PIC18F47J13 FAMILY

PIC18F47J13 Family 28/44-Pin, High-Performance Microcontrollers with XLP Technology

Power Management Features with XLP for eXtreme Low Power:

- Deep Sleep mode: CPU Off, Peripherals Off, SRAM Off, Currents Down to 9 nA and 700 nA with RTCC:
  - Able to wake-up on external triggers, programmable WDT or RTCC alarm
  - Ultra Low-Power Wake-up (ULPWU)
- Sleep mode: CPU Off, Peripherals Off, SRAM On, Fast Wake-up, Currents Down to 0.2 μA, 2V Typical
- Idle: CPU Off, SRAM On, Currents Down to 1.7 μA Typical
- Run: CPU On, SRAM On, Currents Down to 5.8 μA Typical
- Timer1 Oscillator w/RTCC: 0.7 μA, 32 kHz Typical
- Watchdog Timer: 0.33 μA, 2V Typical

Flexible Oscillator Structure:

- Two External Clock modes, Up to 48 MHz (12 MIPS)
- Integrated Crystal/Resonator Driver
- Low-Power 31 kHz Internal RC Oscillator
- Tunable Internal Oscillator (31 kHz to 8 MHz, ±0.15% Typical, ±1% Max.)
- Precision 48 MHz PLL or 4x PLL Options
- Low-Power Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor (FSCM):
  - Allows for safe shutdown if any clock stops
- Programmable Reference Clock Output Generator

Peripheral Highlights:

- Peripheral Pin Select:
  - Allows independent I/O mapping of many peripherals
  - Continuous hardware integrity checking and safety interlocks prevent unintentional configuration changes
- Hardware Real-Time Clock/Calendar (RTCC):
  - Provides clock, calendar and alarm functions
- High-Current Sink/Source 25 mA/25mA (PORTB and PORTC)
- Four Programmable External Interrupts
- Four Input Change Interrupts
- Three Enhanced Capture/Compare/PWM (ECCP) modules:
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
  - Pulse steering control

Peripheral Highlights (cont.):

- Seven Capture/Compare/PWM (CCP) modules
- Two Master Synchronous Serial Port (MSSP) modules featuring:
  - 3-wire SPI (all 4 modes)
  - SPI Direct Memory Access (DMA) channel w/1024 byte count
  - I^2C Master and Slave modes
- 8-Bit Parallel Master Port/Enhanced Parallel Slave Port
- Three Analog Comparators with Input Multiplexing
- 12-Bit Analog-to-Digital (A/D) Converter module:
  - Up to 13 input channels
  - Auto-acquisition capability
  - 10-bit mode for 100 ksps conversion speed
  - Conversion available during Sleep
- High/Low-Voltage Detect module
- Charge Time Measurement Unit (CTMU):
  - Provides a precise resolution time measurement for both flow measurement and simple temperature sensing
  - Supports capacitive touch sensing for touch screens and capacitive switches
- Two Enhanced USART modules:
  - Supports RS-485, RS-232 and LIN/J2602
  - Auto-wake-up on Start bit
  - Auto-Baud Detect (ABD)

Special Microcontroller Features:

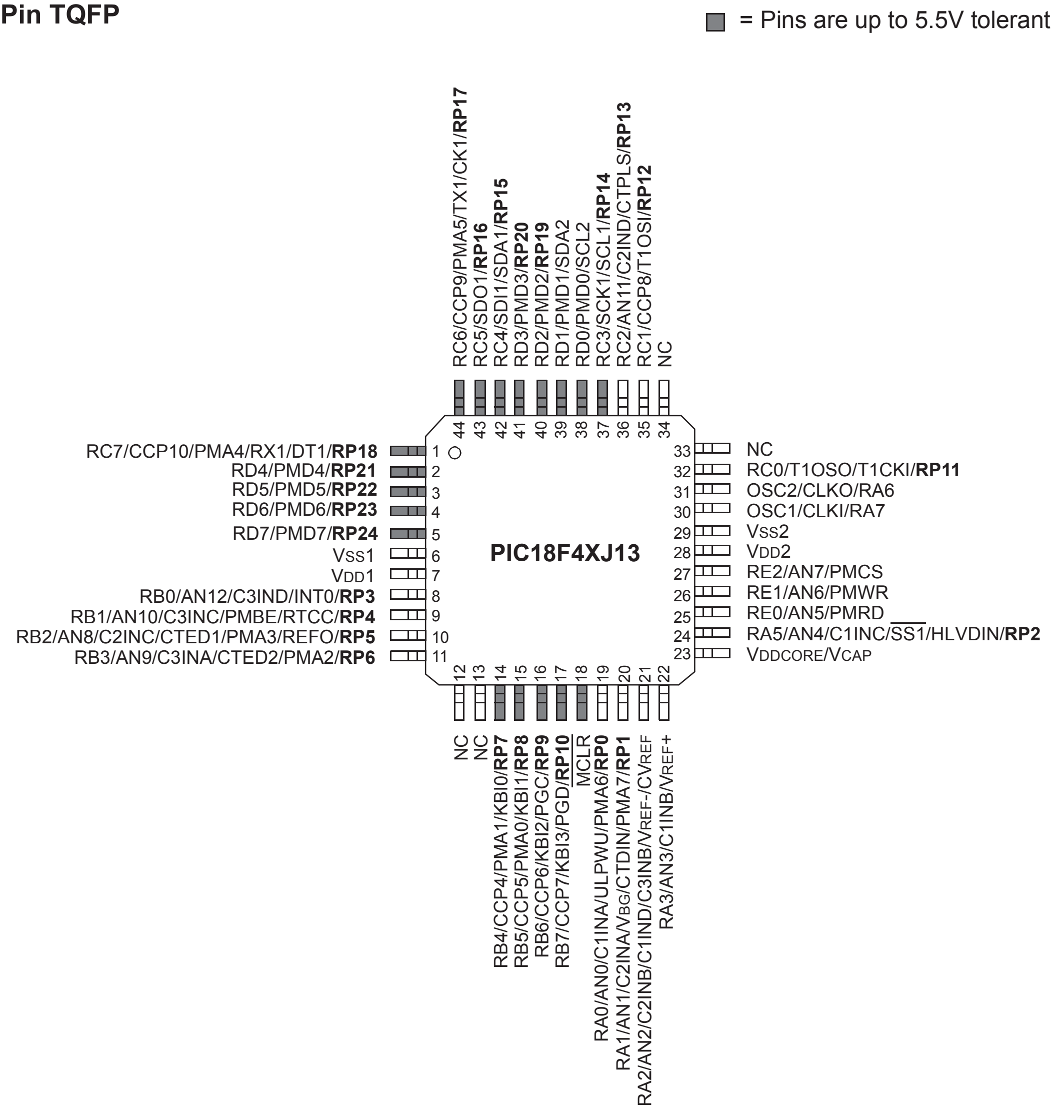
- 5.5V Tolerant Inputs (digital only pins)
- Low-Power, High-Speed CMOS Flash Technology
- C Compiler Optimized Architecture for Re-Entrant Code
- Priority Levels for Interrupts
- Self-Programmable under Software Control
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-Supply In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) with 3 Breakpoints via Two Pins
- Operating Voltage Range of 2.0V to 3.6V
- On-Chip 2.5V Regulator
- Flash Program Memory of 10,000 Erase/Write Cycles Minimum and 20-Year Data Retention

TABLE 1: PIC18F47J13 FAMILY TYPES

PIC18F Device	Pins	Program Memory (bytes)	SRAM (bytes)	Remappable Pins	Timers 8/16-Bit	ECCP/CCP	EUSART	MSSP			12-Bit A/D (ch)	Comparators	Deep Sleep	PMP/PSP	CTMU	RTCC
								SPI w/DMA		I^2C						
PIC18F26J13	28	64K	3760	19	4/4	3/7	2	2	Y	Y	10	3	Y	N	Y	Y
PIC18F27J13	28	128K	3760	19	4/4	3/7	2	2	Y	Y	10	3	Y	N	Y	Y
PIC18F46J13	44	64K	3760	25	4/4	3/7	2	2	Y	Y	13	3	Y	Y	Y	Y
PIC18F47J13	44	128K	3760	25	4/4	3/7	2	2	Y	Y	13	3	Y	Y	Y	Y
PIC18LF26J13	28	64K	3760	19	4/4	3/7	2	2	Y	Y	10	3	N	N	Y	Y
PIC18LF27J13	28	128K	3760	19	4/4	3/7	2	2	Y	Y	10	3	N	N	Y	Y
PIC18LF46J13	44	64K	3760	25	4/4	3/7	2	2	Y	Y	13	3	N	Y	Y	Y
PIC18LF47J13	44	128K	3760	25	4/4	3/7	2	2	Y	Y	13	3	N	Y	Y	Y

