0 \le f_d \le 4095$

Operation: ((FSR2) + $z_s$) → $f_d$

Status Affected: None

Encoding:

| | | | | |
|---|---|---|---|---|
| 1st word (source) | 1110 | 1011 | 0zzz | $zzzz_s$ |
| 2nd word (destin.) | 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).

The MOVSF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an Indirect Addressing register, the value returned will be 00h.

Words: 2

Cycles: 2

Q Cycle Activity:

| 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) + $z_s$) → ((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: $0 \le k \le 255$

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


© 2009-2018 Microchip Technology Inc.

**SUBFSR** **Subtract Literal from FSR**

Syntax: SUBFSR f, k

Operands: $0 \le k \le 63$

$f \in [0, 1, 2]$

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: $0 \le k \le 63$

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)

### 29.2.3 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 29.2.3.1 "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.

#### 29.2.3.1 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.

### 29.2.4 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 PIC18F87K22 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.


© 2009-2018 Microchip Technology Inc.

**ADDWF** **ADD W to Indexed (Indexed Literal Offset mode)**

Syntax: ADDWF [k] {,d}

Operands: $0 \le k \le 95$
$d \in [0,1]$

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: $0 \le f \le 95$
$0 \le b \le 7$

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: $0 \le k \le 95$

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

© 2009-2018 Microchip Technology Inc.

### 29.2.5 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 PIC18F87K22 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.


© 2009-2018 Microchip Technology Inc.

## 30.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICkit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits and Starter Kits
- Third-party development tools

### 30.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

## 30.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

## 30.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

## 30.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

## 30.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

 © 2009-2018 Microchip Technology Inc.

## 30.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

## 30.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

## 30.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

## 30.9 PICkit 3 In-Circuit Debugger/ Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

## 30.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at $V_{DDMIN}$ and $V_{DDMAX}$ for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.

## 30.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

## 30.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

 © 2009-2018 Microchip Technology Inc.

## 31.0 ELECTRICAL CHARACTERISTICS

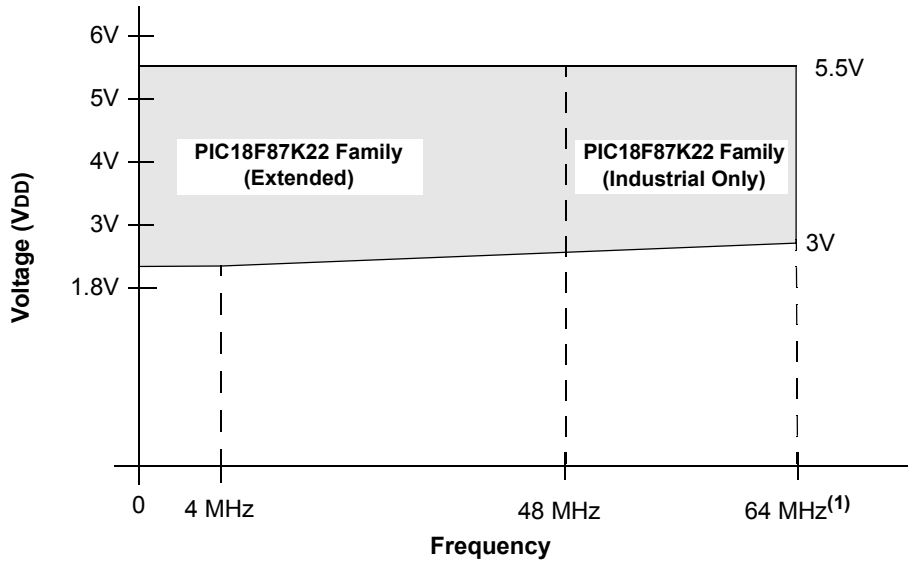
### Absolute Maximum Ratings(†)

Ambient temperature under bias ..... -40°C to +125°C
Storage temperature ..... -65°C to +150°C
Voltage on any digital only I/O pin with respect to $V_{SS}$ (except $V_{DD}$) ..... -0.3V to 7.5V
Voltage on $\overline{MCLR}$ with respect to $V_{SS}$ ..... 0.3V to 9.0V
Voltage on any combined digital and analog pin with respect to $V_{SS}$ (except $V_{DD}$ and $\overline{MCLR}$) ..... -0.3V to ($V_{DD}$ + 0.3V)
Voltage on $V_{DD}$ with respect to $V_{SS}$ (with regulator enabled) ..... -0.3V to 5.5V
Voltage on $V_{DD}$ with respect to $V_{SS}$ (with regulator disabled) ..... -0.3V to 3.6V
Total power dissipation **(Note 1)** ..... 1W
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 $V_I$ > $V_{DD}$) ..... ±20 mA
Output clamp current, $I_{OK}$ ($V_O$ < 0 or $V_O$ > $V_{DD}$) ..... ±20 mA
Maximum output current sunk by PORTA<7:6> and 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 PORTA<5:0> and any PORTF, PORTG and PORTH I/O pins ..... 2 mA
Maximum output current sourced by PORTA<7:6> and 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 PORTA<5:0> and any PORTF, PORTG and PORTH I/O pins ..... 2 mA
Maximum current sunk by all ports combined ..... 200 mA

**Note 1:** Power dissipation is calculated as follows:
$Pdis = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

† **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.

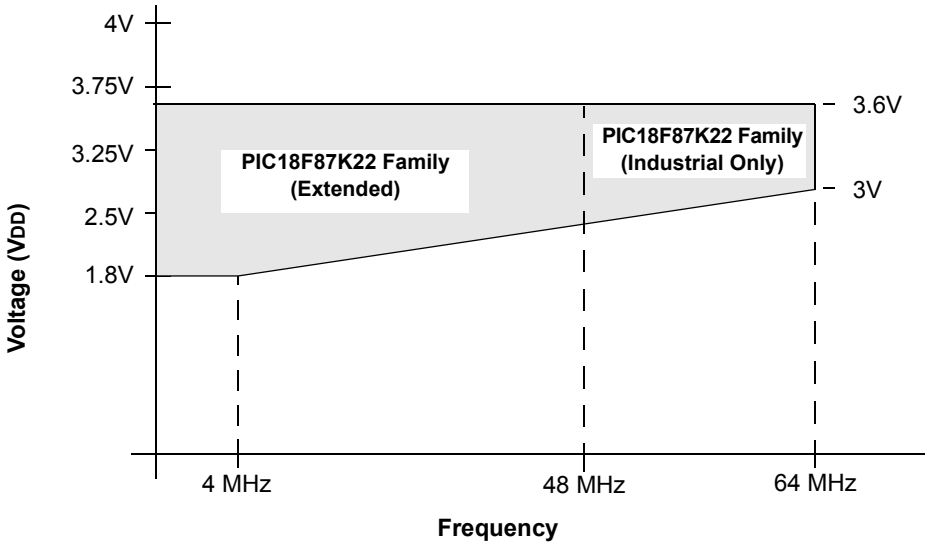
**FIGURE 31-1: VOLTAGE-FREQUENCY GRAPH, REGULATOR ENABLED (INDUSTRIAL/EXTENDED)(1)**




**Note 1:** FMAX = 25 MHz in 8-Bit External Memory mode. For VDD values, 1.8V to 3V, FMAX = (VDD – 1.72)/0.02 MHz.

**2:** FMAX = 64 MHz in all other modes. For VDD values, 1.8V to 3V, FMAX = (VDD – 1.72)/0.02 MHz.

**FIGURE 31-2: VOLTAGE-FREQUENCY GRAPH, REGULATOR DISABLED (INDUSTRIAL/EXTENDED)(1,2)**




**Note 1:** When the on-chip voltage regulator is disabled, VDD must be maintained so that VDD ≤ 3.6V.

**2:** For VDD values, 1.8V to 3V, FMAX = (VDD – 1.72)/0.02 MHz.


© 2009-2018 Microchip Technology Inc.

## 31.1 DC Characteristics: Supply Voltage PIC18F87K22 Family (Industrial/Extended)

| PIC18F87K22 Family (Industrial/Extended) | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| D001 | VDD | **Supply Voltage** | 1.8<br>1.8 | —<br>— | 3.6<br>5.5 | V<br>V | ENVREG tied to VSS<br>ENVREG tied to VDD |
| 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 Start Voltage** to Ensure Internal Power-on Reset Signal | — | — | 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 | BVDD | **Brown-out Reset Voltage (High/Medium/ Low-Power mode)**(2)<br>BORV<1:0> = 11(3)<br>BORV<1:0> = 10<br>BORV<1:0> = 01<br>BORV<1:0> = 00 | <br>1.69<br>1.88<br>2.53<br>2.82 | <br>1.8<br>2.0<br>2.7<br>3.0 | <br>1.91<br>2.12<br>2.86<br>3.18 | | |

**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:** The following values are taken in HP-BOR mode.

**3:** The device will operate normally until Brown-out Reset occurs, even though VDD may be below VDDMIN.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended)

| PIC18F87K22 Family (Industrial/Extended) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | |
| | **Power-Down Current (IPD)(1)** | | | | | |
| | All devices | 10 | 500 | nA | -40°C | VDD = 1.8V(4) (**Sleep** mode) Regulator Disabled |
| | | 20 | 500 | nA | +25°C | |
| | | 120 | 600 | nA | +60°C | |
| | | 630 | 1800 | nA | +85°C | |
| | | 4 | 9 | µA | +125ºC | |
| | All devices | 50 | 700 | nA | -40°C | VDD = 3.3V(4) (**Sleep** mode) Regulator Disabled |
| | | 60 | 700 | nA | +25°C | |
| | | 170 | 800 | nA | +60°C | |
| | | 700 | 2700 | nA | +85°C | |
| | | 5 | 11 | µA | +125ºC | |
| | All devices | 350 | 1300 | nA | -40°C | VDD = 5V(5) (**Sleep** mode) Regulator Enabled |
| | | 400 | 1400 | nA | +25°C | |
| | | 550 | 1500 | nA | +60°C | |
| | | 1350 | 4000 | nA | +85°C | |
| | | 6 | 12 | µA | +125ºC | |

**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, SOSC oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 1).

**5:** Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 0).

**6:** 48 MHz, maximum frequency at +125ºC.


© 2009-2018 Microchip Technology Inc.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended) (Continued)

| PIC18F87K22 Family (Industrial/Extended) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD)(2,3)** | | | | | | |
| | All devices | 5.3 | 10 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 31 kHz (**RC_RUN** mode, LF-INTOSC) |
| | | 5.5 | 10 | µA | +25°C | | |
| | | 5.5 | 10 | µA | +85°C | | |
| | | 12 | 24 | µA | +125ºC | | |
| | All devices | 10 | 15 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 10 | 16 | µA | +25°C | | |
| | | 11 | 17 | µA | +85°C | | |
| | | 15 | 35 | µA | +125ºC | | |
| | All devices | 70 | 180 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 80 | 185 | µA | +25°C | | |
| | | 90 | 190 | µA | +85°C | | |
| | | 200 | 500 | µA | +125ºC | | |
| | All devices | 410 | 850 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 1 MHz (**RC_RUN** mode, HF-INTOSC) |
| | | 410 | 800 | µA | +25°C | | |
| | | 410 | 830 | µA | +85°C | | |
| | | 700 | 1500 | µA | +125ºC | | |
| | All devices | 680 | 990 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 680 | 960 | µA | +25°C | | |
| | | 670 | 950 | µA | +85°C | | |
| | | 800 | 1700 | µA | +125ºC | | |
| | All devices | 760 | 1400 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 780 | 1400 | µA | +25°C | | |
| | | 800 | 1500 | µA | +85°C | | |
| | | 1200 | 2400 | µA | +125ºC | | |
| | All devices | 760 | 1300 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 4 MHz (**RC_RUN** mode, HF-INTOSC) |
| | | 760 | 1400 | µA | +25°C | | |
| | | 770 | 1500 | µA | +85°C | | |
| | | 800 | 1700 | µA | +125ºC | | |
| | All devices | 1.4 | 2.5 | mA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 1.4 | 2.5 | mA | +25°C | | |
| | | 1.4 | 2.5 | mA | +85°C | | |
| | | 1.5 | 3.0 | mA | +125ºC | | |
| | All devices | 1.5 | 2.7 | mA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 1.5 | 2.7 | mA | +25°C | | |
| | | 1.5 | 2.7 | mA | +85°C | | |
| | | 1.6 | 3.3 | mA | +125ºC | | |

**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, SOSC oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 1).

**5:** Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 0).

**6:** 48 MHz, maximum frequency at +125ºC.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended) (Continued)

| PIC18F87K22 Family (Industrial/Extended) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.(2,3)** | | | | | | |
| | All devices | 2.1 | 5.5 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 31 kHz (**RC_IDLE** mode, LF-INTOSC) |
| | | 2.1 | 5.7 | µA | +25°C | | |
| | | 2.2 | 6.0 | µA | +85°C | | |
| | | 10 | 20 | µA | +125ºC | | |
| | All devices | 3.7 | 7.5 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 3.9 | 7.8 | µA | +25°C | | |
| | | 3.9 | 8.5 | µA | +85°C | | |
| | | 12 | 24 | µA | +125ºC | | |
| | All devices | 70 | 180 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 80 | 190 | µA | +25°C | | |
| | | 80 | 200 | µA | +85°C | | |
| | | 200 | 420 | µA | +125ºC | | |
| | All devices | 330 | 650 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 1 MHz (**RC_IDLE** mode, HF-INTOSC) |
| | | 330 | 640 | µA | +25°C | | |
| | | 330 | 630 | µA | +85°C | | |
| | | 500 | 850 | µA | +125ºC | | |
| | All devices | 520 | 850 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 520 | 900 | µA | +25°C | | |
| | | 520 | 850 | µA | +85°C | | |
| | | 800 | 1200 | µA | +125ºC | | |
| | All devices | 590 | 940 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 600 | 960 | µA | +25°C | | |
| | | 620 | 990 | µA | +85°C | | |
| | | 1000 | 1400 | µA | +125ºC | | |
| | All devices | 470 | 770 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 4 MHz (**RC_IDLE** mode, internal HF-INTOSC) |
| | | 470 | 770 | µA | +25°C | | |
| | | 460 | 760 | µA | +85°C | | |
| | | 700 | 1000 | µA | +125ºC | | |
| | All devices | 800 | 1400 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 800 | 1350 | µA | +25°C | | |
| | | 790 | 1300 | µA | +85°C | | |
| | | 1100 | 1400 | µA | +125ºC | | |
| | All devices | 880 | 1600 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 890 | 1700 | µA | +25°C | | |
| | | 910 | 1800 | µA | +85°C | | |
| | | 1200 | 2200 | µA | +125ºC | | |

**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, SOSC oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{RETEN}$ (CONFIG1L<0>) = 1).

**5:** Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{RETEN}$ (CONFIG1L<0>) = 0).

**6:** 48 MHz, maximum frequency at +125ºC.


© 2009-2018 Microchip Technology Inc.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended) (Continued)

| PIC18F87K22 Family (Industrial/Extended) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.(2,3)** | | | | | | |
| | All devices | 130 | 390 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 1 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 130 | 390 | µA | +25°C | | |
| | | 130 | 390 | µA | +85°C | | |
| | | 250 | 500 | µA | +125ºC | | |
| | All devices | 270 | 790 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 270 | 790 | µA | +25°C | | |
| | | 270 | 790 | µA | +85°C | | |
| | | 400 | 900 | µA | +125ºC | | |
| | All devices | 430 | 990 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 450 | 980 | µA | +25°C | | |
| | | 460 | 980 | µA | +85°C | | |
| | | 600 | 1300 | µA | +125ºC | | |
| | All devices | 430 | 860 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 4 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 530 | 900 | µA | +25°C | | |
| | | 490 | 880 | µA | +85°C | | |
| | | 750 | 1600 | µA | +125ºC | | |
| | All devices | 850 | 1750 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 850 | 1700 | µA | +25°C | | |
| | | 850 | 1800 | µA | +85°C | | |
| | | 1150 | 2400 | µA | +125ºC | | |
| | All devices | 1.1 | 2.7 | mA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 1.1 | 2.6 | mA | +25°C | | |
| | | 1.1 | 2.6 | mA | +85°C | | |
| | | 2.0 | 4.0 | mA | +125ºC | | |
| | All devices | 12 | 19 | mA | -40°C | VDD = 3.3V(4) Regulator Disabled | FOSC = 64 MHz (**PRI_RUN** mode, EC oscillator) |
| | | 12 | 19 | mA | +25°C | | |
| | | 12 | 19 | mA | +85°C | | |
| | | 13 | 22 | mA | +125ºC(6) | | |
| | All devices | 13 | 20 | mA | -40°C | VDD = 5V(4) Regulator Enabled | |
| | | 13 | 20 | mA | +25°C | | |
| | | 13 | 20 | mA | +85°C | | |
| | | 14 | 23 | mA | +125ºC(6) | | |

**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, SOSC oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{RETEN}$ (CONFIG1L<0>) = 1).