Pin Diagrams

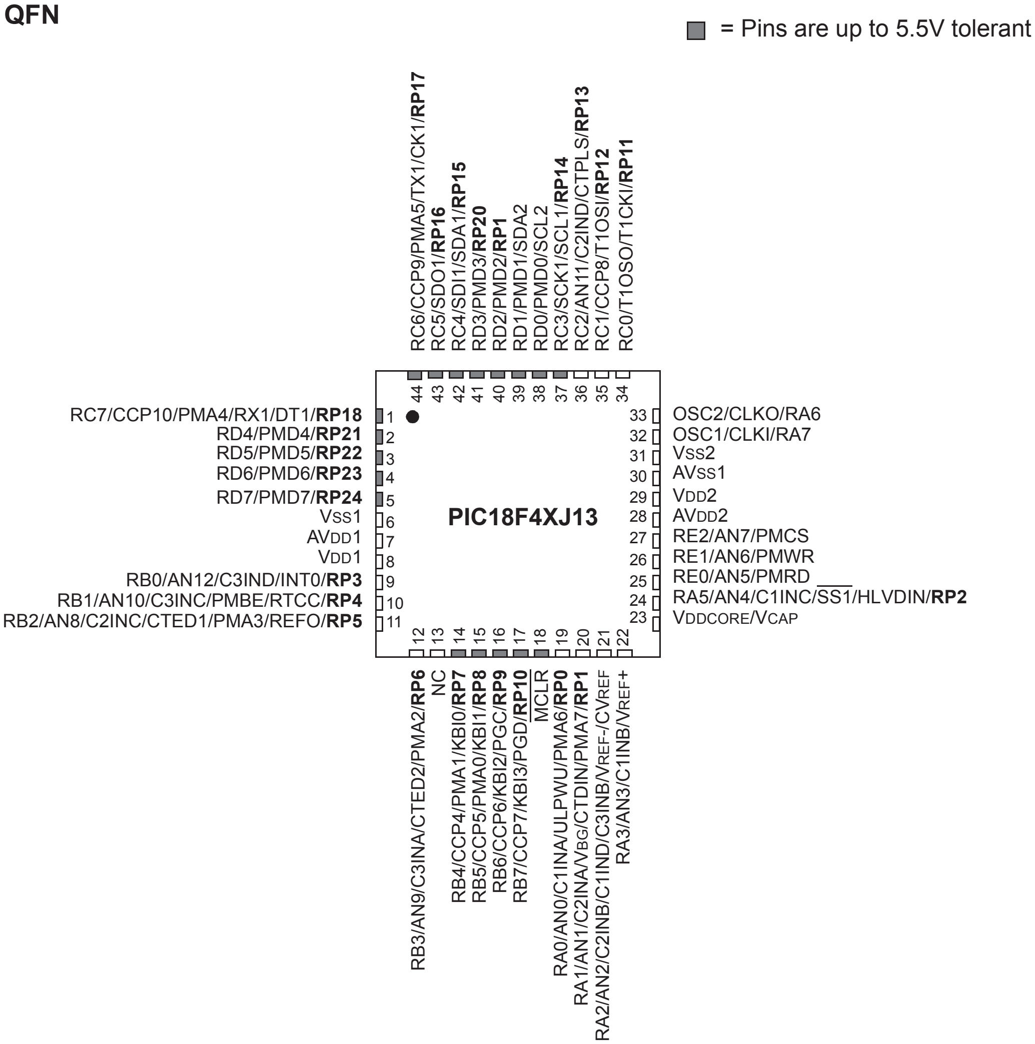
44-Pin TQFP


■ = Pins are up to 5.5V tolerant
PIC18F4XJ13
1 RC7/CCP10/PMA4/RX1/DT1/RP18
2 RD4/PMD4/RP21
3 RD5/PMD5/RP22
4 RD6/PMD6/RP23
5 RD7/PMD7/RP24
6 VSS1
7 VDD1
8 RB0/AN12/C3IND/INT0/RP3
9 RB1/AN10/C3INC/PMBE/RTCC/RP4
10 RB2/AN8/C2INC/CTED1/PMA3/REFO/RP5
11 RB3/AN9/C3INA/CTED2/PMA2/RP6
12 NC
13 NC
14 RB4/CCP4/PMA1/KBI0/RP7
15 RB5/CCP5/PMA0/KBI1/RP8
16 RB6/CCP6/KBI2/PGC/RP9
17 RB7/CCP7/KBI3/PGD/RP10
18 MCLR
19 RA0/AN0/C1INA/ULPWU/PMA6/RP0
20 RA1/AN1/C2INA/VBG/CTDIN/PMA7/RP1
21 RA2/AN2/C2INB/C1IND/C3INB/VREF-/CVREF
22 RA3/AN3/C1INB/VREF+
23 VDDCORE/VCAP
24 RA5/AN4/C1INC/SS1/HLVDIN/RP2
25 RE0/AN5/PMRD
26 RE1/AN6/PMWR
27 RE2/AN7/PMCS
28 VDD2
29 VSS2
30 OSC1/CLKI/RA7
31 OSC2/CLKO/RA6
32 RC0/T1OSO/T1CKI/RP11
33 NC
34 NC
35 RC1/CCP8/T1OSI/RP12
36 RC2/AN11/C2IND/CTPLS/RP13
37 RC3/SCK1/SCL1/RP14
38 RD0/PMD0/SCL2
39 RD1/PMD1/SDA2
40 RD2/PMD2/RP19
41 RD3/PMD3/RP20
42 RC4/SDI1/SDA1/RP15
43 RC5/SDO1/RP16
44 RC6/CCP9/PMA5/TX1/CK1/RP17

Legend: **RPn** represents remappable pins.Some input and output functions are routed through the Peripheral Pin Select (PPS) module and can be dynamically assigned to any of the RPn pins. For a list of the input and output functions, see Table 10-13 and Table 10-14, respectively. For details on configuring the PPS module, see **Section 10.7 “Peripheral Pin Select (PPS)”**.

Pin Diagrams (Continued)

44-Pin QFN


▨ = Pins are up to 5.5V tolerant
PIC18F4XJ13
1 RC7/CCP10/PMA4/RX1/DT1/**RP18**
2 RD4/PMD4/**RP21**
3 RD5/PMD5/**RP22**
4 RD6/PMD6/**RP23**
5 RD7/PMD7/**RP24**
6 VSS1
7 AVDD1
8 VDD1
9 RB0/AN12/C3IND/INT0/**RP3**
10 RB1/AN10/C3INC/PMBE/RTCC/**RP4**
11 RB2/AN8/C2INC/CTED1/PMA3/REFO/**RP5**
12 RB3/AN9/C3INA/CTED2/PMA2/**RP6**
13 NC
14 RB4/CCP4/PMA1/KBI0/**RP7**
15 RB5/CCP5/PMA0/KBI1/**RP8**
16 RB6/CCP6/KBI2/PGC/**RP9**
17 RB7/CCP7/KBI3/PGD/**RP10**
18 MCLR
19 RA0/AN0/C1INA/ULPWU/PMA6/**RP0**
20 RA1/AN1/C2INA/VBG/CTDIN/PMA7/**RP1**
21 RA2/AN2/C2INB/C1IND/C3INB/VREF-/CVREF
22 RA3/AN3/C1INB/VREF+
23 VDDCORE/VCAP
24 RA5/AN4/C1INC/SS1/HLVDIN/**RP2**
25 RE0/AN5/PMRD
26 RE1/AN6/PMWR
27 RE2/AN7/PMCS
28 AVDD2
29 VDD2
30 AVSS1
31 VSS2
32 OSC1/CLKI/RA7
33 OSC2/CLKO/RA6
34 RC0/T1OSO/T1CKI/**RP11**
35 RC1/CCP8/T1OSI/**RP12**
36 RC2/AN11/C2IND/CTPLS/**RP13**
37 RC3/SCK1/SCL1/**RP14**
38 RD0/PMD0/SCL2
39 RD1/PMD1/SDA2
40 RD2/PMD2/**RP1**
41 RD3/PMD3/**RP20**
42 RC4/SDI1/SDA1/**RP15**
43 RC5/SDO1/**RP16**
44 RC6/CCP9/PMA5/TX1/CK1/**RP17**

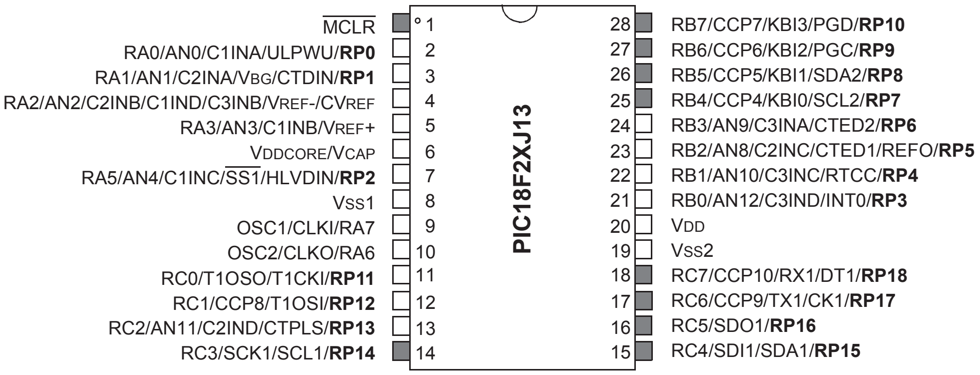
Legend: **RPn** represents remappable pins.Some input and output functions are routed through the Peripheral Pin Select (PPS) module and can be dynamically assigned to any of the RPn pins. For a list of the input and output functions, see Table 10-13 and Table 10-14, respectively. For details on configuring the PPS module, see **Section 10.7 "Peripheral Pin Select (PPS)"**.

Note: For the QFN package, it is recommended that the bottom pad be connected to VSS.

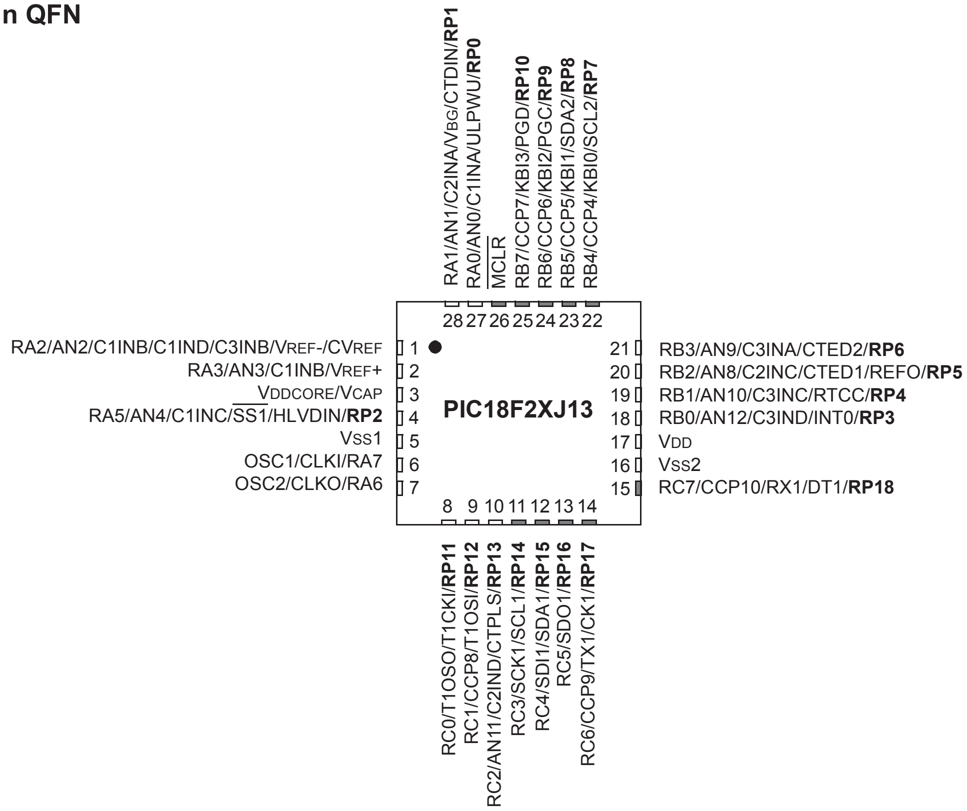
Pin Diagrams (Continued)

28-Pin SPDIP/SOIC/SSOP

▨ = Pins are up to 5.5V tolerant


MCLR
RA0/AN0/C1INA/ULPWU/RP0
RA1/AN1/C2INA/VBG/CTDIN/RP1
RA2/AN2/C2INB/C1IND/C3INB/VREF-/CVREF
RA3/AN3/C1INB/VREF+
VDDCORE/VCAP
RA5/AN4/C1INC/SS1/HLVDIN/RP2
VSS1
OSC1/CLKI/RA7
OSC2/CLKO/RA6
RC0/T1OSO/T1CKI/RP11
RC1/CCP8/T1OSI/RP12
RC2/AN11/C2IND/CTPLS/RP13
RC3/SCK1/SCL1/RP14
1
2
3
4
5
6
7
8
9
10
11
12
13
14
PIC18F2XJ13
28
27
26
25
24
23
22
21
20
19
18
17
16
15
RB7/CCP7/KBI3/PGD/RP10
RB6/CCP6/KBI2/PGC/RP9
RB5/CCP5/KBI1/SDA2/RP8
RB4/CCP4/KBI0/SCL2/RP7
RB3/AN9/C3INA/CTED2/RP6
RB2/AN8/C2INC/CTED1/REFO/RP5
RB1/AN10/C3INC/RTCC/RP4
RB0/AN12/C3IND/INT0/RP3
VDD
VSS2
RC7/CCP10/RX1/DT1/RP18
RC6/CCP9/TX1/CK1/RP17
RC5/SDO1/RP16
RC4/SDI1/SDA1/RP15

28-Pin QFN


RA1/AN1/C2INA/VBG/CTDIN/RP1
RA0/AN0/C1INA/ULPWU/RP0
MCLR
RB7/CCP7/KBI3/PGD/RP10
RB6/CCP6/KBI2/PGC/RP9
RB5/CCP5/KBI1/SDA2/RP8
RB4/CCP4/KBI0/SCL2/RP7
28 27 26 25 24 23 22
RA2/AN2/C1INB/C1IND/C3INB/VREF-/CVREF
RA3/AN3/C1INB/VREF+
VDDCORE/VCAP
RA5/AN4/C1INC/SS1/HLVDIN/RP2
VSS1
OSC1/CLKI/RA7
OSC2/CLKO/RA6
1
2
3
4
5
6
7
PIC18F2XJ13
21
20
19
18
17
16
15
RB3/AN9/C3INA/CTED2/RP6
RB2/AN8/C2INC/CTED1/REFO/RP5
RB1/AN10/C3INC/RTCC/RP4
RB0/AN12/C3IND/INT0/RP3
VDD
VSS2
RC7/CCP10/RX1/DT1/RP18
8 9 10 11 12 13 14
RC0/T1OSO/T1CKI/RP11
RC1/CCP8/T1OSI/RP12
RC2/AN11/C2IND/CTPLS/RP13
RC3/SCK1/SCL1/RP14
RC4/SDI1/SDA1/RP15
RC5/SDO1/RP16
RC6/CCP9/TX1/CK1/RP17

Legend: **RPn** represents remappable pins.Some input and output functions are routed through the Peripheral Pin Select (PPS) module and can be dynamically assigned to any of the RPn pins. For a list of the input and output functions, see Table 10-13 and Table 10-14, respectively. For details on configuring the PPS module, see **Section 10.7 "Peripheral Pin Select (PPS)"**.

Note: For the QFN package, it is recommended that the bottom pad be connected to VSS.

Table of Contents

1.0 Device Overview 9
2.0 Guidelines for Getting Started with PIC18FJ Microcontrollers 28
3.0 Oscillator Configurations 32
4.0 Low-Power Modes 43
5.0 Reset 60
6.0 Memory Organization 75
7.0 Flash Program Memory 101
8.0 8 x 8 Hardware Multiplier 111
9.0 Interrupts 113
10.0 I/O Ports 133
11.0 Parallel Master Port (PMP) 173
12.0 Timer0 Module 198
13.0 Timer1 Module 202
14.0 Timer2 Module 212
15.0 Timer3/5 Module 214
16.0 Timer4/6/8 Module 226
17.0 Real-Time Clock and Calendar (RTCC) 229
18.0 Capture/Compare/PWM (CCP) Modules 248
19.0 Enhanced Capture/Compare/PWM (ECCP) Module 260
20.0 Master Synchronous Serial Port (MSSP) Module 282
21.0 Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) 336
22.0 10/12-Bit Analog-to-Digital Converter (A/D) Module 358
23.0 Comparator Module 370
24.0 Comparator Voltage Reference Module 377
25.0 High/Low Voltage Detect (HLVD) 380
26.0 Charge Time Measurement Unit (CTMU) 386
27.0 Special Features of the CPU 403
28.0 Instruction Set Summary 420
29.0 Development Support 470
30.0 Electrical Characteristics 474
31.0 Packaging Information 516
Appendix A: Revision History 533
Appendix B: Migration From PIC18F46J11 to PIC18F47J13 534
The Microchip Website 535
Customer Change Notification Service 535
Customer Support 535
Product Identification System 536

TO OUR VALUED CUSTOMERS

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

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

Most Current Data Sheet

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

http://www.microchip.com

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

Errata

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

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

- Microchip's Worldwide Website; **http://www.microchip.com**
- Your local Microchip sales office (see last page)

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

Customer Notification System

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

1.0 DEVICE OVERVIEW

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

• PIC18F26J13	• PIC18LF26J13
• PIC18F27J13	• PIC18LF27J13
• PIC18F46J13	• PIC18LF46J13
• PIC18F47J13	• PIC18LF47J13

1.1 Core Features

1.1.1 XLP TECHNOLOGY

All of the devices in the PIC18F47J13 Family incorporate a range of features that can significantly reduce power consumption during operation. Key features are:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal RC oscillator, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operational requirements.
- **On-the-Fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the users to incorporate power-saving ideas into their application's software design.
- **Deep Sleep:** The 2.5V internal core voltage regulator on F parts can be shutdown to cut power consumption to as low as 15 nA (typical). Certain features can remain operating during Deep Sleep, such as the Real-Time Clock Calendar.
- **Ultra Low Power Wake-Up:** Waking from Sleep or Deep Sleep modes after a period of time can be done without an oscillator/clock source, saving power for applications requiring periodic activity.

1.1.2 OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F47J13 Family offer five different oscillator options, allowing users a range of choices in developing application hardware. These include:

- Two Crystal modes, using crystals or ceramic resonators.
- Two External Clock modes, offering the option of a divide-by-4 clock output.
- An internal oscillator block, which provides an 8 MHz clock and an INTRC source (approximately 31 kHz, stable over temperature and V_{DD}), as well as a range of six user selectable clock frequencies, between 125 kHz to 4 MHz, for a total of eight clock frequencies. This option frees an oscillator pin for use as an additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier available to the high-speed crystal, and external and internal oscillators, providing a clock speed up to 48 MHz.

The internal oscillator block provides a stable reference source that gives the PIC18F47J13 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 (POR), or wake-up from Sleep mode, until the primary clock source is available.

1.1.3 EXPANDED MEMORY

The PIC18F47J13 Family provides ample room for application code, from 64 Kbytes to 128 Kbytes of code space. The Flash cells for program memory are rated to last in excess of 10000 erase/write cycles. Data retention without refresh is conservatively estimated to be greater than 20 years.

The Flash program memory is readable and writable during normal operation. The PIC18F47J13 Family also provides plenty of room for dynamic application data with up to 3.8 Kbytes of data RAM.

1.1.4 EXTENDED INSTRUCTION SET

The PIC18F47J13 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.

1.1.5 EASY MIGRATION

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

The consistent pinout scheme used throughout the entire family also aids in migrating to the next larger device.

The PIC18F47J13 Family is also pin compatible with other PIC18 families, such as the PIC18F4550, PIC18F2450 and PIC18F46J50. This allows a new dimension to the evolution of applications, allowing developers to select different price points within Microchip's PIC18 portfolio, while maintaining the same feature set.