**5:** Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{RETEN}$ (CONFIG1L<0>) = 0).

**6:** 48 MHz, maximum frequency at +125ºC.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended) (Continued)

| PIC18F87K22 Family (Industrial/Extended) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.(2,3)** | | | | | | |
| | All devices | 3.3 | 5.6 | mA | -40°C | VDD = 3.3V(4) Regulator Disabled | FOSC = 16 MHz, (**PRI_RUN** mode, 4 MHz EC oscillator with PLL) |
| | | 3.3 | 5.5 | mA | +25°C | | |
| | | 3.3 | 5.5 | mA | +85°C | | |
| | | 3.6 | 6.0 | mA | +125ºC | | |
| | All devices | 3.5 | 5.9 | mA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 3.5 | 5.8 | mA | +25°C | | |
| | | 3.5 | 5.8 | mA | +85°C | | |
| | | 3.8 | 7.0 | mA | +125ºC | | |
| | All devices | 12 | 18 | mA | -40°C | VDD = 3.3V(4) Regulator Disabled | FOSC = 64 MHz, (**PRI_RUN** mode, 16 MHz EC oscillator with PLL) |
| | | 12 | 18 | mA | +25°C | | |
| | | 12 | 18 | mA | +85°C | | |
| | | 13 | 22 | mA | +125ºC(6) | | |
| | All devices | 13 | 20 | mA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 13 | 20 | mA | +25°C | | |
| | | 13 | 20 | mA | +85°C | | |
| | | 14 | 24 | mA | +125ºC(6) | | |

**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, SOSC oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 1).

**5:** Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 0).

**6:** 48 MHz, maximum frequency at +125ºC.


© 2009-2018 Microchip Technology Inc.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended) (Continued)

| PIC18F87K22 Family (Industrial/Extended) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.(2,3)** | | | | | | |
| | All devices | 42 | 73 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 1 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 42 | 73 | µA | +25°C | | |
| | | 43 | 74 | µA | +85°C | | |
| | | 53 | 100 | µA | +125ºC | | |
| | All devices | 110 | 190 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 110 | 195 | µA | +25°C | | |
| | | 110 | 195 | µA | +85°C | | |
| | | 130 | 250 | µA | +125ºC | | |
| | All devices | 280 | 450 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 290 | 440 | µA | +25°C | | |
| | | 300 | 460 | µA | +85°C | | |
| | | 330 | 500 | µA | +125ºC | | |
| | All devices | 160 | 360 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 4 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 160 | 360 | µA | +25°C | | |
| | | 170 | 370 | µA | +85°C | | |
| | | 200 | 400 | µA | +125ºC | | |
| | All devices | 330 | 650 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 340 | 660 | µA | +25°C | | |
| | | 340 | 660 | µA | +85°C | | |
| | | 370 | 700 | µA | +125ºC | | |
| | All devices | 510 | 900 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 520 | 950 | µA | +25°C | | |
| | | 540 | 990 | µA | +85°C | | |
| | | 600 | 1200 | µA | +125ºC | | |
| | All devices | 4.7 | 9 | mA | -40°C | VDD = 3.3V(4) Regulator Disabled | FOSC = 64 MHz (**PRI_IDLE** mode, EC oscillator) |
| | | 4.8 | 9 | mA | +25°C | | |
| | | 4.8 | 10 | mA | +85°C | | |
| | | 5.2 | 12 | mA | +125ºC(6) | | |
| | All devices | 5.1 | 11 | mA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 5.1 | 11 | mA | +25°C | | |
| | | 5.2 | 12 | mA | +85°C | | |
| | | 5.7 | 14 | mA | +125ºC(6) | | |

**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, SOSC oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{RETEN}$ (CONFIG1L<0>) = 1).

**5:** Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{RETEN}$ (CONFIG1L<0>) = 0).

**6:** 48 MHz, maximum frequency at +125ºC.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended) (Continued)

| PIC18F87K22 Family (Industrial/Extended) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Supply Current (IDD) Cont.(2,3)** | | | | | | |
| | All devices | 3.7 | 8.5 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 32 kHz(3) (**SEC_RUN** mode, SOSCSEL = 01) |
| | | 5.4 | 10 | µA | +25°C | | |
| | | 6.6 | 13 | µA | +85°C | | |
| | | 13 | 30 | µA | +125ºC | | |
| | All devices | 8.7 | 18 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 10 | 20 | µA | +25°C | | |
| | | 12 | 23 | µA | +85°C | | |
| | | 25 | 60 | µA | +125ºC | | |
| | All devices | 60 | 160 | µA | -40°C | VDD = 5V(4) Regulator Enabled | |
| | | 90 | 190 | µA | +25°C | | |
| | | 100 | 240 | µA | +85°C | | |
| | | 200 | 450 | µA | +125ºC | | |
| | All devices | 1.2 | 4 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | FOSC = 32 kHz(3) (**SEC_IDLE** mode, SOSCSEL = 01) |
| | | 1.7 | 5 | µA | +25°C | | |
| | | 2.6 | 6 | µA | +85°C | | |
| | | 9 | 20 | µA | +125ºC | | |
| | All devices | 1.6 | 7 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 2.8 | 9 | µA | +25°C | | |
| | | 4.1 | 10 | µA | +85°C | | |
| | | 17 | 40 | µA | +125ºC | | |
| | All devices | 60 | 150 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 80 | 180 | µA | +25°C | | |
| | | 100 | 240 | µA | +85°C | | |
| | | 180 | 440 | µA | +125ºC | | |

**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, SOSC oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{RETEN}$ (CONFIG1L<0>) = 1).

**5:** Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{RETEN}$ (CONFIG1L<0>) = 0).

**6:** 48 MHz, maximum frequency at +125ºC.


© 2009-2018 Microchip Technology Inc.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended) (Continued)

| PIC18F87K22 Family (Industrial/Extended) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| | **Module Differential Currents (ΔIWDT, ΔIBOR, ΔIHLVD, ΔIOSCB, ΔIAD)** | | | | | | |
| | **Watchdog Timer** | | | | | | |
| D022 (ΔIWDT) | All devices | 0.3 | 1 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | |
| | | 0.3 | 1 | µA | +25°C | | |
| D022 (ΔIWDT) | | 0.3 | 1 | µA | +85°C | | |
| | | 0.5 | 2 | µA | +125ºC | | |
| | All devices | 0.6 | 2 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 0.6 | 2 | µA | +25°C | | |
| | | 0.7 | 2 | µA | +85°C | | |
| | | 1 | 3 | µA | +125ºC | | |
| | All Devices | 0.6 | 2 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 0.6 | 2 | µA | +25°C | | |
| | | 0.7 | 2 | µA | +85°C | | |
| | | 1.5 | 4 | µA | +125ºC | | |
| D022A (ΔIBOR) (ΔIBOR) | **Brown-out Reset** | | | | | | |
| | All devices | 4.6 | 19 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | High-Power BOR |
| | | 4.5 | 20 | µA | +25°C | | |
| | | 4.7 | 20 | µA | +85°C | | |
| | | 18 | 40 | µA | +125ºC | | |
| | All devices | 4.2 | 20 | µA | -40°C | VDD = 5V(5) Regulator Enabled | High-Power BOR |
| | | 4.3 | 20 | µA | +25°C | | |
| | | 4.4 | 20 | µA | +85°C | | |
| | | 20 | 40 | µA | +125ºC | | |
| D022B (ΔIHLVD) | **High/Low-Voltage Detect** | | | | | | |
| | All devices | 3.8 | 9 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | |
| | | 4.2 | 9 | µA | +25°C | | |
| | | 4.3 | 10 | µA | +85°C | | |
| | | 4.5 | 12 | µA | +125ºC | | |
| | All devices | 4.5 | 11 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 4.8 | 12 | µA | +25°C | | |
| | | 4.8 | 12 | µA | +85°C | | |
| | | 5.0 | 14 | µA | +125ºC | | |
| | All devices | 4.9 | 13 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 4.9 | 13 | µA | +25°C | | |
| | | 4.9 | 13 | µA | +85°C | | |
| | | 5.3 | 15 | µA | +125ºC | | |

**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, SOSC oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{RETEN}$ (CONFIG1L<0>) = 1).

**5:** Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{RETEN}$ (CONFIG1L<0>) = 0).

**6:** 48 MHz, maximum frequency at +125ºC.

## 31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended) (Continued)

| PIC18F87K22 Family (Industrial/Extended) | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Device** | **Typ** | **Max** | **Units** | **Conditions** | | |
| D025 (ΔIRTCC) | **Real-Time Clock/Calendar with SOSC Oscillator** | | | | | | |
| | All devices | 0.7 | 2.7 | µA | -40°C | VDD = 1.8V(4) Regulator Disabled | 32.768 kHz, SOSCRUN = 1 |
| | | 0.7 | 2.8 | µA | +25°C | | |
| | | 1.1 | 2.8 | µA | +60°C | | |
| | | 1.1 | 2.9 | µA | +85°C | | |
| | | 2.2 | 4.4 | µA | +125ºC | | |
| | All devices | 1.2 | 2.9 | µA | -40°C | VDD = 3.3V(4) Regulator Disabled | |
| | | 1.1 | 2.8 | µA | +25°C | | |
| | | 2 | 4.6 | µA | +60°C | | |
| | | 2 | 4.8 | µA | +85°C | | |
| | | 4 | 6.5 | µA | +125ºC | | |
| | All devices | 1.5 | 4.4 | µA | -40°C | VDD = 5V(5) Regulator Enabled | |
| | | 1.5 | 4.4 | µA | +25°C | | |
| | | 1.7 | 4.7 | µA | +60°C | | |
| | | 1.7 | 4.7 | µA | +85°C | | |
| | | 3.5 | 6.9 | µA | +125ºC | | |

**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, SOSC oscillator, BOR, etc.).

**2:** The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

**3:** Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

**4:** Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 1).

**5:** Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 0).

**6:** 48 MHz, maximum frequency at +125ºC.


© 2009-2018 Microchip Technology Inc.

## 31.3 DC Characteristics: PIC18F87K22 Family (Industrial/Extended)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min** | **Max** | **Units** | **Conditions** |
| | VIL | **Input Low Voltage** | | | | |
| | | I/O Ports: | | | | |
| D030 | | with TTL Buffer | VSS | 0.8 | V | 4.5V < VDD < 5.5V |
| D030A | | | VSS | 0.15 VDD | V | VDD ≤ 4.5V |
| D031 | | with Schmitt Trigger Levels | — | 0.2 VDD | V | |
| | | with $I^2C$ Levels | VSS | 0.3 VDD | V | |
| | | with SMBus Levels | VSS | 0.8 | V | 2.7V ≤ VDD ≤ 5.5V |
| D032 | | $\overline{MCLR}$ | VSS | 0.2 VDD | V | |
| D033 | | OSC1 | VSS | 0.2 VDD | V | |
| D034 | | SOSCI | VSS | 0.3 VDD | V | |
| | VIH | **Input High Voltage** | | | | |
| | | I/O Ports: | | | | |
| D040 | | with TTL Buffer | 2.0 | — | V | 4.5V < VDD ≤ 5.5V |
| | | | 0.25 VDD | | V | 1.8V ≤ VDD ≤ 4.5V |
| D041 | | with Schmitt Trigger Buffer | 0.8 VDD | — | V | 2.0V ≤ VDD ≤ 5.5V |
| | | with $I^2C$ Levels | 0.7 VDD | — | V | |
| | | with SMBus Levels | 2.1 | — | V | 2.7V ≤ VDD ≤ 5.5V |
| D042 | | $\overline{MCLR}$ | 0.8 VDD | — | V | |
| D043 | | OSC1 (HS mode) | 0.7 VDD | — | V | |
| D043A | | OSC1 (EC/ECPLL mode) | 0.8 VDD | — | V | |
| D044 | | SOSCI | 0.7 VDD | | V | |
| | IIL | **Input Leakage Current**(1) | | | | |
| D060 | | I/O Ports | ±50 | ±200 | nA | VSS ≤ VPIN ≤ VDD,<br>Pin at high-impedance |
| D061 | | $\overline{MCLR}$ | — | ±5 | µA | Vss ≤ VPIN ≤ VDD, +85°C |
| D063 | | OSC1 | — | ±5 | µA | Vss ≤ VPIN ≤ VDD |
| | IPU | **Weak Pull-up Current** | | | | |
| D070 | IPURB | PORTB Weak Pull-up Current | 50 | 400 | µA | VDD = 3.3V, VPIN = VSS |

**Note 1:** Negative current is defined as current sourced by the pin.

## 31.3 DC Characteristics: PIC18F87K22 Family (Industrial/Extended) (Continued)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | |
|---|---|---|---|---|---|---|
| **Param No.** | **Symbol** | **Characteristic** | **Min** | **Max** | **Units** | **Conditions** |
| | VOL | **Output Low Voltage** | | | | |
| D080 | | I/O Ports: | | | | |
| | | PORTA, PORTB, PORTC | — | 0.6 | V | IOL = 8.5 mA, VDD = 4.5V, -40°C to +125°C |
| | | PORTD, PORTE, PORTF, PORTG, PORTH, PORTJ | — | 0.6 | V | IOL = 3.5 mA, VDD = 4.5V, -40°C to +125°C |
| D083 | | OSC2/CLKO (EC modes) | — | 0.6 | V | IOL = 1.6 mA. VDD = 5.5V, -40°C to +125°C |
| | VOH | **Output High Voltage(1)** | | | | |
| D090 | | I/O Ports: | | | V | |
| | | PORTA, PORTB, PORTC | VDD – 0.7 | — | V | IOH = -3 mA, VDD = 4.5V, -40°C to +125°C |
| | | PORTD, PORTE, PORTF, PORTG, PORTH, PORTJ | VDD – 0.7 | — | V | IOH = -2 mA, VDD = 4.5V, -40°C to +125°C |
| D092 | | OSC2/CLKO (INTOSC, EC modes) | VDD – 0.7 | — | V | IOH = -1 mA, VDD = 5.5V, -40°C to +125°C |
| | | **Capacitive Loading Specs on Output Pins** | | | | |
| D100 | COSC2 | OSC2 Pin | — | 20 | pF | In HS mode when external clock is used to drive OSC1 |
| D101 | CIO | All I/O Pins and OSC2 | — | 50 | pF | To meet the AC Timing Specifications |
| D102 | CB | SCLx, SDAx | — | 400 | pF | $I^2C$ Specification |

**Note 1:** Negative current is defined as current sourced by the pin.

## 31.4 DC Characteristics: CTMU Current Source Specifications

| DC CHARACTERISTICS | | | Standard Operating Conditions: 1.8V to 5.5V (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristic** | **Min** | **Typ(1)** | **Max** | **Units** | **Conditions** |
| | IOUT1 | CTMU Current Source, Base Range | — | 550 | — | nA | CTMUICON<1:0> = 01 |
| | IOUT2 | CTMU Current Source, 10x Range | — | 5.5 | — | µA | CTMUICON<1:0> = 10 |
| | IOUT3 | CTMU Current Source, 100x Range | — | 55 | — | µA | CTMUICON<1:0> = 11 |

**Note 1:** Nominal value at center point of current trim range (CTMUICON<7:2> = 000000).


© 2009-2018 Microchip Technology Inc.