1.2 Other Special Features

- **Communications:** The PIC18F47J13 Family incorporates a range of serial and parallel communication peripherals. This device includes two independent Enhanced USARTs and two Master Synchronous Serial Port (MSSP) modules, capable of both Serial Peripheral Interface (SPI) and I^{2}C (Master and Slave) modes of operation. The device also has a parallel port and can be configured to serve as either a Parallel Master Port (PMP) or as a Parallel Slave Port (PSP).
- **CCP/ECCP Modules:** All devices in the family incorporate seven Capture/Compare/PWM (CCP) modules and three Enhanced Capture/Compare/PWM (ECCP) modules to maximize flexibility in control applications. ECCPs offer up to four PWM output signals each. The ECCPs also offer many beneficial features, including polarity selection, programmable dead time, auto-shutdown and restart and Half-Bridge and Full-Bridge Output modes.
- **10/12-Bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period, and thus, reducing code overhead.
- **Extended Watchdog Timer (WDT):** This enhanced version incorporates a 16-bit prescaler, allowing an extended time-out range that is stable across operating voltage and temperature. See **Section 30.0 "Electrical Characteristics"** for time-out periods.

1.3 Details on Individual Family Devices

Devices in the PIC18F47J13 Family are available in 28-pin and 44-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 two ways:

- Flash program memory (two sizes: 64 Kbytes for the PIC18FX6J13 and 128 Kbytes for PIC18FX7J13)
- I/O ports (three bidirectional ports on 28-pin devices, five bidirectional ports on 44-pin devices)

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

The pinouts for the PIC18F2XJ13 devices are listed in Table 1-3. The pinouts for the PIC18F4XJ13 devices are shown in Table 1-4.

The PIC18F47J13 Family of devices provides an on-chip voltage regulator to supply the correct voltage levels to the core. Parts designated with an "F" part number (such as PIC18**F**47J13) have the voltage regulator enabled.

These parts can run from 2.15V-3.6V on VDD, but should have the VDDCORE pin connected to VSS through a low-ESR capacitor. Parts designated with an "LF" part number (such as PIC18**LF**47J13) do not enable the voltage regulator nor support Deep Sleep mode. For "LF" parts, an external supply of 2.0V-2.7V has to be supplied to the VDDCORE pin while 2.0V-3.6V can be supplied to VDD (VDDCORE should never exceed VDD).

For more details about the internal voltage regulator, see **Section 27.3 "On-Chip Voltage Regulator"**.

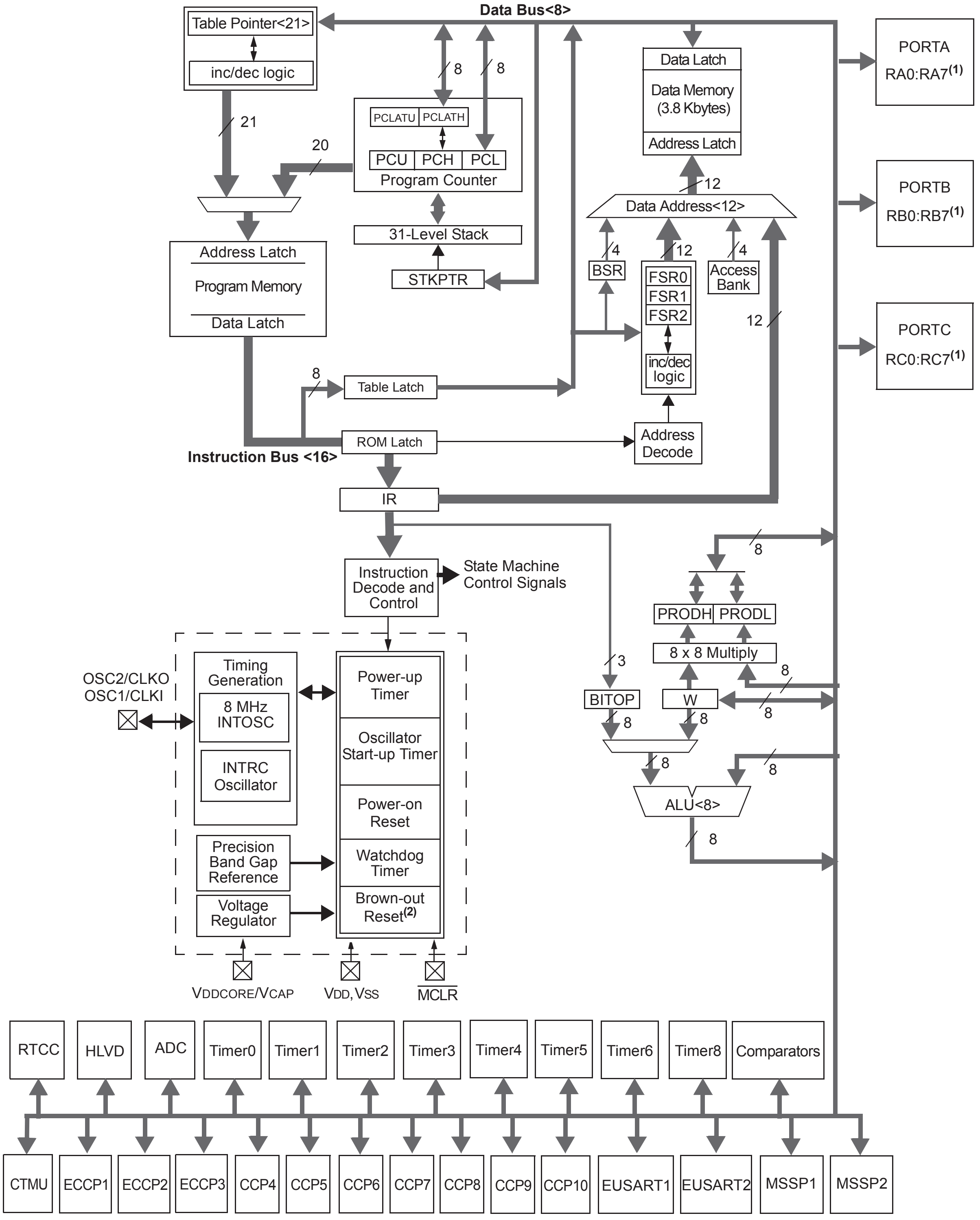
TABLE 1-1: DEVICE FEATURES FOR THE PIC18F2XJ13 (28-PIN DEVICES)

Features	PIC18F26J13	PIC18F27J13
Operating Frequency	DC – 48 MHz	DC – 48 MHz
Program Memory (Kbytes)	64	128
Program Memory (Instructions)	32,768	65,536
Data Memory (Kbytes)	3.8	3.8
Interrupt Sources	30	
I/O Ports	Ports A, B, C	
Timers	8	
Enhanced Capture/Compare/PWM Modules	3 ECCP and 7 CCP	
Serial Communications	MSSP (2), Enhanced USART (2)	
Parallel Communications (PMP/PSP)	No	
10/12-Bit Analog-to-Digital Module	10 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	28-Pin QFN, SOIC, SSOP and SPDIP (300 mil)	

TABLE 1-2: DEVICE FEATURES FOR THE PIC18F4XJ13 (44-PIN DEVICES)

Features	PIC18F46J13	PIC18F47J13
Operating Frequency	DC – 48 MHz	DC – 48 MHz
Program Memory (Kbytes)	64	128
Program Memory (Instructions)	32,768	65,536
Data Memory (Kbytes)	3.8	3.8
Interrupt Sources	30	
I/O Ports	Ports A, B, C, D, E	
Timers	8	
Enhanced Capture/Compare/PWM Modules	3 ECCP and 7 CCP	
Serial Communications	MSSP (2), Enhanced USART (2)	
Parallel Communications (PMP/PSP)	Yes	
10/12-Bit Analog-to-Digital Module	13 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	44-Pin QFN and TQFP	

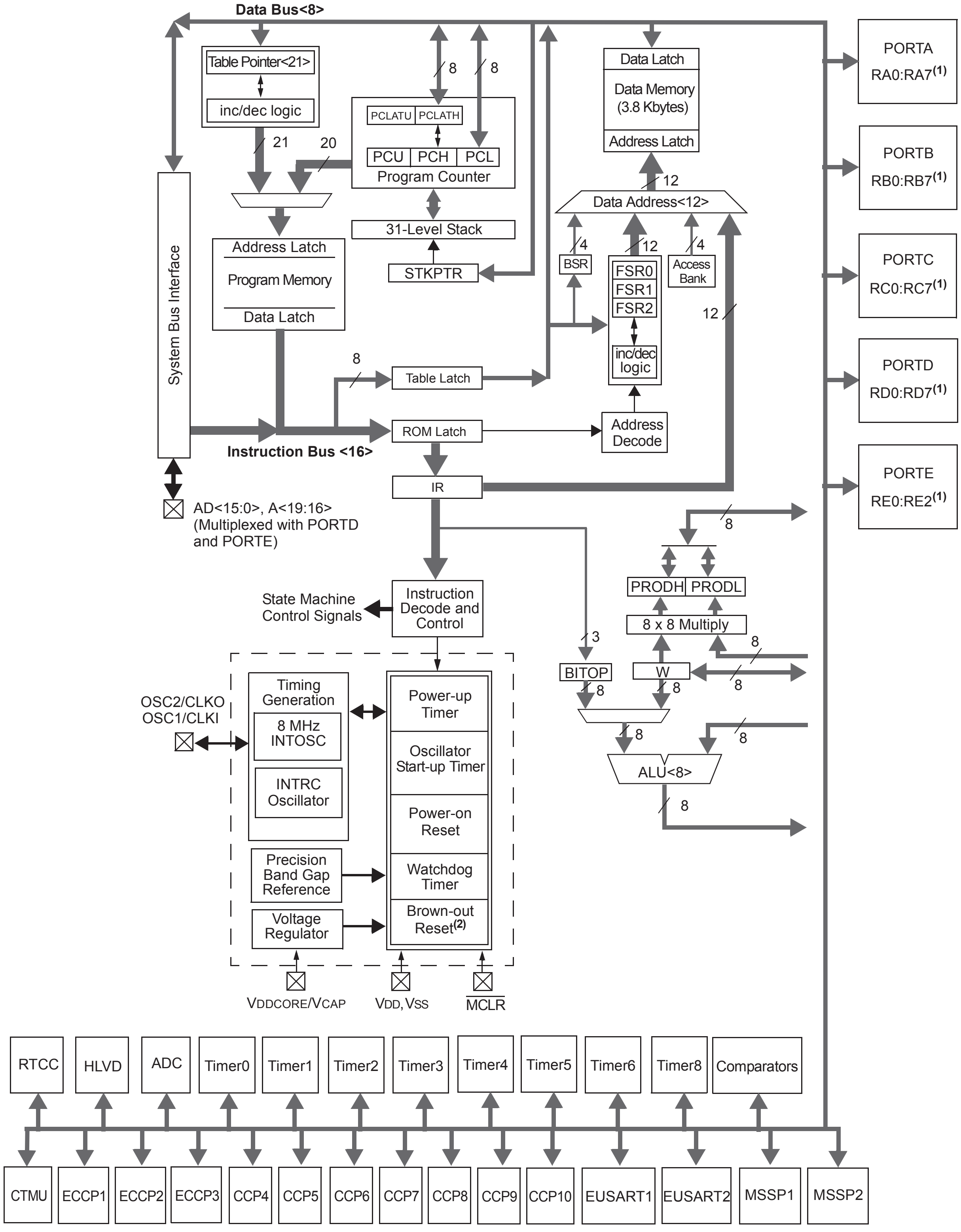
FIGURE 1-1: PIC18F2XJ13 (28-PIN) BLOCK DIAGRAM