**TABLE 31-1: MEMORY PROGRAMMING REQUIREMENTS**

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| | | **Internal Program Memory Programming Specifications(1)** | | | | | |
| D110 | VPP | Voltage on $\overline{MCLR}$/VPP/RE5 pin | VDD + 1.5 | — | 10 | V | **(Note 3, Note 4)** |
| D113 | IDDP | Supply Current during Programming | — | — | 10 | mA | |
| | | **Data EEPROM Memory** | | | | | **(Note 2)** |
| D120 | ED | Byte Endurance | 100K | — | — | E/W | -40°C to +125°C |
| D121 | VDRW | VDD for Read/Write | 1.8 | — | 5.5 | V | Using EECON to read/ write, ENVREG tied to VDD |
| | | | 1.8 | — | 3.6 | V | Using EECON to read/ write, ENVREG tied to VSS |
| D122 | TDEW | Erase/Write Cycle Time | — | 4 | — | ms | |
| D123 | TRETD | Characteristic Retention | 40 | — | — | Year | Provided no other specifications are violated |
| D124 | TREF | Number of Total Erase/Write Cycles before Refresh(2) | 1M | 10M | — | E/W | -40°C to +125°C |
| | | **Program Flash Memory** | | | | | |
| D130 | EP | Cell Endurance | 10K | — | — | E/W | -40°C to +125°C |
| D131 | VPR | VDD for Read | 1.8 | — | 5.5 | V | ENVREG tied to VDD |
| | | | 1.8 | — | 3.6 | V | ENVREG tied to VSS |
| D132B | VPEW | Voltage for Self-Timed Erase or Write Operations<br>VDD | 1.8 | — | 5.5 | V | ENVREG tied to VDD |
| D133A | TIW | Self-Timed Write Cycle Time | — | 2 | — | ms | |
| D134 | TRETD | Characteristic Retention | 40 | — | — | Year | Provided no other specifications are violated |
| D135 | IDDP | Supply Current during Programming | — | — | 10 | mA | |
| D140 | TWE | 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.

**Note 1:** These specifications are for programming the on-chip program memory through the use of table write instructions.

**2:** Refer to **Section 9.8 "Using the Data EEPROM"** for a more detailed discussion on data EEPROM endurance.

**3:** Required only if Single-Supply Programming is disabled.

**4:** The MPLAB® ICD 2 does not support variable VPP output. Circuitry to limit the MPLAB ICD 2 VPP voltage must be placed between the MPLAB ICD 2 and the target system when programming or debugging with the MPLAB ICD 2.

## 31.5 DC Characteristics: Injection Current (Industrial Only)

| DC CHARACTERISTICS | | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristic** | **Min** | **Typ†** | **Max** | **Units** | **Conditions** |
| 160A | IICL | Input Low Injection Current | 0 | — | -5(1) | mA | All pins except VDD, VSS, AVDD, AVSS, MCLR, VCAP, SOSCI, SOSCO |
| 160B | IICH | Input High Injection Current | 0 | — | +5(1) | mA | All pins except VDD, VSS, AVDD, AVSS, MCLR, VCAP, SOSCI, SOSCO |
| 160C | ΣIICT | Total Input Injection Current (sum of all I/O and control pins) | -20(1,2) | — | +20(1,2) | mA | Absolute instantaneous sum of all input injection currents from all I/O pins ( \| IICL + \| IICH \| ) ≤ ΣIICT |

† Data in "Typ" column is at 3.3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note 1:** Injection currents > | 0 | can affect the A/D results.

**2:** Any number and/or combination of I/O pins not excluded under |ICL or |ICH conditions are permitted.


© 2009-2018 Microchip Technology Inc.

**TABLE 31-2: COMPARATOR SPECIFICATIONS**

| **Operating Conditions:** 1.8V ≤ VDD ≤ 5V, -40°C ≤ TA ≤ +125°C (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| D300 | VIOFF | Input Offset Voltage | — | ±5.0 | 40 | mV | |
| D301 | VICM | Input Common Mode Voltage | — | — | AVDD – 1.5 | V | |
| D302 | CMRR | Common Mode Rejection Ratio | 55 | — | — | dB | |
| D303 | TRESP | Response Time(1) | — | 675 | 1200 | ns | |
| D304 | TMC2OV | Comparator Mode Change to Output Valid* | — | — | 10 | μs | |

**Note 1:** Response time measured with one comparator input at (AVDD – 1.5)/2, while the other input transitions from VSS to VDD.

**TABLE 31-3: VOLTAGE REFERENCE SPECIFICATIONS**

| **Operating Conditions:** 1.8V ≤ VDD ≤ 5V, -40°C ≤ TA ≤ +125°C (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| D310 | VRES | Resolution | — | 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 measured while CVRR = 1 and CVR<3:0> transitions from '0000' to '1111'.

**TABLE 31-4: INTERNAL VOLTAGE REGULATOR SPECIFICATIONS**

| **Operating Conditions:** -40°C ≤ TA ≤ +125°C (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristics** | **Min** | **Typ** | **Max** | **Units** | **Comments** |
| | VRGOUT | Regulator Output Voltage | — | 3.3 | — | V | |
| | CEFC | External Filter Capacitor Value | 4.7 | 10 | — | μF | Capacitor must be low-ESR, a low series resistance (< 5Ω) |

**TABLE 31-5: THERMAL CHARACTERISTICS**

| **Standard Operating Conditions** (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Typ.** | **Units** | **Conditions** |
| TH01 | θJA | Thermal Resistance Junction to Ambient | 47.6 | °C/W | 64-pin TQFP package |
| | | | 27.7 | °C/W | 64-pin QFN package |
| | | | 51.1 | °C/W | 80-pin TQFP package |
| TH02 | θJC | Thermal Resistance Junction to Case | 14.4 | °C/W | 64-pin TQFP package |
| | | | 13.6 | °C/W | 64-pin QFN package |
| | | | 18.6 | °C/W | 80-pin TQFP package |

## 31.6 AC (Timing) Characteristics

### 31.6.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST ($I^2C$ specifications only)
4. Ts ($I^2C$ specifications only)

| | | | |
|---|---|---|---|
| T | | | |
| F | Frequency | T | Time |

Lowercase letters (pp) and their meanings:

| | | | |
|---|---|---|---|
| pp | | | |
| cc | CCP1 | osc | OSC1 |
| ck | CLKO | rd | $\overline{RD}$ |
| cs | $\overline{CS}$ | rw | $\overline{RD}$ or $\overline{WR}$ |
| di | SDI | sc | SCK |
| do | SDO | ss | $\overline{SS}$ |
| dt | Data in | t0 | T0CKI |
| io | I/O port | t1 | T1CKI |
| mc | $\overline{MCLR}$ | wr | $\overline{WR}$ |

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 |
| $I^2C$ only | | | |
| AA | output access | High | High |
| BUF | Bus free | Low | Low |

TCC:ST ($I^2C$ specifications only)

| | | | |
|---|---|---|---|
| CC | | | |
| HD | Hold | SU | Setup |
| ST | | | |
| DAT | DATA input hold | STO | Stop condition |
| STA | Start condition | | |


© 2009-2018 Microchip Technology Inc.

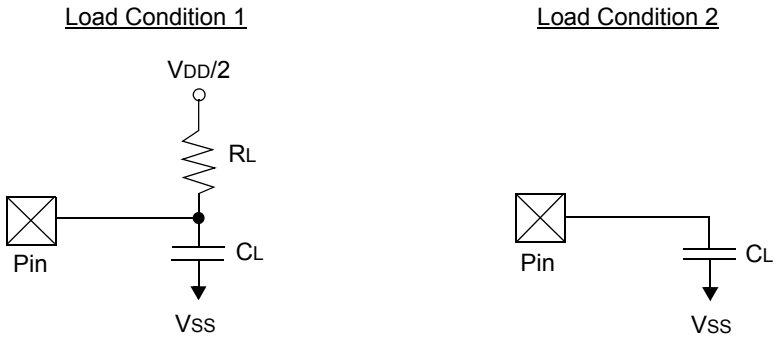
### 31.6.2 TIMING CONDITIONS

The temperature and voltages specified in Table 31-6 apply to all timing specifications unless otherwise noted. Figure 31-3 specifies the load conditions for the timing specifications.

**TABLE 31-6: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC**

| | |
|---|---|
| **AC CHARACTERISTICS** | **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended<br>Operating voltage VDD range as described in **Section 31.1** and **Section 31.3**. |

**FIGURE 31-3: 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

### 31.6.3 TIMING DIAGRAMS AND SPECIFICATIONS

**FIGURE 31-4: EXTERNAL CLOCK TIMING**

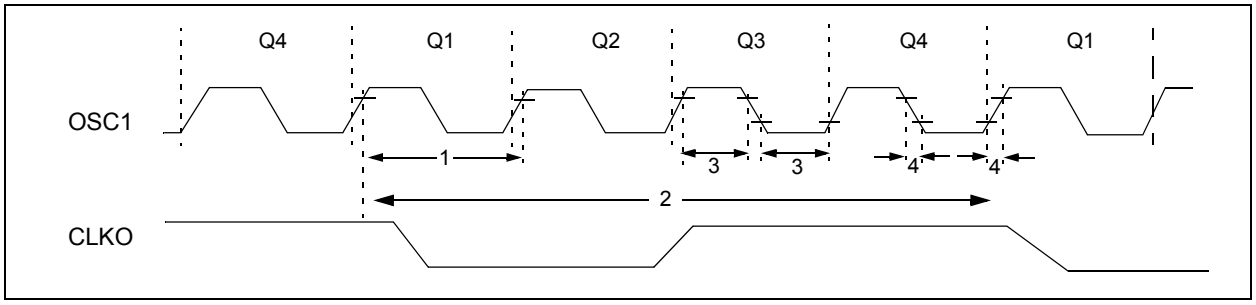



**TABLE 31-7: EXTERNAL CLOCK TIMING REQUIREMENTS**

| Param. No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 1A | FOSC | External CLKIN Frequency(1) | DC | 64 | MHz | EC, ECIO Oscillator mode -40°C ≤ TA ≤ +85°C |
| | | | DC | 48 | MHz | -40°C ≤ TA ≤ +125°C |
| | | Oscillator Frequency(1) | DC | 4 | MHz | RC Oscillator mode |
| | | | 0.1 | 4 | MHz | XT Oscillator mode |
| | | | 4 | 16 | MHz | HS Oscillator mode |
| | | | 4 | 16 | MHz | HS + PLL Oscillator mode |
| | | | 5 | 33 | kHz | LP Oscillator mode |
| 1 | TOSC | External CLKIN Period(1) | 15.6 | — | ns | EC, ECIO Oscillator mode |
| | | Oscillator Period(1) | 250 | — | ns | RC Oscillator mode |
| | | | 250 | 10,000 | ns | XT Oscillator mode |
| | | | 40 | 250 | ns | HS Oscillator mode |
| | | | 62.5 | 250 | ns | HS + PLL Oscillator mode |
| | | | 5 | 200 | µs | LP Oscillator mode |
| 2 | TCY | Instruction Cycle Time(1) | 62.5 | — | ns | TCY = 4/FOSC |
| 3 | TOSL, TOSH | External Clock in (OSC1) High or Low Time | 30 | — | ns | XT Oscillator mode |
| | | | 2.5 | — | µs | LP Oscillator mode |
| | | | 10 | — | ns | HS Oscillator mode |
| 4 | TOSR, TOSF | External Clock in (OSC1) Rise or Fall Time | — | 20 | ns | XT Oscillator mode |
| | | | — | 50 | ns | LP Oscillator mode |
| | | | — | 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/CLKIN pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.


© 2009-2018 Microchip Technology Inc.

**TABLE 31-8: PLL CLOCK TIMING SPECIFICATIONS ($V_{DD}$ = 1.8V TO 5.5V)**

| Param No. | Sym | Characteristic | Min | Typ† | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| F10 | FOSC | Oscillator Frequency Range | 4 | — | 5 | MHz | VDD = 1.8-5.5V |
| | | | 4 | — | 16 | MHz | VDD = 3.0-5.5V, -40°C to +85°C |
| | | | 4 | — | 12 | MHz | VDD = 3.0-5.5V, -40°C to +125°C |
| F11 | FSYS | On-Chip VCO System Frequency | 16 | — | 20 | MHz | VDD = 1.8-5.5V |
| | | | 16 | — | 64 | MHz | VDD = 3.0-5.5V, -40°C to +85°C |
| | | | 16 | — | 48 | MHz | VDD = 3.0-5.5V, -40°C to +125°C |
| F12 | $t_{rc}$ | PLL Start-up Time (Lock Time) | — | — | 2 | ms | |
| F13 | ΔCLK | CLKOUT Stability (Jitter) | -2 | — | +2 | % | |

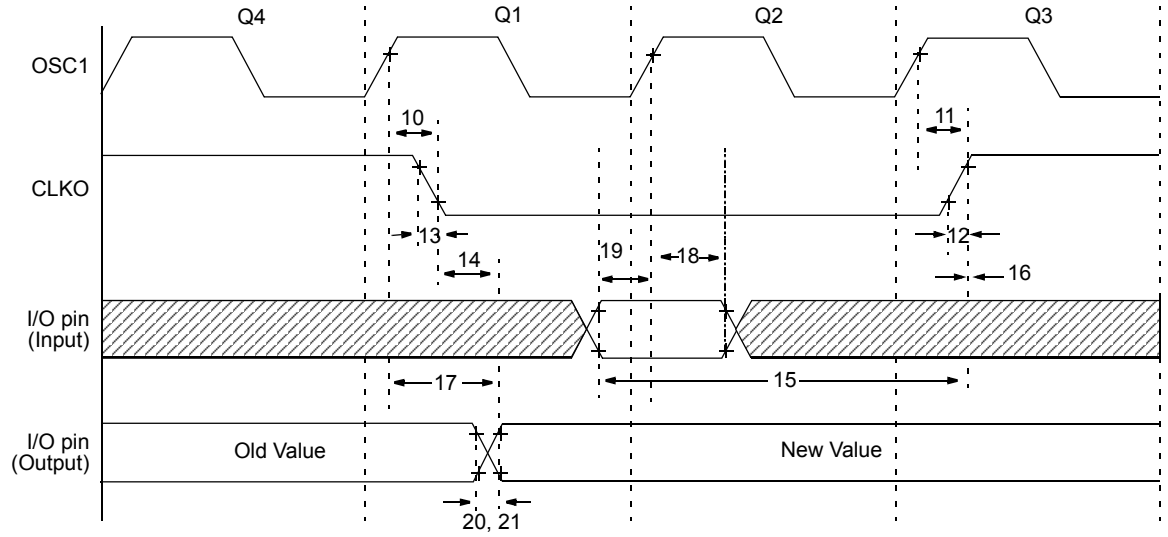
† Data in "Typ" column is at 3V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**TABLE 31-9: INTERNAL RC ACCURACY (INTOSC)**

| PIC18F87K22 Family | | Standard Operating Conditions (unless otherwise stated)<br>Operating temperature -40°C ≤ TA ≤ +125°C | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | | **Min** | **Typ** | **Max** | **Units** | **Conditions** | |
| OA1 | **HFINTOSC/MFINTOSC Accuracy @ Freq = 16 MHz, 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz(1)** | | | | | | |
| | | -2 | — | 2 | % | +25°C | VDD = 3.0-5.5V |
| | | -5 | — | 5 | % | | |
| | | -5 | — | 5 | % | -40°C to +85°C | VDD = 1.8-5.5V |
| | | -10 | — | 10 | % | -40°C to +125°C | VDD = 1.8-5.5V |
| OA2 | **LFINTOSC Accuracy @ Freq = 31 kHz** | | | | | | |
| | | -15 | — | 15 | % | -40°C to +125°C | VDD = 1.8-5.5V |

**Note 1:** Frequency calibrated at 25°C. OSCTUNE register can be used to compensate for temperature drift.

**FIGURE 31-5: CLKO AND I/O TIMING**




**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-10: CLKO AND I/O TIMING REQUIREMENTS**

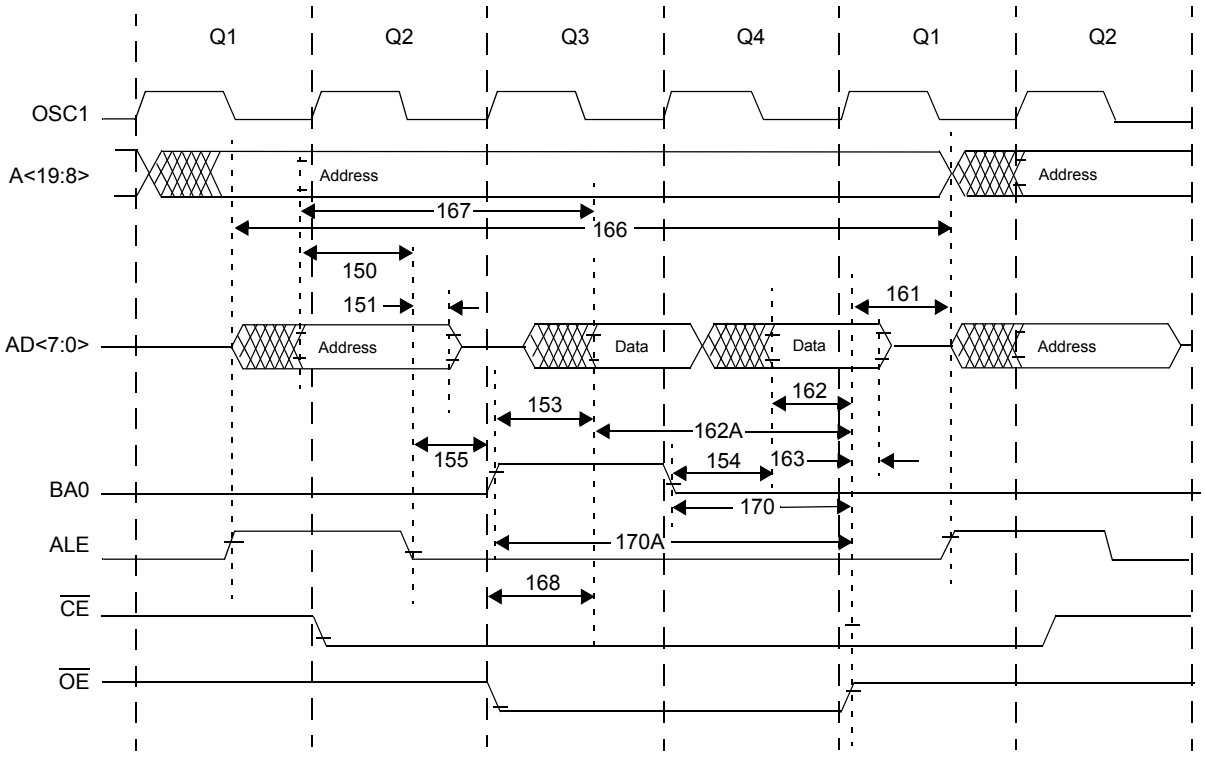
| Param No. | Symbol | Characteristic | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 10 | TOSH2CKL | OSC1 ↑ to CLKO ↓ | — | 75 | 200 | ns | **(Note 1)** |
| 11 | TOSH2CKH | OSC1 ↑ to CLKO ↑ | — | 75 | 200 | ns | **(Note 1)** |
| 12 | TCKR | CLKO Rise Time | — | 15 | 30 | 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 | — | 10 | 25 | ns | |
| 21 | TIOF | Port Output Fall Time | — | 10 | 25 | ns | |
| 22† | TINP | INTx pin High or Low Time | 20 | — | — | 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.


© 2009-2018 Microchip Technology Inc.

**FIGURE 31-6: PROGRAM MEMORY FETCH TIMING DIAGRAM (8-BIT)**

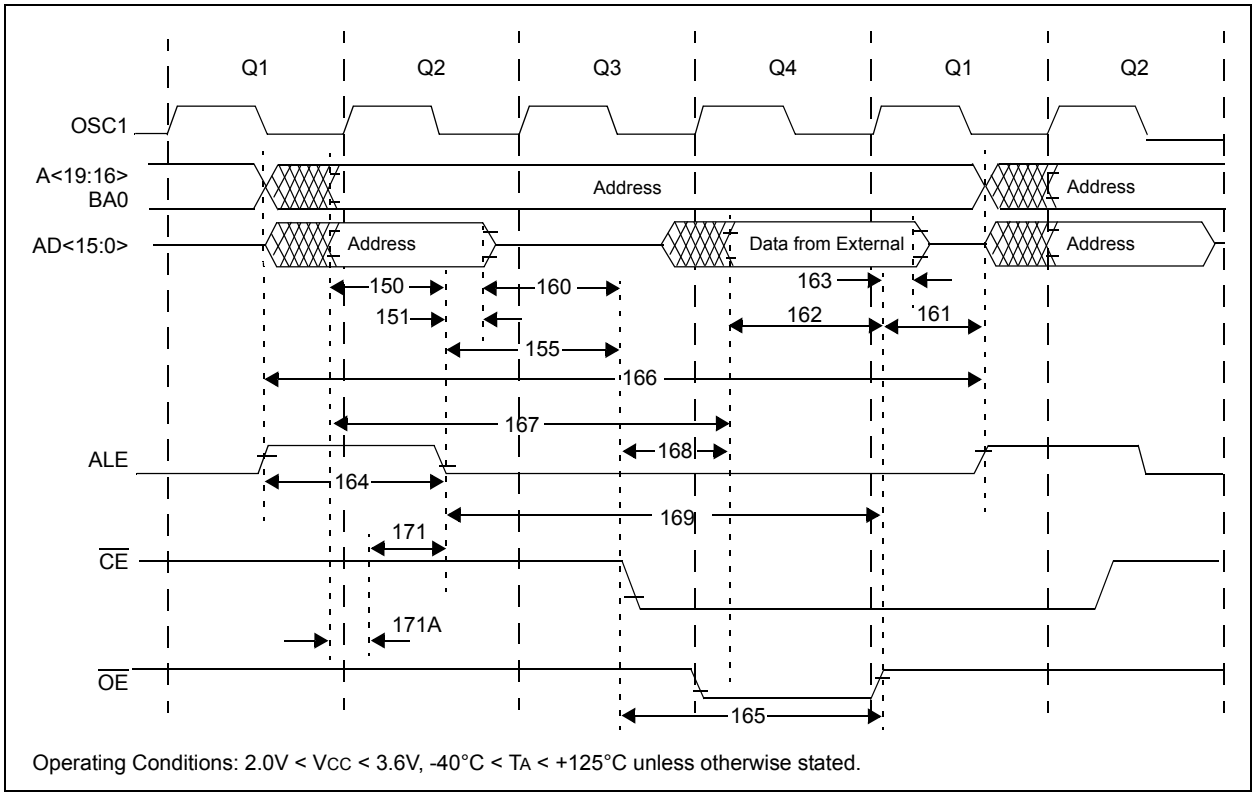



**Note:** Fmax = 25 MHz in 8-Bit External Memory mode.

**TABLE 31-11: PROGRAM MEMORY FETCH TIMING REQUIREMENTS (8-BIT)**

| Param No | Symbol | Characteristics | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| 150 | TadV2alL | Address Out Valid to ALE ↓ (address setup time) | 0.25 TCY – 10 | — | — | ns |
| 151 | TalL2adI | 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 | TalL2oeL | ALE ↓ to $\overline{OE}$ ↓ | 0.125 TCY | — | — | ns |
| 161 | ToeH2adD | $\overline{OE}$ ↑ to A/D Driven | 0.125 TCY – 5 | — | — | ns |
| 162 | TadV2oeH | Least Significant Data Valid Before $\overline{OE}$ ↑ (data setup time) | 20 | — | — | ns |
| 162A | TadV2oeH | Most Significant Data Valid Before $\overline{OE}$ ↑ (data setup time) | 0.25 TCY + 20 | — | — | ns |
| 163 | ToeH2adI | $\overline{OE}$ ↑ to Data in Invalid (data hold time) | 0 | — | — | ns |
| 166 | TalH2alH | ALE ↑ to ALE ↑ (cycle time) | — | TCY | — | ns |
| 167 | TACC | Address Valid to Data Valid | 0.5 TCY – 10 | — | — | ns |
| 168 | Toe | $\overline{OE}$ ↓ to Data Valid | — | — | 0.125 TCY + 5 | ns |
| 170 | TubH2oeH | BA0 = 0 Valid Before $\overline{OE}$ ↑ | 0.25 TCY | — | — | ns |
| 170A | TubL2oeH | BA0 = 1 Valid Before $\overline{OE}$ ↑ | 0.5 TCY | — | — | ns |

**FIGURE 31-7: PROGRAM MEMORY READ TIMING DIAGRAM**



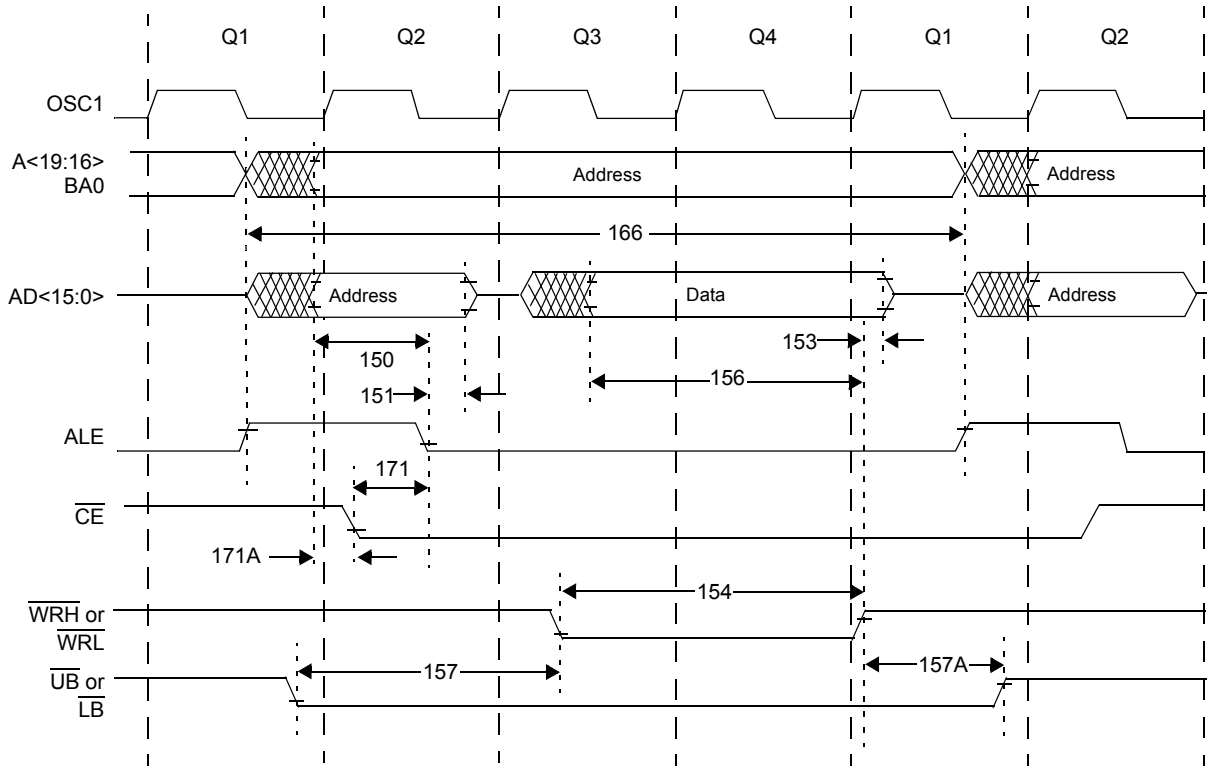

Operating Conditions: 2.0V < VCC < 3.6V, -40°C < TA < +125°C unless otherwise stated.

**TABLE 31-12: CLKO AND I/O 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 | TalL2adI | ALE ↓ to Address Out Invalid (address hold time) | 5 | — | — | ns |
| 155 | TalL2oeL | ALE ↓ to $\overline{OE}$ ↓ | 10 | 0.125 TCY | — | ns |
| 160 | TadZ2oeL | AD High-Z to $\overline{OE}$ ↓ (bus release to $\overline{OE}$) | 0 | — | — | ns |
| 161 | ToeH2adD | $\overline{OE}$ ↑ to AD Driven | 0.125 TCY – 5 | — | — | ns |
| 162 | TadV2oeH | LS Data Valid before $\overline{OE}$ ↑ (data setup time) | 20 | — | — | ns |
| 163 | ToeH2adI | $\overline{OE}$ ↑ to Data In Invalid (data hold time) | 0 | — | — | ns |
| 164 | TalH2alL | ALE Pulse Width | — | 0.25 TCY | — | ns |
| 165 | ToeL2oeH | $\overline{OE}$ 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 | $\overline{OE}$ ↓ to Data Valid | | — | 0.5 TCY – 25 | ns |
| 169 | TalL2oeH | ALE ↓ to $\overline{OE}$ ↑ | 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 |


© 2009-2018 Microchip Technology Inc.

**FIGURE 31-8: PROGRAM MEMORY WRITE TIMING DIAGRAM**

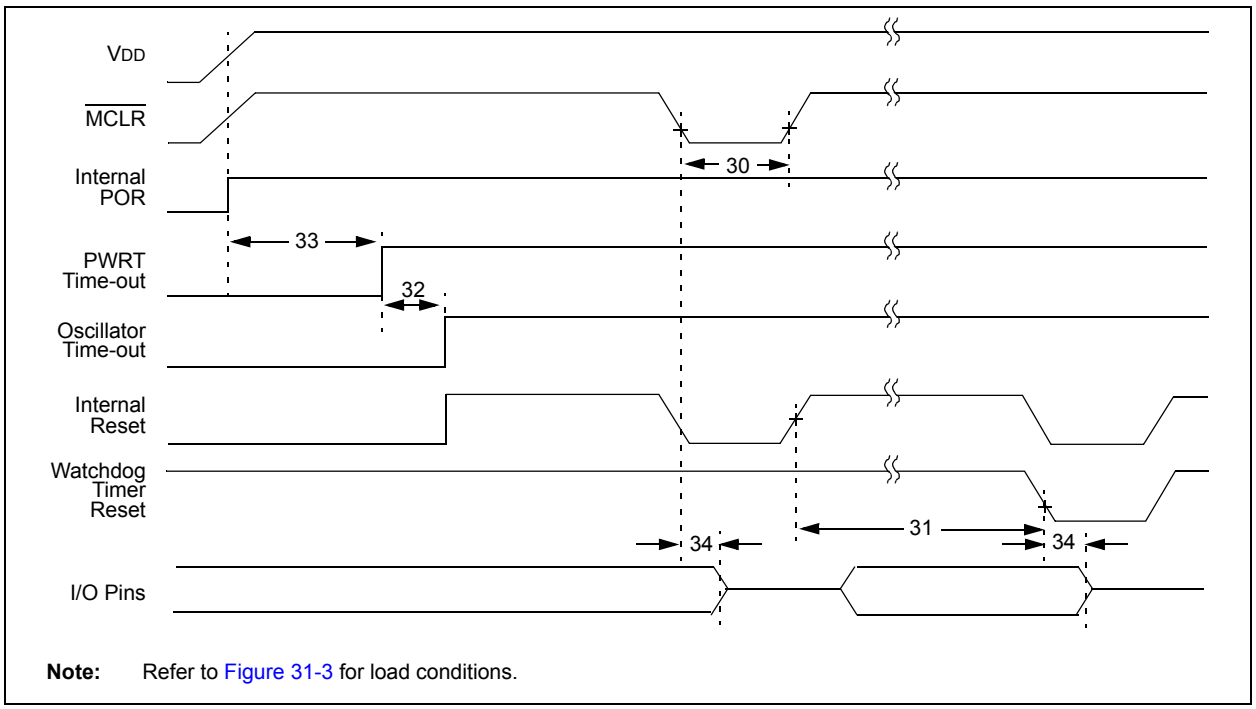



Operating Conditions: 2.0V < VCC < 3.6V, -40°C < TA < +125°C unless otherwise stated.