Data Bus<8>
Table Pointer<21>
inc/dec logic
21
20
8
8
PCLATU
PCLATH
PCU
PCH
PCL
Program Counter
31-Level Stack
STKPTR
Address Latch
Program Memory
Data Latch
8
Table Latch
ROM Latch
Instruction Bus <16>
IR
Data Latch
Data Memory (3.8 Kbytes)
Address Latch
12
Data Address<12>
4
12
4
BSR
FSR0
FSR1
FSR2
inc/dec logic
Access Bank
12
Address Decode
PORTA
RA0:RA7(1)
PORTB
RB0:RB7(1)
PORTC
RC0:RC7(1)
Instruction Decode and Control
State Machine Control Signals
8
PRODH
PRODL
8 x 8 Multiply
3
8
BITOP
W
8
8
8
8
8
ALU<8>
8
OSC2/CLKO
OSC1/CLKI
Timing Generation
8 MHz INTOSC
INTRC Oscillator
Precision Band Gap Reference
Voltage Regulator
Power-up Timer
Oscillator Start-up Timer
Power-on Reset
Watchdog Timer
Brown-out Reset(2)
VDDCORE/VCAP
VDD, VSS
MCLR
RTCC
HLVD
ADC
Timer0
Timer1
Timer2
Timer3
Timer4
Timer5
Timer6
Timer8
Comparators
CTMU
ECCP1
ECCP2
ECCP3
CCP4
CCP5
CCP6
CCP7
CCP8
CCP9
CCP10
EUSART1
EUSART2
MSSP1
MSSP2

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

2: BOR functionality is provided when the on-board voltage regulator is enabled.

FIGURE 1-2: PIC18F4XJ13 (44-PIN) BLOCK DIAGRAM


Data Bus<8>
Table Pointer<21>
inc/dec logic
21
20
8
8
PCLATU
PCLATH
PCU
PCH
PCL
Program Counter
31-Level Stack
STKPTR
Data Latch
Data Memory
(3.8 Kbytes)
Address Latch
12
Data Address<12>
4
BSR
12
FSR0
FSR1
FSR2
inc/dec logic
4
Access Bank
12
Address Latch
Program Memory
Data Latch
System Bus Interface
8
Table Latch
ROM Latch
Address Decode
Instruction Bus <16>
IR
AD<15:0>, A<19:16>
(Multiplexed with PORTD and PORTE)
PORTA
RA0:RA7(1)
PORTB
RB0:RB7(1)
PORTC
RC0:RC7(1)
PORTD
RD0:RD7(1)
PORTE
RE0:RE2(1)
8
PRODH
PRODL
8 x 8 Multiply
8
3
BITOP
W
8
8
8
8
8
ALU<8>
8
State Machine Control Signals
Instruction Decode and Control
OSC2/CLKO
OSC1/CLKI
Timing Generation
8 MHz INTOSC
INTRC Oscillator
Precision Band Gap Reference
Voltage Regulator
Power-up Timer
Oscillator Start-up Timer
Power-on Reset
Watchdog Timer
Brown-out Reset(2)
VDDCORE/VCAP
VDD, VSS
MCLR
RTCC
HLVD
ADC
Timer0
Timer1
Timer2
Timer3
Timer4
Timer5
Timer6
Timer8
Comparators
CTMU
ECCP1
ECCP2
ECCP3
CCP4
CCP5
CCP6
CCP7
CCP8
CCP9
CCP10
EUSART1
EUSART2
MSSP1
MSSP2

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

2: The on-chip voltage regulator is always enabled by default.

TABLE 1-3: PIC18F2XJ13 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
$\overline{\text{MCLR}}$	1(2)	26(2)	I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
OSC1/CLKI/RA7	9	6			Oscillator crystal or external clock input.
OSC1			I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; CMOS otherwise. Main oscillator input connection.
CLKI			I	CMOS	External clock source input; always associated with pin function, OSC1 (see related OSC1/CLKI pins).
RA7(1)			I/O	TTL/DIG	Digital I/O.
OSC2/CLKO/RA6	10	7			Oscillator crystal or clock output.
OSC2			O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O	DIG	Main oscillator feedback output connection. In RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6(1)			I/O	TTL/DIG	Digital I/O.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^2C = Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.
2: 5.5V tolerant.

TABLE 1-3: PIC18F2XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
					PORTA is a bidirectional I/O port.
RA0/AN0/C1INA/ULPWU/RP0	2	27			
RA0			I/O	TTL/DIG	Digital I/O.
AN0			I	Analog	Analog Input 0.
C1INA			I	Analog	Comparator 1 Input A.
ULPWU			I	Analog	Ultra low-power wake-up input.
RP0			I/O	ST/DIG	Remappable Peripheral Pin 0 input/output.
RA1/AN1/C2INA/VBG/CTDIN/ RP1	3	28			
RA1			I/O	TTL/DIG	Digital I/O.
AN1			O	Analog	Analog Input 1.
C2INA			I	Analog	Comparator 2 Input A.
VBG			O	Analog	Band Gap Reference Voltage (VBG) output.
CTDIN			I	ST	CTMU pulse delay input.
RP1			I/O	ST/DIG	Remappable Peripheral Pin 1 input/output.
RA2/AN2/C2INB/C1IND/ C3INB/VREF-/CVREF	4	1			
RA2			I/O	TTL/DIG	Digital I/O.
AN2			I	Analog	Analog Input 2.
C2INB			I	Analog	Comparator 2 Input B.
C1IND			I	Analog	Comparator 1 Input D.
C3INB			I	Analog	Comparator 3 Input B.
VREF-			O	Analog	A/D reference voltage (low) input.
CVREF			I	Analog	Comparator reference voltage output.
RA3/AN3/C1INB/VREF+	5	2			
RA3			I/O	TTL/DIG	Digital I/O.
AN3			I	Analog	Analog Input 3.
C1INB			I	Analog	Comparator 1 Input B.
VREF+			I	Analog	A/D reference voltage (high) input.
RA5/AN4/C1INC/$\overline{\text{SS1}}$/ HLVDIN/RP2	7	4			
RA5			I/O	TTL/DIG	Digital I/O.
AN4			I	Analog	Analog Input 4.
C1INC			I	Analog	Comparator 1 Input C.
$\overline{\text{SS1}}$			I	TTL	SPI slave select input.
HLVDIN			I	Analog	High/Low-Voltage Detect input.
RP2			I/O	ST/DIG	Remappable Peripheral Pin 2 input/output.
RA6(1)					See the OSC2/CLKO/RA6 pin.
RA7(1)					See the OSC1/CLKI/RA7 pin.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^2C = Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

2: 5.5V tolerant.

TABLE 1-3: PIC18F2XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/C3IND/INT0/RP3	21	18			
RB0			I/O	TTL/DIG	Digital I/O.
AN12			I	Analog	Analog Input 12.
C3IND			I	Analog	Comparator 3 Input D.
INT0			I	ST	External Interrupt 0.
RP3			I/O	ST/DIG	Remappable Peripheral Pin 3 input/output.
RB1/AN10/C3INC/RTCC/RP4	22	19			
RB1			I/O	TTL/DIG	Digital I/O.
AN10			I	Analog	Analog Input 10.
C3INC			I	Analog	Comparator 3 input.
RTCC			O	DIG	Real-Time Clock Calendar output.
RP4			I/O	ST/DIG	Remappable Peripheral Pin 4 input/output.
RB2/AN8/C2INC/CTED1/ REFO/RP5	23	20			
RB2			I/O	TTL/DIG	Digital I/O.
AN8			I	Analog	Analog Input 8.
C2INC			I	Analog	Comparator 2 Input C.
CTED1			I	ST	CTMU Edge 1 input.
REFO			O	DIG	Reference output clock.
RP5			I/O	ST/DIG	Remappable Peripheral Pin 5 input/output.
RB3/AN9/C3INA/CTED2/ RP6	24	21			
RB3			I/O	TTL/DIG	Digital I/O.
AN9			I	Analog	Analog Input 9.
C3INA			I	Analog	Comparator 3 Input A.
CTED2			I	ST	CTMU edge 2 Input.
RP6			I	ST/DIG	Remappable Peripheral Pin 6 input/output.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^{2}C = Open-Drain, I^{2}C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

2: 5.5V tolerant.