**TABLE 31-13: PROGRAM MEMORY WRITE TIMING REQUIREMENTS**

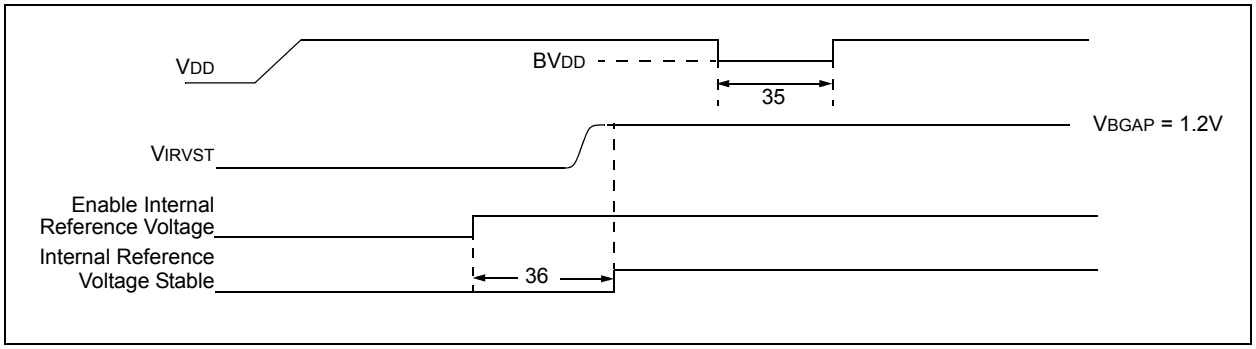
| Param. No | Symbol | Characteristics | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| 150 | TadV2alL | Address Out Valid to ALE ↓ (address setup time) | 0.25 TCY – 10 | — | — | ns |
| 151 | TalL2adl | ALE ↓ to Address Out Invalid (address hold time) | 5 | — | — | ns |
| 153 | TwrH2adl | $\overline{WRn}$ ↑ to Data Out Invalid (data hold time) | 5 | — | — | ns |
| 154 | TwrL | $\overline{WRn}$ Pulse Width | 0.5 TCY – 5 | 0.5 TCY | — | ns |
| 156 | TadV2wrH | Data Valid before $\overline{WRn}$ ↑ (data setup time) | 0.5 TCY – 10 | — | — | ns |
| 157 | TbsV2wrL | Byte Select Valid before $\overline{WRn}$ ↓ (byte select setup time) | 0.25 TCY | — | — | ns |
| 157A | TwrH2bsI | $\overline{WRn}$ ↑ 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 31-9: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING**




**Note:** Refer to Figure 31-3 for load conditions.

**FIGURE 31-10: BROWN-OUT RESET TIMING**





© 2009-2018 Microchip Technology Inc.

**TABLE 31-14: 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 | $\overline{\text{MCLR}}$ Pulse Width (low) | 2 | — | — | μs | |
| 31 | TWDT | Watchdog Timer Time-out Period (no postscaler) | — | 4.00 | — | ms | |
| 32 | TOST | Oscillation Start-up Timer Period | 1024 TOSC | — | 1024 TOSC | — | TOSC = OSC1 period |
| 33 | TPWRT | Power-up Timer Period | — | 1 | — | ms | |
| 34 | TIOZ | I/O High-Impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset | — | 2 | — | μs | |
| 35 | TBOR | Brown-out Reset Pulse Width | 200 | — | — | μs | VDD ≤ BVDD (see D005) |
| 36 | TIRVST | Time for Internal Reference Voltage to become Stable | — | 25 | — | μs | |
| 37 | THLVD | High/Low-Voltage Detect Pulse Width | 200 | — | — | μs | VDD ≤ VHLVD |
| 38 | TCSD | CPU Start-up Time | 5 | — | 10 | μs | |
| 39 | TIOBST | Time for INTOSC to Stabilize | — | 1 | — | μs | |

**FIGURE 31-11: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS**

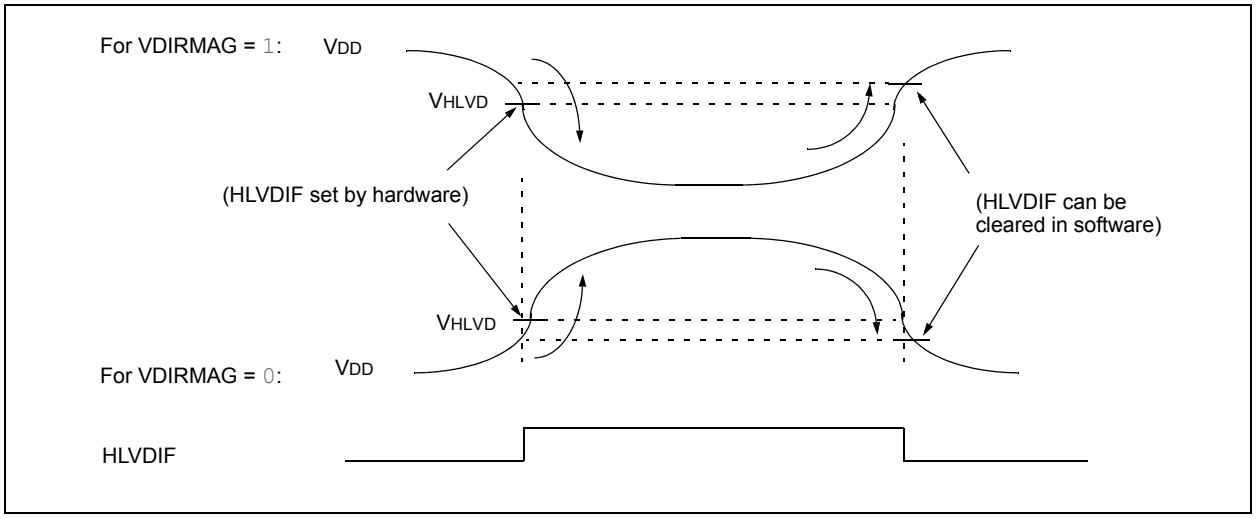
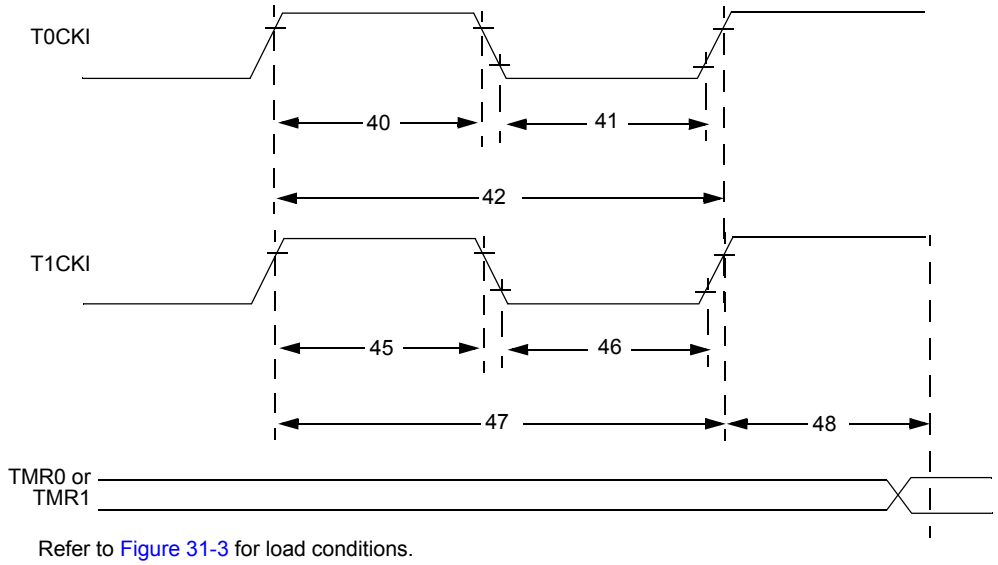



**TABLE 31-15: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS**

| **Standard Operating Conditions (unless otherwise stated)**<br>Operating temperature -40°C ≤ TA ≤ +85°C for industrial<br>-40°C ≤ TA ≤ +125°C for extended | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym** | **Characteristic** | | **Min** | **Typ** | **Max** | **Units** | **Conditions** |
| D420 | | HLVD Voltage on VDD Transition High-to-Low | HLVDL<3:0> = 0000 | 1.69 | 1.84 | 1.99 | V | |
| | | | HLVDL<3:0> = 0001 | 1.92 | 2.07 | 2.22 | V | |
| | | | HLVDL<3:0> = 0010 | 2.08 | 2.28 | 2.48 | V | |
| | | | HLVDL<3:0> = 0011 | 2.24 | 2.44 | 2.64 | V | |
| | | | HLVDL<3:0> = 0100 | 2.34 | 2.54 | 2.74 | V | |
| | | | HLVDL<3:0> = 0101 | 2.54 | 2.74 | 2.94 | V | |
| | | | HLVDL<3:0> = 0110 | 2.62 | 2.87 | 3.12 | V | |
| | | | HLVDL<3:0> = 0111 | 2.76 | 3.01 | 3.26 | V | |
| | | | HLVDL<3:0> = 1000 | 3.00 | 3.30 | 3.60 | V | |
| | | | HLVDL<3:0> = 1001 | 3.18 | 3.48 | 3.78 | V | |
| | | | HLVDL<3:0> = 1010 | 3.44 | 3.69 | 3.94 | V | |
| | | | HLVDL<3:0> = 1011 | 3.66 | 3.91 | 4.16 | V | |
| | | | HLVDL<3:0> = 1100 | 3.90 | 4.15 | 4.40 | V | |
| | | | HLVDL<3:0> = 1101 | 4.11 | 4.41 | 4.71 | V | |
| | | | HLVDL<3:0> = 1110 | 4.39 | 4.74 | 5.09 | V | |
| | | | HLVDL<3:0> = 1111 | HLVDIN pin input | | | V | |


© 2009-2018 Microchip Technology Inc.

**FIGURE 31-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS**

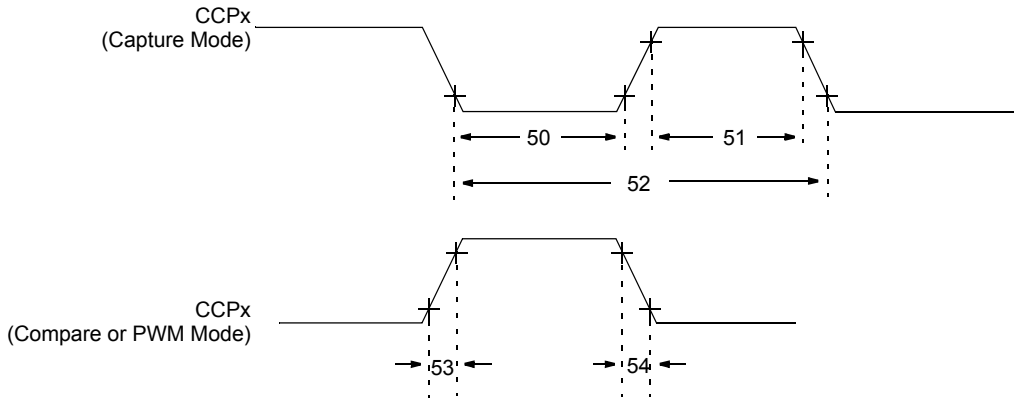



**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-16: 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 | T1CKI High Time | Synchronous, no prescaler | 0.5 TCY + 20 | — | ns | |
| | | | Synchronous, with prescaler | 10 | — | ns | |
| | | | Asynchronous | 30 | — | ns | |
| 46 | TT1L | T1CKI Low Time | Synchronous, no prescaler | 0.5 TCY + 5 | — | ns | |
| | | | Synchronous, with prescaler | 10 | — | ns | |
| | | | Asynchronous | 30 | — | ns | |
| 47 | TT1P | T1CKI Input Period | Synchronous | Greater of: 20 ns or (TCY + 40)/N | — | ns | N = prescale value (1, 2, 4, 8) |
| | | | Asynchronous | 60 | — | ns | |
| | FT1 | T1CKI Oscillator Input Frequency Range | | DC | 50 | kHz | |
| 48 | TCKE2TMRI | Delay from External T1CKI Clock Edge to Timer Increment | | 2 TOSC | 7 TOSC | — | |

**FIGURE 31-13: CAPTURE/COMPARE/PWM TIMINGS (ECCP1, ECCP2 MODULES)**



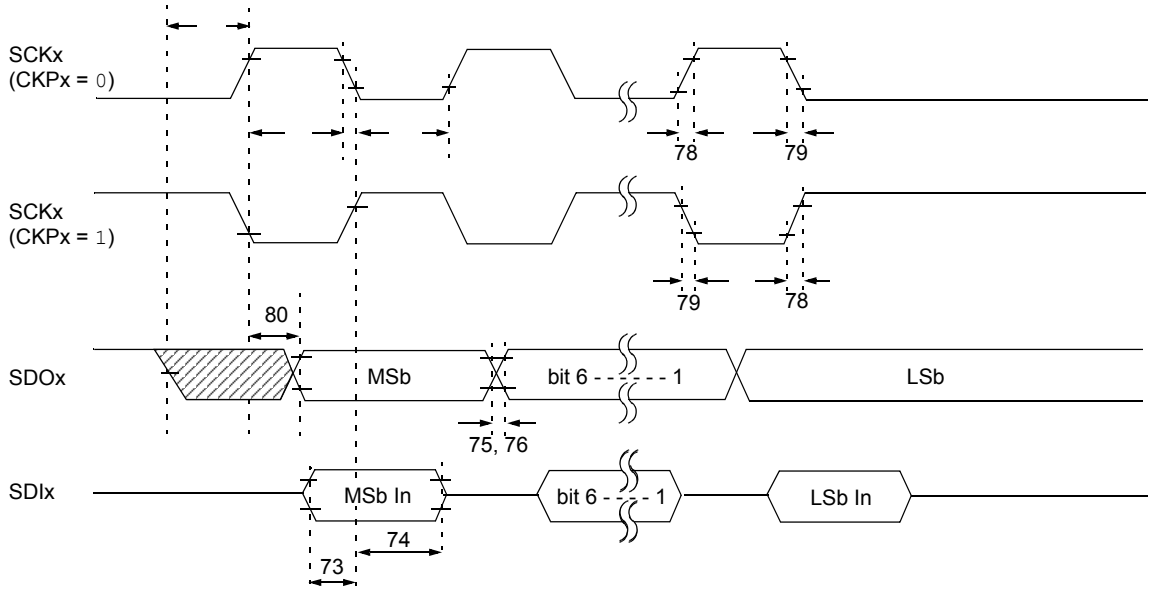

**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-17: CAPTURE/COMPARE/PWM REQUIREMENTS (ECCP1, ECCP2 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 | |


© 2009-2018 Microchip Technology Inc.