TABLE 1-3: PIC18F2XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number 28-SPDIP/ SSOP/ SOIC	Pin Number 28-QFN	Pin Type	Buffer Type	Description
					PORTB (continued)
RB4/CCP4/KBI0/SCL2/RP7	25[2]	22[2]			
RB4			I/O	TTL/DIG	Digital I/O.
CCP4			I/O	ST/DIG	Capture/Compare/PWM input/output.
KBI0			I	TTL	Interrupt-on-change pin.
SCL2			I/O	I^2C	I^2C clock input/output.
RP7			I/O	ST/DIG	Remappable Peripheral Pin 7 input/output.
RB5/CCP5/KBI1/SDA2/RP8	26[2]	23[2]			
RB5			I/O	TTL/DIG	Digital I/O.
CCP5			I/O	ST/DIG	Capture/Compare/PWM input/output.
KBI1			I	TTL	Interrupt-on-change pin.
SDA2			I/O	I^2C	I^2C data input/output.
RP8			I/O	ST/DIG	Remappable Peripheral Pin 8 input/output.
RB6/CCP6/KBI2/PGC/RP9	27[2]	24[2]			
RB6			I/O	TTL/DIG	Digital I/O.
CCP6			I/O	ST/DIG	Capture/Compare/PWM input/output.
KBI2			I	TTL	Interrupt-on-change pin.
PGC			I	ST	ICSP™ clock input.
RP9			I/O	ST/DIG	Remappable Peripheral Pin 9 input/output.
RB7/CCP7/KBI3/PGD/RP10	28[2]	25[2]			
RB7			I/O	TTL/DIG	Digital I/O.
CCP7			I/O	ST/DIG	Capture/Compare/PWM input/output.
KBI3			I	TTL	Interrupt-on-change pin.
PGD			I/O	ST/DIG	In-Circuit Debugger and ICSP programming data pin.
RP10			I/O	ST/DIG	Remappable Peripheral Pin 10 input/output.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^2C = Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

2: 5.5V tolerant.

TABLE 1-3: PIC18F2XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI/RP11	11	8			
RC0			I/O	ST/DIG	Digital I/O.
T1OSO			O	Analog	Timer1 oscillator output.
T1CKI			I	ST	Timer1 external digital clock input.
RP11			I/O	ST/DIG	Remappable Peripheral Pin 11 input/output.
RC1/CCP8/T1OSI/RP12	12	9			
RC1			I/O	ST/DIG	Digital I/O.
CCP8			I/O	ST/DIG	Capture/Compare/PWM input/output.
T1OSI			I	Analog	Timer1 oscillator input.
RP12			I/O	ST/DIG	Remappable Peripheral Pin 12 input/output.
RC2/AN11/C2IND/CTPLS/RP13	13	10			
RC2			I/O	ST/DIG	Digital I/O.
AN11			I	Analog	Analog Input 11.
C2IND			I	Analog	Comparator 2 Input D.
CTPLS			O	DIG	CTMU pulse generator output.
RP13			I/O	ST/DIG	Remappable Peripheral Pin 13 input/output.
RC3/SCK1/SCL1/RP14	14	11			
RC3			I/O	ST/DIG	Digital I/O.
SCK1			I/O	ST/DIG	SPI clock input/output.
SCL1			I/O	I^{2}C	I^{2}C clock input/output.
RP14			I/O	ST/DIG	Remappable Peripheral Pin 14 input/output.
RC4/SDI1/SDA1/RP15	15	12			
RC4			I/O	ST/DIG	Digital I/O.
SDI1			I	ST	SPI data input.
SDA1			I/O	I^{2}C	I^{2}C data input/output.
RP15			I/O	ST/DIG	Remappable Peripheral Pin 15 input/output.
RC5/SDO1/RP16	16	13			
RC5			I/O	ST/DIG	Digital I/O.
SDO1			O	DIG	SPI data output.
RP16			I/O	ST/DIG	Remappable Peripheral Pin 16 input/output.
RC6/CCP9/TX1/CK1/RP17	17(2)	14(2)			
RC6			I/O	ST/DIG	Digital I/O.
CCP9			I/O	ST/DIG	Capture/Compare/PWM input/output.
TX1			O	DIG	EUSART1 asynchronous transmit.
CK1			I/O	ST/DIG	EUSART1 synchronous clock (see related RX1/DT1).
RP17			I/O	ST/DIG	Remappable Peripheral Pin 17 input/output.
RC7/CCP10/RX1/DT1/RP18	18(2)	15(2)			
RC7			I/O	ST/DIG	Digital I/O.
CCP10			I/O	ST/DIG	Capture/Compare/PWM input/output.
RX1			I	ST	Asynchronous serial receive data input.
DT1			I/O	ST/DIG	Synchronous serial data output/input.
RP18			I/O	ST/DIG	Remappable Peripheral Pin 18 input/output.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^{2}C = Open-Drain, I^{2}C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.
2: 5.5V tolerant.

TABLE 1-3: PIC18F2XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	28-SPDIP/ SSOP/ SOIC	28-QFN			
VSS1	8	5	P	—	Ground reference for logic and I/O pins.
VSS2	19	16	—	—	
VDD	20	17	P	—	Positive supply for peripheral digital logic and I/O pins.
VDDCORE/VCAP	6	3	—	—	Core logic power or external filter capacitor connection.
VDDCORE			P	—	Positive supply for microcontroller core logic (regulator disabled).
VCAP			P	—	External filter capacitor connection (regulator enabled).

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^2C = Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

2: 5.5V tolerant.

TABLE 1-4: PIC18F4XJ13 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
$\overline{\text{MCLR}}$	18[3]	18	I	ST	Master Clear (Reset) input; this is an active-low Reset to the device.
OSC1/CLKI/RA7	32	30			Oscillator crystal or external clock input.
OSC1			I	ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS. Main oscillator input connection.
CLKI			I	CMOS	External clock source input; always associated with pin function, OSC1 (see related OSC1/CLKI pins).
RA7[1]			I/O	TTL/DIG	Digital I/O.
OSC2/CLKO/RA6	33	31			Oscillator crystal or clock output.
OSC2			O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO			O	—	Main oscillator feedback output connection in RC mode, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6[1]			I/O	TTL/DIG	Digital I/O.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^2C = Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.
2: Available only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: 5.5V tolerant.

TABLE 1-4: PIC18F4XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTA is a bidirectional I/O port.
RA0/AN0/C1INA/ULPWU/PMA6/RP0	19	19			
RA0			I/O	TTL/DIG	Digital I/O.
AN0			I	Analog	Analog Input 0.
C1INA			I	Analog	Comparator 1 Input A.
ULPWU			I	Analog	Ultra low-power wake-up input.
PMA6			I/O	ST/TTL/DIG	Parallel Master Port digital I/O.
RP0			I/O	ST/DIG	Remappable Peripheral Pin 0 input/output.
RA1/AN1/C2INA/VBG/CTDIN/PMA7/RP1	20	20			
RA1			I/O	TTL/DIG	Digital I/O.
AN1			O	Analog	Analog Input 1.
C2INA			I	Analog	Comparator 2 Input A.
VBG			O	Analog	Band Gap Reference Voltage (VBG) output.
CTDIN			I	ST	CTMU pulse delay input.
PMA7			I/O	ST/TTL/DIG	Parallel Master Port digital I/O.
RP1			I/O	ST/DIG	Remappable Peripheral Pin 1 input/output.
RA2/AN2/C2INB/C1IND/C3INB/VREF-/CVREF	21	21			
RA2			I/O	TTL/DIG	Digital I/O.
AN2			I	Analog	Analog Input 2.
C2INB			I	Analog	Comparator 2 Input B.
C1IND			I	Analog	Comparator 1 Input D.
C3INB			I	Analog	Comparator 3 Input B.
VREF-			I	Analog	A/D reference voltage (low) input.
CVREF			I	Analog	Comparator reference voltage output.
RA3/AN3/C1INB/VREF+	22	22			
RA3			I/O	TTL/DIG	Digital I/O.
AN3			I	Analog	Analog Input 3.
C1INB			I	Analog	Comparator 1 Input B.
VREF+			I	Analog	A/D reference voltage (high) input.
RA5/AN4/C1INC/$\overline{\text{SS1}}$/HLVDIN/RP2	24	24			
RA5			I/O	TTL/DIG	Digital I/O.
AN4			I	Analog	Analog Input 4.
C1INC			I	Analog	SPI slave select input.
$\overline{\text{SS1}}$			I	TTL	Comparator 1 Input C.
HLVDIN			I	Analog	High/Low-Voltage Detect input.
RP2			I/O	ST/DIG	Remappable Peripheral Pin 2 input/output.
RA6[1]					See the OSC2/CLKO/RA6 pin.
RA7[1]					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)
DIG	= Digital output	I^2C	= Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

2: Available only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

3: 5.5V tolerant.