**FIGURE 31-14: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)**

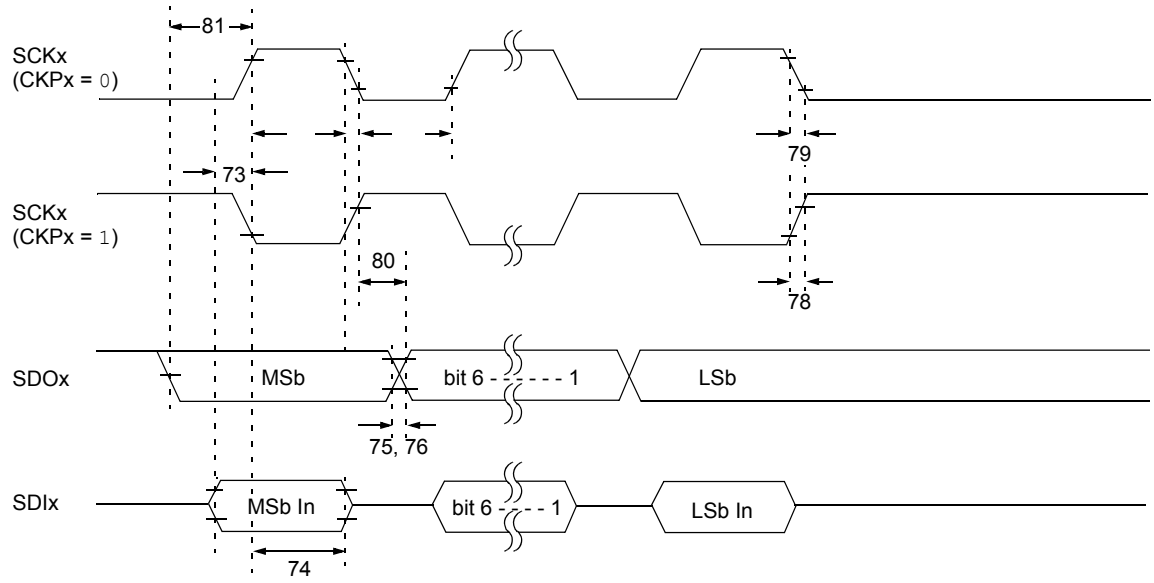



**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-18: 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 | 20 | — | ns | |
| 73A | TB2B | Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2 | 1.5 TCY + 40 | — | ns | |
| 74 | TSCH2DIL, TSCL2DIL | Hold Time of SDIx Data Input to SCKx Edge | 40 | — | 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 (Master mode) | — | 25 | ns | |
| 79 | TSCF | SCKx Output Fall Time (Master mode) | — | 25 | ns | |
| 80 | TSCH2DOV, TSCL2DOV | SDOx Data Output Valid after SCKx Edge | — | 50 | ns | |

**FIGURE 31-15: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)**



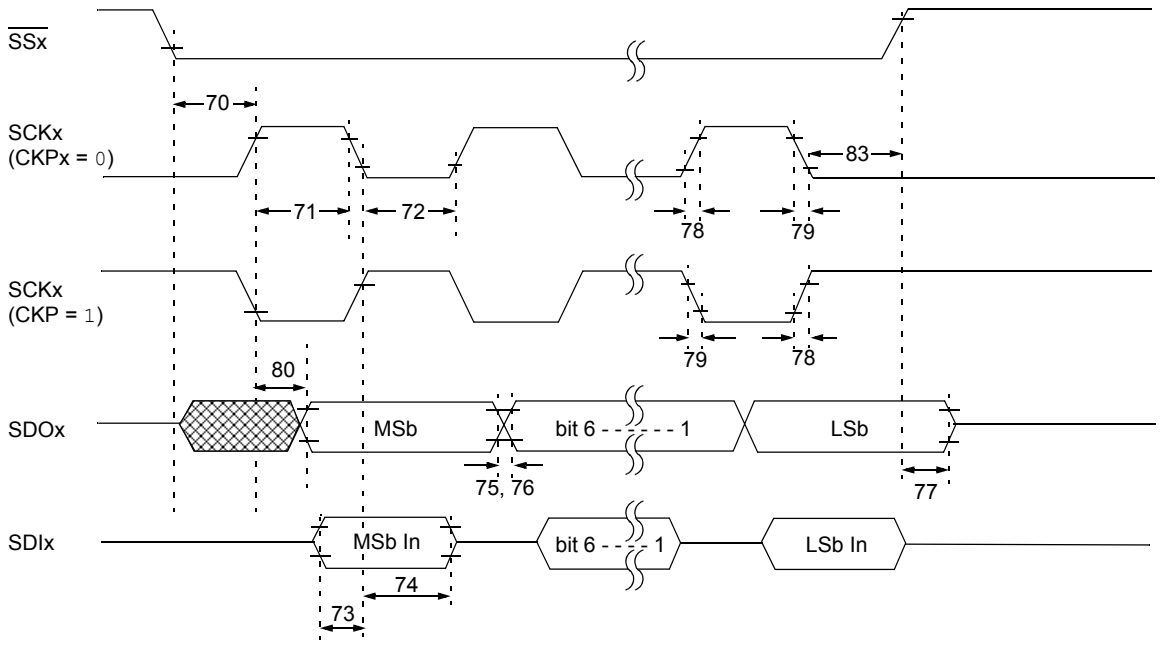

**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-19: 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 | 20 | — | ns | |
| 73A | TB2B | Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2 | 1.5 TCY + 40 | — | ns | |
| 74 | TSCH2DIL, TSCL2DIL | Hold Time of SDIx Data Input to SCKx Edge | 40 | — | 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 (Master mode) | — | 25 | ns | |
| 79 | TSCF | SCKx Output Fall Time (Master mode) | — | 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 | |


© 2009-2018 Microchip Technology Inc.

**FIGURE 31-16: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)**




**Note:** Refer to Figure 31-3 for load conditions.

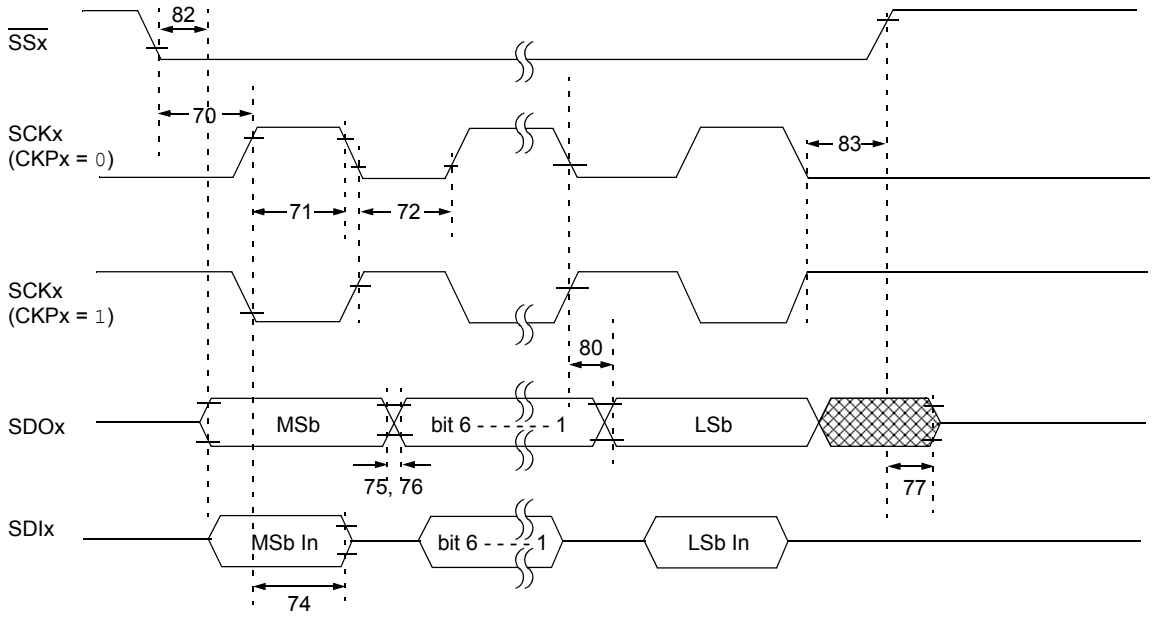
**TABLE 31-20: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)**

| 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 SSPBUF | | 3 Tcy | — | ns | |
| 71 | TscH | SCKx Input High Time (Slave mode) | Continuous | 1.25 Tcy + 30 | — | ns | |
| 71A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 72 | TscL | SCKx Input Low Time (Slave mode) | 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 | | 20 | — | 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 | | 40 | — | ns | |
| 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 | |
| 78 | TscR | SCKx Output Rise Time (Master mode) | | — | 25 | ns | |
| 79 | TscF | SCKx Output Fall Time (Master mode) | | — | 25 | 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 31-17: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)**




**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-21: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)**

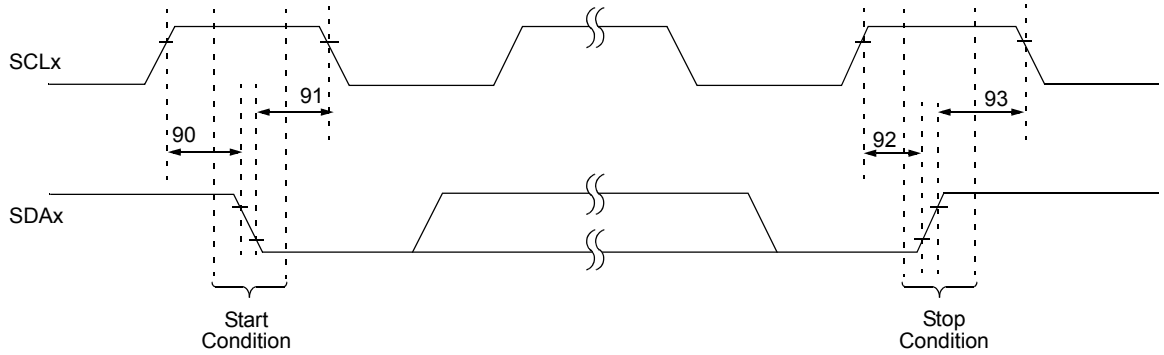
| Param No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 70 | TssL2scH, TssL2scL | $\overline{SSx}$ ↓ to SCKx ↓ or SCKx ↑ Input | | 3 TCY | — | ns | |
| 70A | TssL2WB | $\overline{SSx}$ to Write to SSPBUF | | 3 TCY | — | ns | |
| 71 | TscH | SCKx Input High Time (Slave mode) | Continuous | 1.25 TCY + 30 | — | ns | |
| 71A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 72 | TscL | SCKx Input Low Time (Slave mode) | Continuous | 1.25 TCY + 30 | — | ns | |
| 72A | | | Single Byte | 40 | — | ns | **(Note 1)** |
| 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 | | 40 | — | ns | |
| 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 | |
| 78 | TscR | SCKx Output Rise Time (Master mode) | | — | 25 | ns | |
| 79 | TscF | SCKx Output Fall Time (Master mode) | | — | 25 | 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.


© 2009-2018 Microchip Technology Inc.

**FIGURE 31-18: I²C BUS START/STOP BITS TIMING**

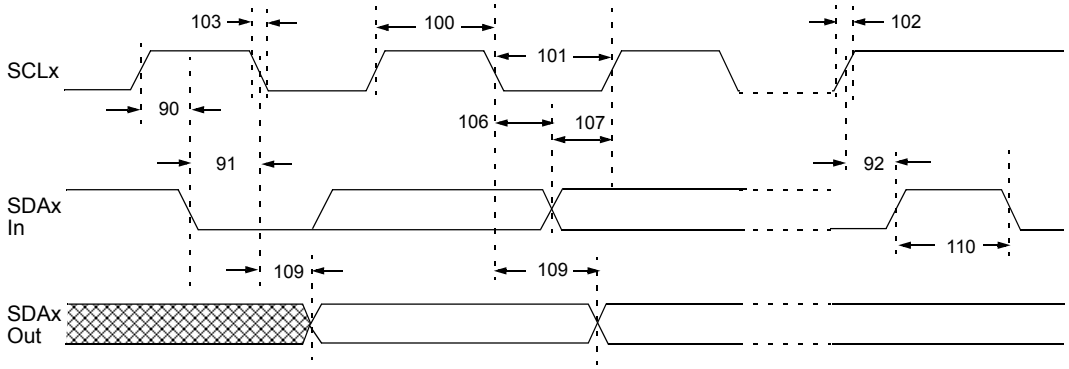



**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-22: 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 31-19: I²C BUS DATA TIMING**




**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-23: I²C BUS DATA REQUIREMENTS (SLAVE MODE)**

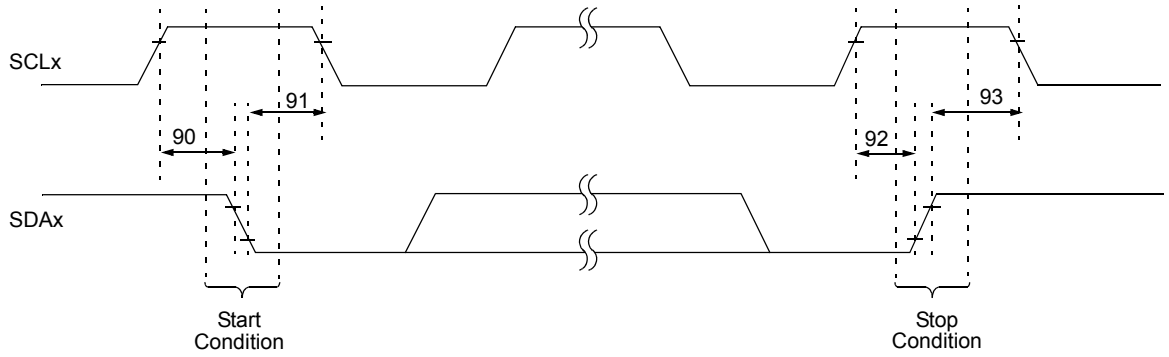
| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 4.0 | — | µs | |
| | | | 400 kHz mode | 0.6 | — | µs | |
| | | | MSSP module | 1.5 TCY | — | | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 4.7 | — | µs | |
| | | | 400 kHz mode | 1.3 | — | µs | |
| | | | MSSP module | 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.


© 2009-2018 Microchip Technology Inc.

**FIGURE 31-20: MSSP I²C BUS START/STOP BITS TIMING WAVEFORMS**



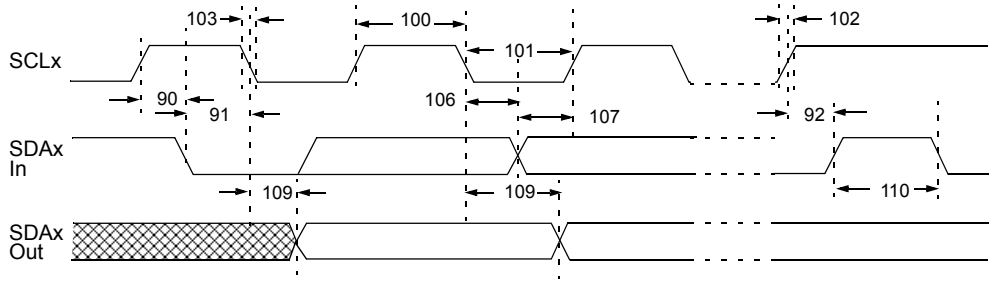

**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-24: MSSP 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 31-21: MSSP I²C BUS DATA TIMING**




**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-25: MSSP I²C BUS DATA REQUIREMENTS**

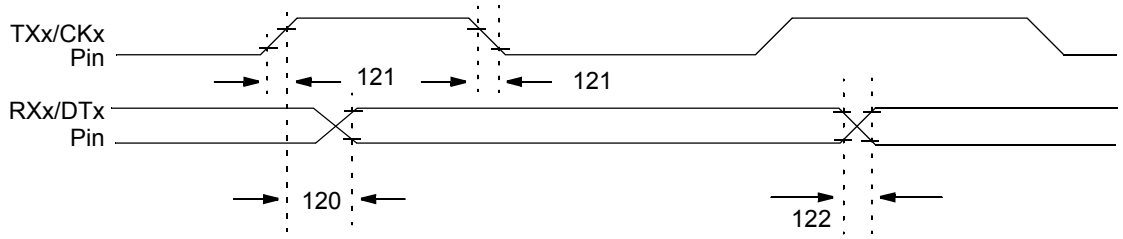
| Param. No. | Symbol | Characteristic | | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| 100 | THIGH | Clock High Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | — | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | — | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | — | |
| 101 | TLOW | Clock Low Time | 100 kHz mode | 2(TOSC)(BRG + 1) | — | — | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | — | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | — | |
| 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) | — | — | 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) | — | — | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | — | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | — | |
| 106 | THD:DAT | Data Input Hold Time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | µs | |
| | | | 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) | — | — | |
| | | | 400 kHz mode | 2(TOSC)(BRG + 1) | — | — | |
| | | | 1 MHz mode(1) | 2(TOSC)(BRG + 1) | — | — | |
| 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 | — | µs | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | µs | |
| | | | 1 MHz mode(1) | — | — | µs | |
| 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.


© 2009-2018 Microchip Technology Inc.