TABLE 1-4: PIC18F4XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/C3IND/INT0/RP3	9	8			
RB0			I/O	TTL/DIG	Digital I/O.
AN12			I	Analog	Analog Input 12.
C3IND			I	Analog	Comparator 3 Input D.
INT0			I	ST	External Interrupt 0.
RP3			I/O	ST/DIG	Remappable Peripheral Pin 3 input/output.
RB1/AN10/C3INC/PMBE/RTCC/RP4	10	9			
RB1			I/O	TTL/DIG	Digital I/O.
AN10			I	Analog	Analog Input 10.
C3INC			I	Analog	Comparator 3 Input C.
PMBE(2)			O	DIG	Parallel Master Port byte enable.
RTCC			O	DIG	Asynchronous serial transmit data output.
RP4			I/O	ST/DIG	Remappable Peripheral Pin 4 input/output.
RB2/AN8/C2INC/CTED1/PMA3/REFO/RP5	11	10			
RB2			I/O	TTL/DIG	Digital I/O.
AN8			I	Analog	Analog Input 8.
C2INC			I	Analog	Comparator 2 Input C.
CTED1			I	ST	CTMU Edge 1 input.
PMA3(2)			O	DIG	Parallel Master Port address.
REFO			O	DIG	Reference output clock.
RP5			I/O	ST/DIG	Remappable Peripheral Pin 5 input/output.
RB3/AN9/C3INA/CTED2/PMA2/RP6	12	11			
RB3			I/O	TTL/DIG	Digital I/O.
AN9			I	Analog	Analog Input 9.
C3INA			I	Analog	Comparator 3 Input A.
CTED2			I	ST	CTMU Edge 2 input.
PMA2(2)			O	DIG	Parallel Master Port address.
RP6			I/O	ST/DIG	Remappable Peripheral Pin 6 input/output.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^{2}C = Open-Drain, I^{2}C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.
2: Available only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: 5.5V tolerant.

TABLE 1-4: PIC18F4XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number 44-QFN	44-TQFP	Pin Type	Buffer Type	Description
					PORTB (continued)
RB4/CCP4/PMA1/KBI0/RP7	14[3]	14[3]			
RB4			I/O	TTL/DIG	Digital I/O.
CCP4[2]			I/O	ST/DIG	Capture/Compare/PWM input/output.
PMA1[2]			I/O	ST/TTL/DIG	Parallel Master Port address.
KBI0			I	TTL	Interrupt-on-change pin.
RP7			I/O	ST/DIG	Remappable Peripheral Pin 7 input/output.
RB5/CCP5/PMA0/KBI1/RP8	15[3]	15[3]			
RB5			I/O	TTL/DIG	Digital I/O.
CCP5			I/O	ST/DIG	Capture/Compare/PWM input/output.
PMA0[2]			I/O	ST/TTL/DIG	Parallel Master Port address.
KBI1			I	TTL	Interrupt-on-change pin.
RP8			I/O	ST/DIG	Remappable Peripheral Pin 8 input/output.
RB6/CCP6/KBI2/PGC/RP9	16[3]	16[3]			
RB6			I/O	TTL/DIG	Digital I/O.
CCP6			I/O	ST/DIG	Capture/Compare/PWM input/output.
KBI2			I	TTL	Interrupt-on-change pin.
PGC			I	ST	ICSP™ clock input.
RP9			I/O	ST/DIG	Remappable Peripheral Pin 9 input/output.
RB7/CCP7/KBI3/PGD/RP10	17[3]	17[3]			
RB7			I/O	TTL/DIG	Digital I/O.
CCP7			I/O	ST/DIG	Capture/Compare/PWM input/output.
KBI3			I	TTL	Interrupt-on-change pin.
PGD			I/O	ST/DIG	In-Circuit Debugger and ICSP programming data pin.
RP10			I/O	ST/DIG	Remappable Peripheral Pin 10 input/output.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^2C = Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.
2: Available only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: 5.5V tolerant.

TABLE 1-4: PIC18F4XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTC is a bidirectional I/O port.
RC0/T1OSO/T1CKI/RP11	34	32			
RC0			I/O	STDIG	Digital I/O.
T1OSO			O	Analog	Timer1 oscillator output.
T1CKI			I	ST	Timer1/Timer3 external clock input.
RP11			I/O	ST/DIG	Remappable Peripheral Pin 11 input/output.
RC1/CCP8/T1OSI/RP12	35	35			
RC1			I/O	ST/DIG	Digital I/O.
CCP8			I/O	ST/DIG	Capture/Compare/PWM input/output.
T1OSI			I	Analog	Timer1 oscillator input.
RP12			I/O	ST/DIG	Remappable Peripheral Pin 12 input/output.
RC2/AN11/C2IND/CTPLS/RP13	36	36			
RC2			I/O	ST/DIG	Digital I/O.
AN11			I	Analog	Analog Input 11.
C2IND			I	Analog	Comparator 2 Input D.
CTPLS			O	DIG	CTMU pulse generator output.
RP13			I/O	ST/DIG	Remappable Peripheral Pin 13 input/output.
RC3/SCK1/SCL1/RP14	37	37			
RC3			I/O	ST/DIG	Digital I/O.
SCK1			I/O	ST/DIG	SPI clock input/output.
SCL1			I/O	I^2C	I^2C clock input/output.
RP14			I/O	ST/DIG	Remappable Peripheral Pin 14 input/output.
RC4/SDI1/SDA1/RP15	42	42			
RC4			I/O	ST/DIG	Digital I/O.
SDI1			I	ST	SPI data input.
SDA1			I/O	I^2C	I^2C data input/output.
RP15			I/O	ST/DIG	Remappable Peripheral Pin 15 input/output.
RC5/SDO1/RP16	43	43			
RC5			I/O	ST/DIG	Digital I/O.
SDO1			O	DIG	SPI data output.
RP16			I/O	ST/DIG	Remappable Peripheral Pin 16 input/output.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^2C = Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.
2: Available only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: 5.5V tolerant.

TABLE 1-4: PIC18F4XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTC (continued)
RC6/CCP9/PMA5/TX1/CK1/RP17	44[3]	44[3]			
RC6			I/O	ST/DIG	Digital I/O.
CCP9			I/O	ST/DIG	Capture/Compare/PWM input/output.
PMA5			I/O	DIG	Parallel Master Port address.
TX1			O	ST/TTL/DIG	EUSART1 asynchronous transmit.
CK1			I/O	ST/DIG	EUSART1 synchronous clock (see related RX1/DT1).
RP17			I/O	ST/DIG	Remappable Peripheral Pin 17 input/output.
RC7/CCP10/PMA4/RX1/DT1/RP18	1[3]	1[3]			
RC7			I/O	ST/DIG	Digital I/O.
CCP10			I/O	ST/DIG	Capture/Compare/PWM input/output.
PMA4			I/O	ST/TTL/DIG	Parallel Master Port address.
RX1			I	ST	EUSART1 asynchronous receive.
DT1			I/O	ST/DIG	EUSART Synchronous data (see related TX1/CK1).
RP18			I/O	ST/DIG	Remappable Peripheral Pin 18 input/output.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^2C = Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.

2: Available only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

3: 5.5V tolerant.

TABLE 1-4: PIC18F4XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTD is a bidirectional I/O port.
RD0/PMD0/SCL2	38(3)	38(3)			
RD0			I/O	ST/DIG	Digital I/O.
PMD0			I/O	ST/TTL/DIG	Parallel Master Port data.
SCL2			I/O	I^{2}C	I^{2}C data input/output.
RD1/PMD1/SDA2	39(3)	39(3)			
RD1			I/O	ST/DIG	Digital I/O.
PMD1			I/O	ST/TTL/DIG	Parallel Master Port data.
SDA2			I/O	I^{2}C	I^{2}C data input/output.
RD2/PMD2/RP19	40(3)	40(3)			
RD2			I/O	ST/DIG	Digital I/O.
PMD2			I/O	ST/TTL/DIG	Parallel Master Port data.
RP19			I/O	ST/DIG	Remappable Peripheral Pin 19 input/output.
RD3/PMD3/RP20	41(3)	41(3)			
RD3			I/O	ST/DIG	Digital I/O.
PMD3			I/O	ST/TTL/DIG	Parallel Master Port data.
RP20			I/O	ST/DIG	Remappable Peripheral Pin 20 input/output.
RD4/PMD4/RP21	2(3)	2(3)			
RD4			I/O	ST/DIG	Digital I/O.
PMD4			I/O	ST/TTL/DIG	Parallel Master Port data.
RP21			I/O	ST/DIG	Remappable Peripheral Pin 21 input/output.
RD5/PMD5/RP22	3(3)	3(3)			
RD5			I/O	ST/DIG	Digital I/O.
PMD5			I/O	ST/TTL/DIG	Parallel Master Port data.
RP22			I/O	ST/DIG	Remappable Peripheral Pin 22 input/output.
RD6/PMD6/RP23	4(3)	4(3)			
RD6			I/O	ST/DIG	Digital I/O.
PMD6			I/O	ST/TTL/DIG	Parallel Master Port data.
RP23			I/O	ST/DIG	Remappable Peripheral Pin 23 input/output.
RD7/PMD7/RP24	5(3)	5(3)			
RD7			I/O	ST/DIG	Digital I/O.
PMD7			I/O	ST/TTL/DIG	Parallel Master Port data.
RP24			I/O	ST/DIG	Remappable Peripheral Pin 24 input/output.

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

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.
2: Available only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: 5.5V tolerant.

TABLE 1-4: PIC18F4XJ13 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	44-QFN	44-TQFP			
					PORTE is a bidirectional I/O port.
RE0/AN5/PMRD	25	25			
RE0			I/O	ST/DIG	Digital I/O.
AN5			I	Analog	Analog Input 5.
PMRD			I/O	ST/TTL/DIG	Parallel Master Port input/output.
RE1/AN6/PMWR	26	26			
RE1			I/O	ST/DIG	Digital I/O.
AN6			I	Analog	Analog Input 6.
PMWR			I/O	ST/TTL/DIG	Parallel Master Port write strobe.
RE2/AN7/PMCS	27	27			
RE2			I/O	ST/DIG	Digital I/O.
AN7			I	Analog	Analog Input 7.
PMCS			O	DIG	Parallel Master Port byte enable.
VSS1	6	6	P	—	Ground reference for logic and I/O pins.
VSS2	31	29	—	—	
AVSS1	30	—	P	—	Ground reference for analog modules.
VDD1	8	7	P	—	Positive supply for peripheral digital logic and I/O pins.
VDD2	29	28	P	—	
VDDCORE/VCAP	23	23			Core logic power or external filter capacitor connection.
VDDCORE			P	—	Positive supply for microcontroller core logic (regulator disabled).
VCAP			P	—	External filter capacitor connection (regulator enabled).
AVDD1	7	—	P	—	Positive supply for analog modules.
AVDD2	28	—	—	—	Positive supply for analog modules.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
DIG = Digital output
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)
I^2C = Open-Drain, I^2C specific

Note 1: RA7 and RA6 will be disabled if OSC1 and OSC2 are used for the clock function.
2: Available only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: 5.5V tolerant.