**FIGURE 31-22: EUSART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING**

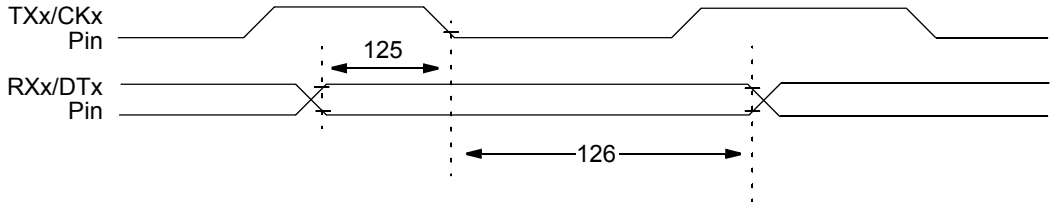



**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-26: EUSART/AUSART 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 31-23: EUSART/AUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING**




**Note:** Refer to Figure 31-3 for load conditions.

**TABLE 31-27: EUSART/AUSART 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 31-28: A/D CONVERTER CHARACTERISTICS: PIC18F87K22 FAMILY (INDUSTRIAL)**

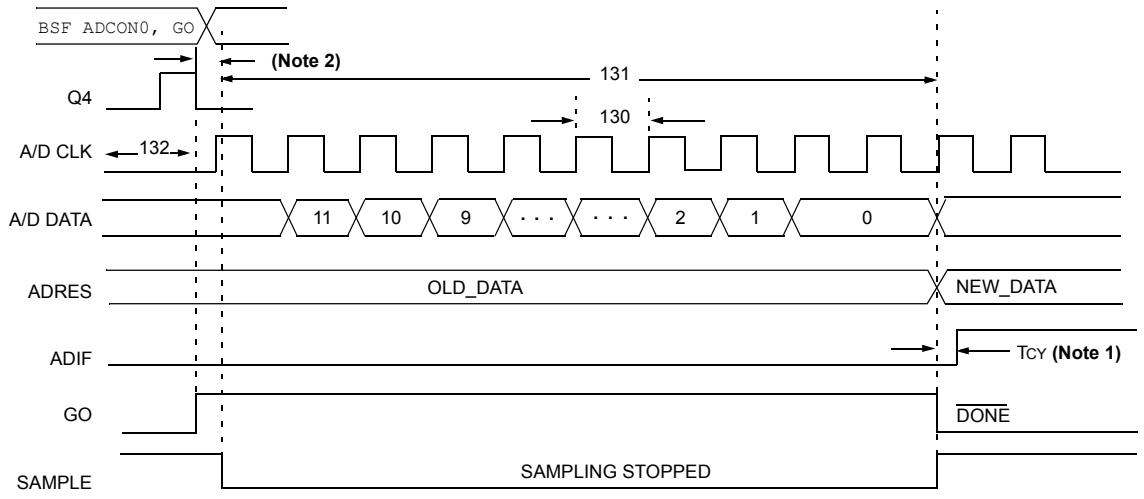
| Param No. | Sym | Characteristic | Min | Typ | Max | Units | Conditions |
|---|---|---|---|---|---|---|---|
| A01 | NR | Resolution | — | — | 12 | bit | ΔVREF ≥ 5.0V |
| A03 | EIL | Integral Linearity Error | — | ±1 | ±6.0 | LSB | ΔVREF = 5.0V |
| A04 | EDL | Differential Linearity Error | — | ±1 | +3.0/-1.0 | LSB | ΔVREF = 5.0V |
| A06 | EOFF | Offset Error | — | ±1 | ±18.0 | LSB | ΔVREF = 5.0V |
| A07 | EGN | Gain Error | — | ±1 | ±8.0 | LSB | ΔVREF = 5.0V |
| A10 | — | Monotonicity(1) | — | — | — | — | VSS ≤ VAIN ≤ VREF |
| A20 | ΔVREF | Reference Voltage Range (VREFH – VREFL) | 3 | — | VDD – VSS | V | |
| A21 | VREFH | Reference Voltage High | VSS + 3.0V | — | VDD + 0.3V | 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 | kΩ | |
| A50 | IREF | VREF Input Current(2) | —<br>— | —<br>— | 5<br>150 | μA<br>μA | During VAIN acquisition<br>During A/D conversion cycle |

**Note 1:** The A/D conversion result doesn't decrease with an increase in the input voltage.

**2:** VREFH current is from the RA3/AN3/VREF+ pin or VDD, whichever is selected as the VREFH source. VREFL current is from the RA2/AN2/VREF- pin or VSS, whichever is selected as the VREFL source.


© 2009-2018 Microchip Technology Inc.

**FIGURE 31-24: 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 31-29: A/D CONVERSION REQUIREMENTS**

| Param No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|---|---|---|---|---|---|---|
| 130 | TAD | A/D Clock Period | 0.8 | 12.5(1) | µs | TOSC-based, VREF ≥ 3.0V |
| | | | 1.4 | 25(1) | µs | VDD = 3.0V; TOSC-based, VREF full range |
| | | | — | **1** | µs | A/D RC mode |
| | | | — | 3 | µs | VDD = 3.0V; A/D RC mode |
| 131 | TCNV | Conversion Time (not including acquisition time)(2) | 14 | 15 | TAD | |
| 132 | TACQ | Acquisition Time(3) | 1.4 | — | µs | -40°C to +125°C |
| 135 | TSWC | Switching Time from Convert → Sample | — | **(Note 4)** | | |
| 137 | TDIS | Discharge Time | 0.2 | — | µs | -40°C to +125°C |

**Note 1:** The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

**2:** ADRES registers may be read on the following TCY cycle.

**3:** The time for the holding capacitor to acquire the "New" input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD). The source impedance (*RS*) on the input channels is 50Ω.

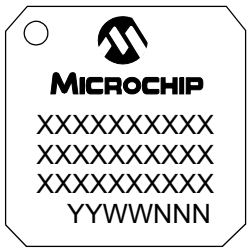
**4:** On the following cycle of the device clock.

© 2009-2018 Microchip Technology Inc.

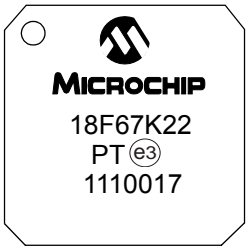
## 32.0 PACKAGING INFORMATION

### 32.1 Package Marking Information

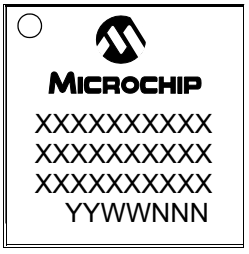
64-Lead TQFP



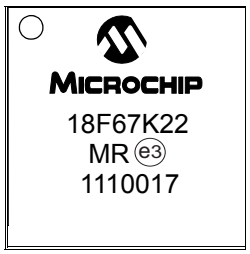

Example



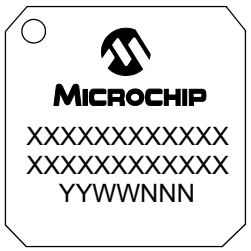

64-Lead QFN




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. |

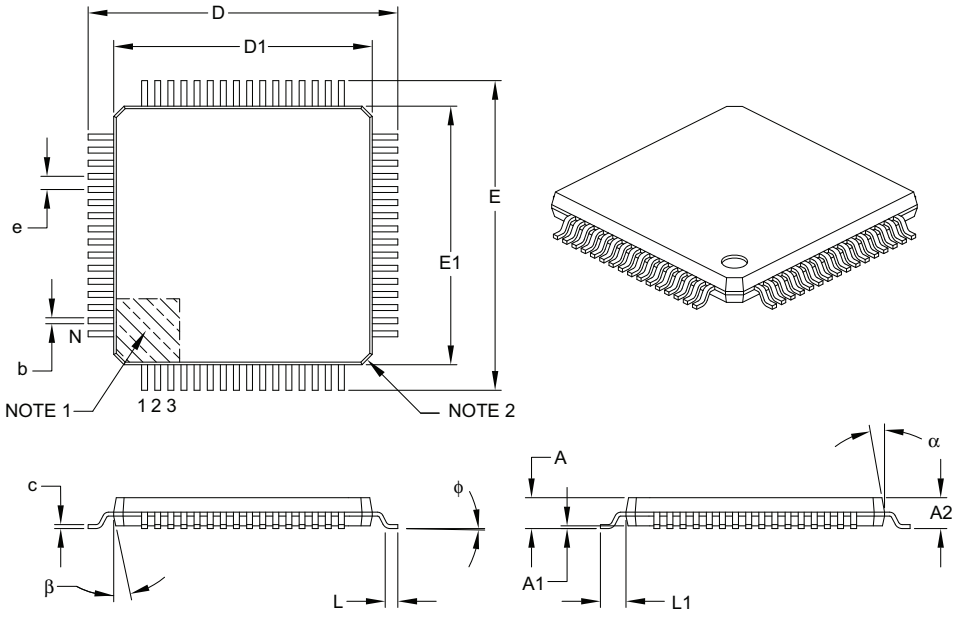
**Note**: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

## 32.2 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.

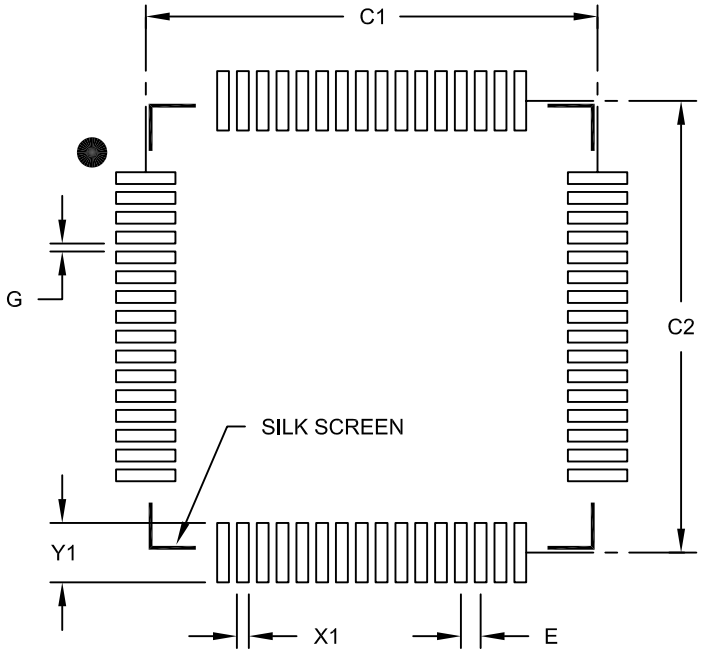
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-085B


© 2009-2018 Microchip Technology Inc.

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:

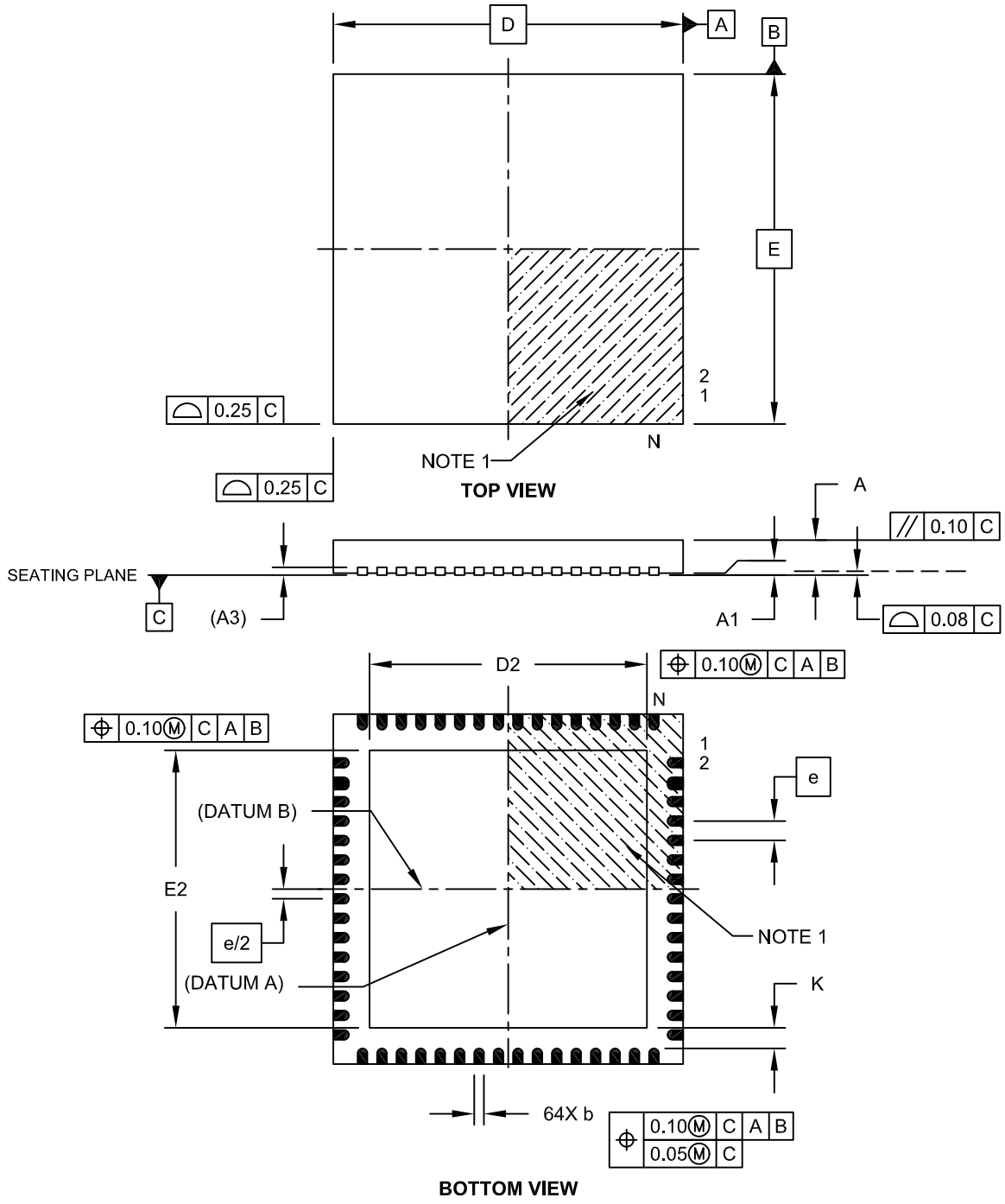
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2085B

### 64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

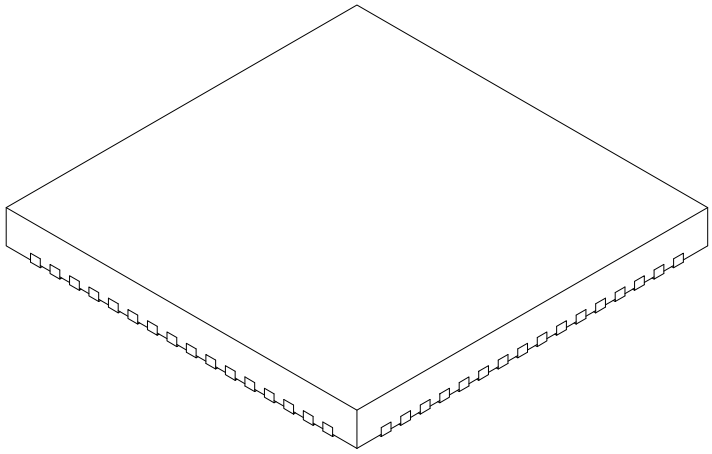



Microchip Technology Drawing C04-149B Sheet 1 of 2


© 2009-2018 Microchip Technology Inc.

## 64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 64 | |
| Pitch | e | | 0.50 BSC | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Contact Thickness | A3 | | 0.20 REF | |
| Overall Width | E | | 9.00 BSC | |
| Exposed Pad Width | E2 | 7.05 | 7.15 | 7.50 |
| Overall Length | D | | 9.00 BSC | |
| Exposed Pad Length | D2 | 7.05 | 7.15 | 7.50 |
| Contact Width | b | 0.18 | 0.25 | 0.30 |
| Contact Length | L | 0.30 | 0.40 | 0.50 |
| Contact-to-Exposed Pad | K | 0.20 | - | - |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

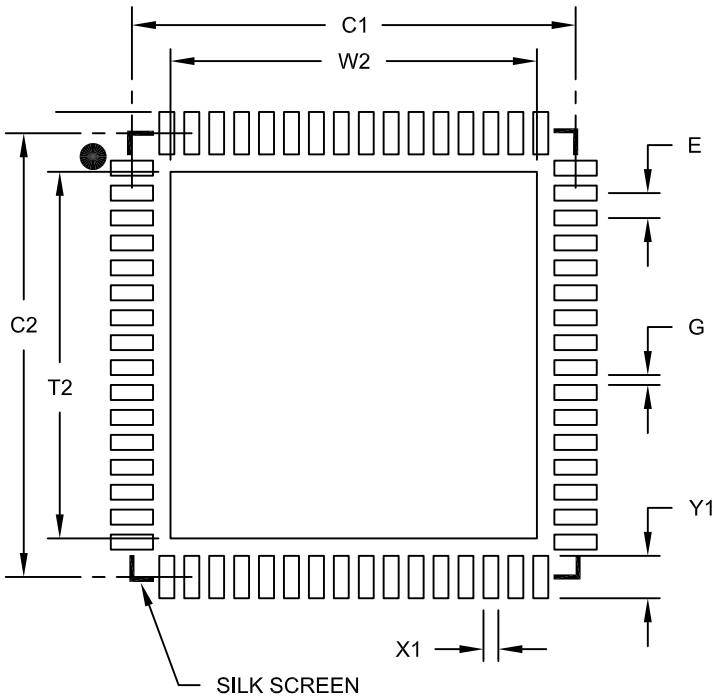
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-149B Sheet 2 of 2

64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN]
With 0.40 mm Contact Length

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.50 BSC | |
| Optional Center Pad Width | W2 | | | 7.35 |
| Optional Center Pad Length | T2 | | | 7.35 |
| Contact Pad Spacing | C1 | | 8.90 | |
| Contact Pad Spacing | C2 | | 8.90 | |
| Contact Pad Width (X64) | X1 | | | 0.30 |
| Contact Pad Length (X64) | Y1 | | | 0.85 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

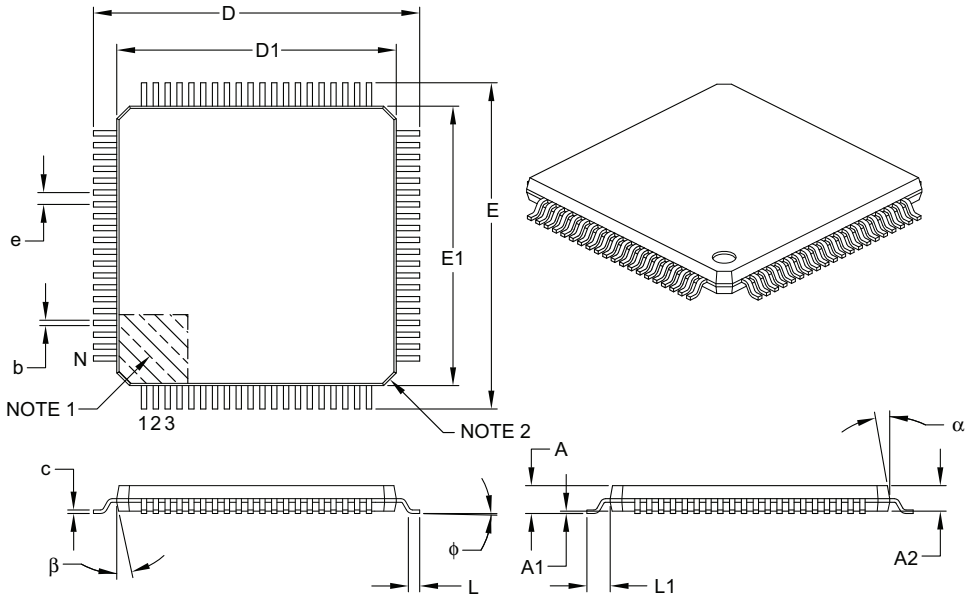
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2149A


© 2009-2018 Microchip Technology Inc.

### 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.

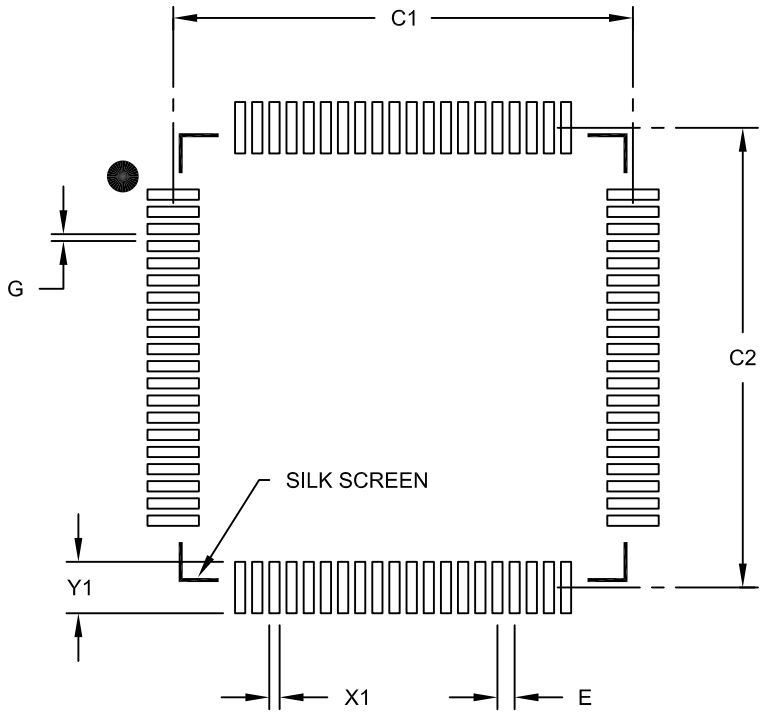
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   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


© 2009-2018 Microchip Technology Inc.

# APPENDIX A: REVISION HISTORY

## Revision F (December 2018)

Updated Sections 2.4 and 31.3. Updated Tables 31-27 and 32-2. Updated the Product Identification System section. Added Section 3.6.5: LFINTOSC Operation in Sleep. Added Section 31.5: DC Characteristics: Injection Current (Industrial Only).

## Revision E (April 2018)

Updated **Section 3.9 "Power-up Delays"**, **Section 5.0 "Reset"**, and **Section 5.6 "Power-up Timer (PWRT)"**.

Updated Power-up Timer Period ($T_{PWRT}$) from Table 31-14.

Other minor corrections.

## Revision D (June 2011)

Updated **Section 31.2 "DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended)"** and Table 31-9 with new electrical specification information. The extended temperature information has been included in this revision.

## Revision C (March 2011)

**Section 2.4 "Voltage Regulator Pins (ENVREG and $V_{CAP}/V_{DDCORE}$)"** has been replaced with a new and more detailed description and the 80-pin TQFP diagrams have been updated. Minor text edits throughout document.

## Revision B (May 2010)

Minor edits to text throughout document. Replaced all TBDs with the correct value.

## Revision A (November 2009)

Original data sheet for PIC18F87K22 family devices.

## APPENDIX B: MIGRATION FROM PIC18F87J11 AND PIC18F8722 TO PIC18F87K22

Devices in the PIC18F87K22, PIC18F87J11 and PIC18F8722 families are similar in their functions and features. Code can be migrated from the other families to the PIC18F87K22 without many changes. The differences between the device families are listed in Table B-1.

**TABLE B-1: NOTABLE DIFFERENCES BETWEEN PIC18F87K22, PIC18F87J11 AND PIC18F8722 FAMILIES**

| Characteristic | PIC18F87K22 Family | 18F87J11 Family | PIC18F8722 Family |
|---|---|---|---|
| Max Operating Frequency | 64 MHz | 48 MHz | 40 MHz |
| Max Program Memory | 128 Kbytes | 128 Kbytes | 128 Kbytes |
| Data Memory | 3,862 bytes | 3,930 bytes | 3,930 bytes |
| Program Memory Endurance | 10,000 Write/Erase (minimum) | 10,000 Write/Erase (minimum) | 10,000 Write/Erase (minimum) |
| Single Word Write for Flash | No | Yes | No |
| Oscillator Options | PLL can be used with INTOSC | PLL can be used with INTOSC | PLL can be used with INTOSC |
| CTMU | Yes | No | No |
| RTCC | Yes | No | No |
| SOSC Oscillator Options<br>TICKI Clock | Low-Power Oscillator Option for SOSC<br>T1CKI can be used as a Clock without Enabling the SOSC Oscillator | No | No |
| INTOSC | Up to 16 MHz | Up to 8 MHz | Up to 8 MHz |
| SPI/$I^2C$ | 2 | 2 | 2 |
| Timers | 11 | 5 | 5 |
| ECCP | 3 | 3 | |
| CCP | 7 | 2 | 2 |
| Data EEPROM | Yes | No | Yes |
| Programmable BOR | Multiple Level BOR | One Level BOR | Multiple Level BOR |
| WDT Prescale Options | 22 | 16 | 16 |
| 5V Operation | Yes | No (3.3V) | Yes |
| XLP | Yes | No | No |
| Regulator | Yes | Yes | No |
| Low-Power BOR | Yes | No | No |
| A/D Converter | 12-Bit Resolution, 24 Input Channels, Differential | 10-Bit Resolution, 15 Input Channels, Non-Differential | 10-Bit Resolution, 16 Input Channels, Non-Differential |
| Internal Temperature Sensor | Yes | No | No |
| Programmable HLVD | Yes | No | Yes |
| EUSART | 2 EUSARTs | 2 EUSARTs | 2 EUSARTs |
| Comparators | 3 | 2 | 2 |
| Oscillator options | 14 Options by FOSC<3:0> | 8 Options by FOSC<3:0> | 12 Options by FOSC<3:0> |
| Ultra-Low-Power Wake-up (ULPW) | Yes | No | No |
| Power-up Timer | Yes | Yes | Yes |
| $\overline{MCLR}$ Pin as Input Port | Yes | No | Yes |


© 2009-2018 Microchip Technology Inc.

## THE MICROCHIP WEBSITE

Microchip provides online support via our WWW site at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

## CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

**Technical support is available through the website at: http://microchip.com/support**

**NOTES:**


© 2009-2018 Microchip Technology Inc.

## PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| PART NO. | [X](1) | X | /XX | XXX |
|---|---|---|---|---|
| Device | Tape and Reel Option | Temperature Range | Package | Pattern |

| | |
|---|---|
| **Device:** | PIC18F65K22, PIC18F65K22T<br>PIC18F66K22, PIC18F66K22T<br>PIC18F67K22, PIC18F67K22T<br>PIC18F85K22, PIC18F85K22T<br>PIC18F86K22, PIC18F86K22T<br>PIC18F87K22, PIC18F87K22T |
| **Tape and Reel Option:** | Blank = Standard packaging (tube or tray)<br>T = Tape and Reel(1) |
| **Temperature Range:** | I = -40°C to +85°C (Industrial)<br>E = -40°C to +125°C (Extended) |
| **Package:** | PT = TQFP (Thin Quad Flatpack)<br>MR = QFN (Plastic Quad Flat) |
| **Pattern:** | QTP, SQTP, Code or Special Requirements (blank otherwise) |

**Examples:**

a) PIC18F87K22-I/PT 301 = Industrial temperature, TQFP package, QTP pattern #301.

b) PIC18F87K22T-I/PT = Tape and reel, Industrial temperature, TQFP package

c) PIC18F87K22T-E/PT = Tape and reel, Extended temperature, TQFP package

**Note 1:** Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is nto printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.

**2:** F = Standard Voltage Range

**3:** T = In tape and reel PLCC and TQFP packages only

**4:** RSL = Silicon Revision A3.

© 2009-2018 Microchip Technology Inc.

**Note the following details of the code protection feature on Microchip devices:**

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

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

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

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

**QUALITY MANAGEMENT SYSTEM**
**CERTIFIED BY DNV**
**═ ISO/TS 16949 ═**

**Trademarks**

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

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

© 2018, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-3984-4

# 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

**Austin, TX**
Tel: 512-257-3370

**Boston**
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

**Chicago**
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

**Dallas**
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

**Detroit**
Novi, MI
Tel: 248-848-4000

**Houston, TX**
Tel: 281-894-5983

**Indianapolis**
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

**Los Angeles**
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

**Raleigh, NC**
Tel: 919-844-7510

**New York, NY**
Tel: 631-435-6000

**San Jose, CA**
Tel: 408-735-9110
Tel: 408-436-4270

**Canada - Toronto**
Tel: 905-695-1980
Fax: 905-695-2078

## ASIA/PACIFIC

**Australia - Sydney**
Tel: 61-2-9868-6733

**China - Beijing**
Tel: 86-10-8569-7000

**China - Chengdu**
Tel: 86-28-8665-5511

**China - Chongqing**
Tel: 86-23-8980-9588

**China - Dongguan**
Tel: 86-769-8702-9880

**China - Guangzhou**
Tel: 86-20-8755-8029

**China - Hangzhou**
Tel: 86-571-8792-8115

**China - Hong Kong SAR**
Tel: 852-2943-5100

**China - Nanjing**
Tel: 86-25-8473-2460

**China - Qingdao**
Tel: 86-532-8502-7355

**China - Shanghai**
Tel: 86-21-3326-8000

**China - Shenyang**
Tel: 86-24-2334-2829

**China - Shenzhen**
Tel: 86-755-8864-2200

**China - Suzhou**
Tel: 86-186-6233-1526

**China - Wuhan**
Tel: 86-27-5980-5300

**China - Xian**
Tel: 86-29-8833-7252

**China - Xiamen**
Tel: 86-592-2388138

**China - Zhuhai**
Tel: 86-756-3210040

## ASIA/PACIFIC

**India - Bangalore**
Tel: 91-80-3090-4444

**India - New Delhi**
Tel: 91-11-4160-8631

**India - Pune**
Tel: 91-20-4121-0141

**Japan - Osaka**
Tel: 81-6-6152-7160

**Japan - Tokyo**
Tel: 81-3-6880- 3770

**Korea - Daegu**
Tel: 82-53-744-4301

**Korea - Seoul**
Tel: 82-2-554-7200

**Malaysia - Kuala Lumpur**
Tel: 60-3-7651-7906

**Malaysia - Penang**
Tel: 60-4-227-8870

**Philippines - Manila**
Tel: 63-2-634-9065

**Singapore**
Tel: 65-6334-8870

**Taiwan - Hsin Chu**
Tel: 886-3-577-8366

**Taiwan - Kaohsiung**
Tel: 886-7-213-7830

**Taiwan - Taipei**
Tel: 886-2-2508-8600

**Thailand - Bangkok**
Tel: 66-2-694-1351

**Vietnam - Ho Chi Minh**
Tel: 84-28-5448-2100

## EUROPE

**Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**Denmark - Copenhagen**
Tel: 45-4450-2828
Fax: 45-4485-2829

**Finland - Espoo**
Tel: 358-9-4520-820

**France - Paris**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

**Germany - Garching**
Tel: 49-8931-9700

**Germany - Haan**
Tel: 49-2129-3766400

**Germany - Heilbronn**
Tel: 49-7131-67-3636

**Germany - Karlsruhe**
Tel: 49-721-625370

**Germany - Munich**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

**Germany - Rosenheim**
Tel: 49-8031-354-560

**Israel - Ra'anana**
Tel: 972-9-744-7705

**Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**Italy - Padova**
Tel: 39-049-7625286

**Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

**Norway - Trondheim**
Tel: 47-7288-4388

**Poland - Warsaw**
Tel: 48-22-3325737

**Romania - Bucharest**
Tel: 40-21-407-87-50

**Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

**Sweden - Gothenberg**
Tel: 46-31-704-60-40

**Sweden - Stockholm**
Tel: 46-8-5090-4654

**UK - Wokingham**
Tel: 44-118-921-5800
Fax: 44-118-921-5820



© 2009-2018 Microchip Technology Inc.

## Looking for pricing, stock, or lifecycle information?

Click below to explore more details on WIN SOURCE:

- View PIC18F66K22-I/PTRSL on WIN SOURCE
- Microchip Technology Information

## Optimize Your Supply Chain with WIN SOURCE Solutions

- Global Sourcing Solution
- Obsolete Management
- Cost Control Management
- Shortage Management
- Alternative Solution
- Excess Inventory Management