2.0 GUIDELINES FOR GETTING STARTED WITH PIC18FJ MICROCONTROLLERS

2.1 Basic Connection Requirements

Getting started with the PIC18F47J13 Family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

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

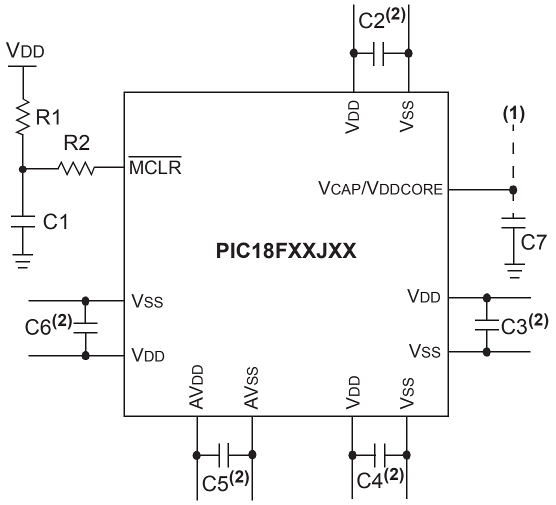
Additionally, the following pins may be required:

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

Note:	On 44-pin QFN packages, the AVDD and AVSS pins must always be connected, regardless of whether any of the analog modules are being used. On other package types, the AVDD and AVSS pins are internally connected to the VDD/VSS pins.

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

FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS



Key (all values are recommendations):

C1 through C6: 0.1 μF, 20V ceramic

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

R1: 10 kΩ

R2: 100Ω to 470Ω

Note 1: See **Section 2.4 “Voltage Regulator Pins (VCAP/VDDCORE)”** for explanation of VCAP/VDDCORE 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.

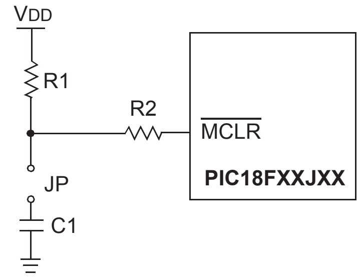
2.3 Master Clear ($\overline{\text{MCLR}}$) Pin

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.

2.4 Voltage Regulator Pins (VCAP/VDDCORE)

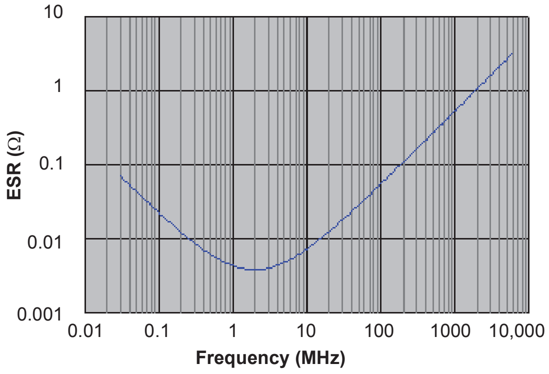
On “F” devices, a low-ESR (< 5Ω) capacitor is required on the VCAP/VDDCORE pin to stabilize the voltage regulator output voltage. The VCAP/VDDCORE pin must not be connected to VDD and must use a capacitor of 10 µF connected to ground. The type can be ceramic or tantalum. A suitable example is the Murata GRM21BF50J106ZE01 (10 µF, 6.3V) or equivalent. 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 30.0 “Electrical Characteristics”** for additional information.

On “LF” devices, the VCAP/VDDCORE pin must be tied to a voltage supply at the VDDCORE level. Refer to **Section 30.0 “Electrical Characteristics”** for information on VDD and VDDCORE.

Note that the “LF” versions of these devices are provided with the voltage regulator permanently disabled; they must always be provided with a supply voltage on the VDDCORE pin.

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



Note: Data for Murata GRM21BF50J106ZE01 shown. Measurements at 25°C, 0V DC bias.

2.5 ICSP Pins

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

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

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

For more information on available Microchip development tools connection requirements, refer to **Section 29.0 “Development Support”**.

2.6 External Oscillator Pins

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

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

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

Layout suggestions are shown in Figure 2-4. 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).

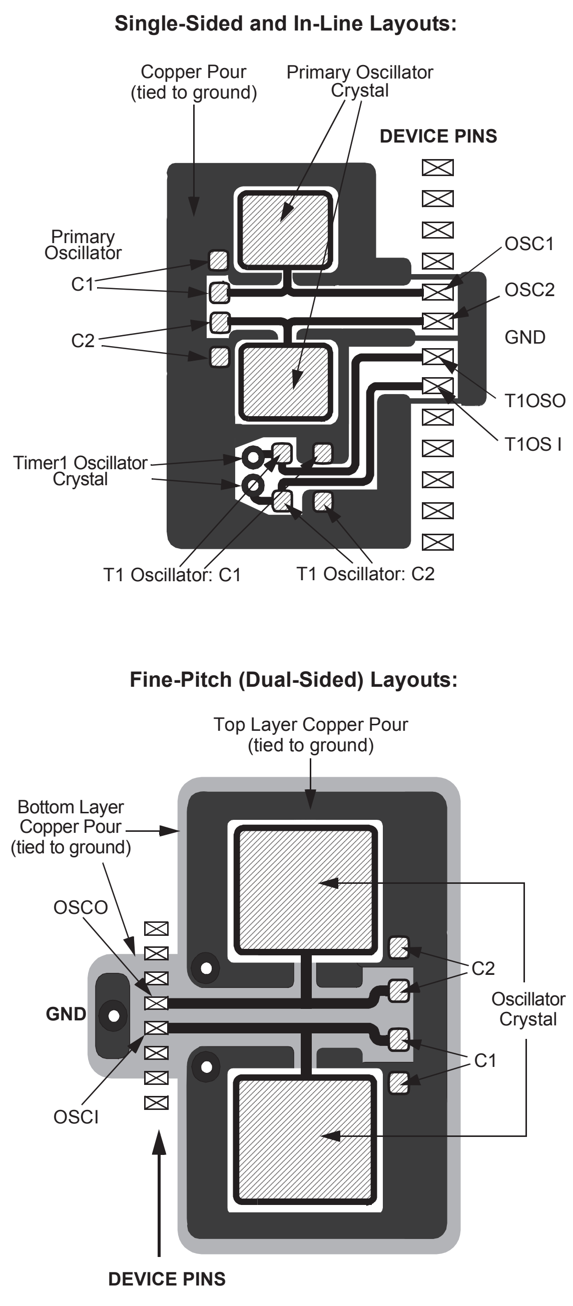
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate website (www.microchip.com):

- *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 2-4: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT


Single-Sided and In-Line Layouts:
Copper Pour (tied to ground)
Primary Oscillator Crystal
DEVICE PINS
Primary Oscillator
C1
C2
OSC1
OSC2
GND
T1OSO
T1OS I
Timer1 Oscillator Crystal
T1 Oscillator: C1
T1 Oscillator: C2
Fine-Pitch (Dual-Sided) Layouts:
Top Layer Copper Pour (tied to ground)
Bottom Layer Copper Pour (tied to ground)
OSCO
GND
OSCI
C2
Oscillator Crystal
C1
DEVICE PINS

3.0 OSCILLATOR CONFIGURATIONS

3.1 Overview

Devices in the PIC18F47J13 Family incorporate a different oscillator and microcontroller clock system than general purpose PIC18F devices.

The PIC18F47J13 Family has additional prescalers and postscalers, which have been added to accommodate a wide range of oscillator frequencies. The PIC18F47J13 provides two PLL circuits: a 4x multiplier PLL and a 96 MHz PLL enabling 48 MHz operation from the 8 MHz internal oscillator. Figure 3-1 provides an overview of the oscillator structure.

Other oscillator features used in PIC18 enhanced microcontrollers, such as the internal oscillator block and clock switching, remain the same. They are discussed later in this chapter.

3.1.1 OSCILLATOR CONTROL

The operation of the oscillator in PIC18F47J13 Family devices is controlled through three Configuration registers and two control registers. Configuration registers, CONFIG1L, CONFIG1H and CONFIG2L, select the oscillator mode, PLL prescaler and CPU divider options. As Configuration bits, these are set when the device is programmed and left in that configuration until the device is reprogrammed.

The OSCCON register (Register 3-2) selects the Active Clock mode. It is primarily used in controlling clock switching in power-managed modes. Its use is discussed in **Section 3.3.1 “Oscillator Control Register”**.

The OSCTUNE register (Register 3-1) is used to trim the INTOSC frequency source and select the low-frequency clock source that drives several special features. The OSCTUNE register is also used to activate or disable the Phase Locked Loop (PLL). Its use is described in **Section 3.2.5.1 “OSCTUNE Register”**.

3.2 Oscillator Types

PIC18F47J13 Family devices can be operated in eight distinct oscillator modes. Users can program the FOSC<2:0> Configuration bits to select one of the modes listed in Table 3-1. For oscillator modes which produce a clock output (CLKO) on pin, RA6, the output frequency will be one fourth of the peripheral clock frequency. The clock output stops when in Sleep mode, but will continue during Idle mode (see Figure 3-1).

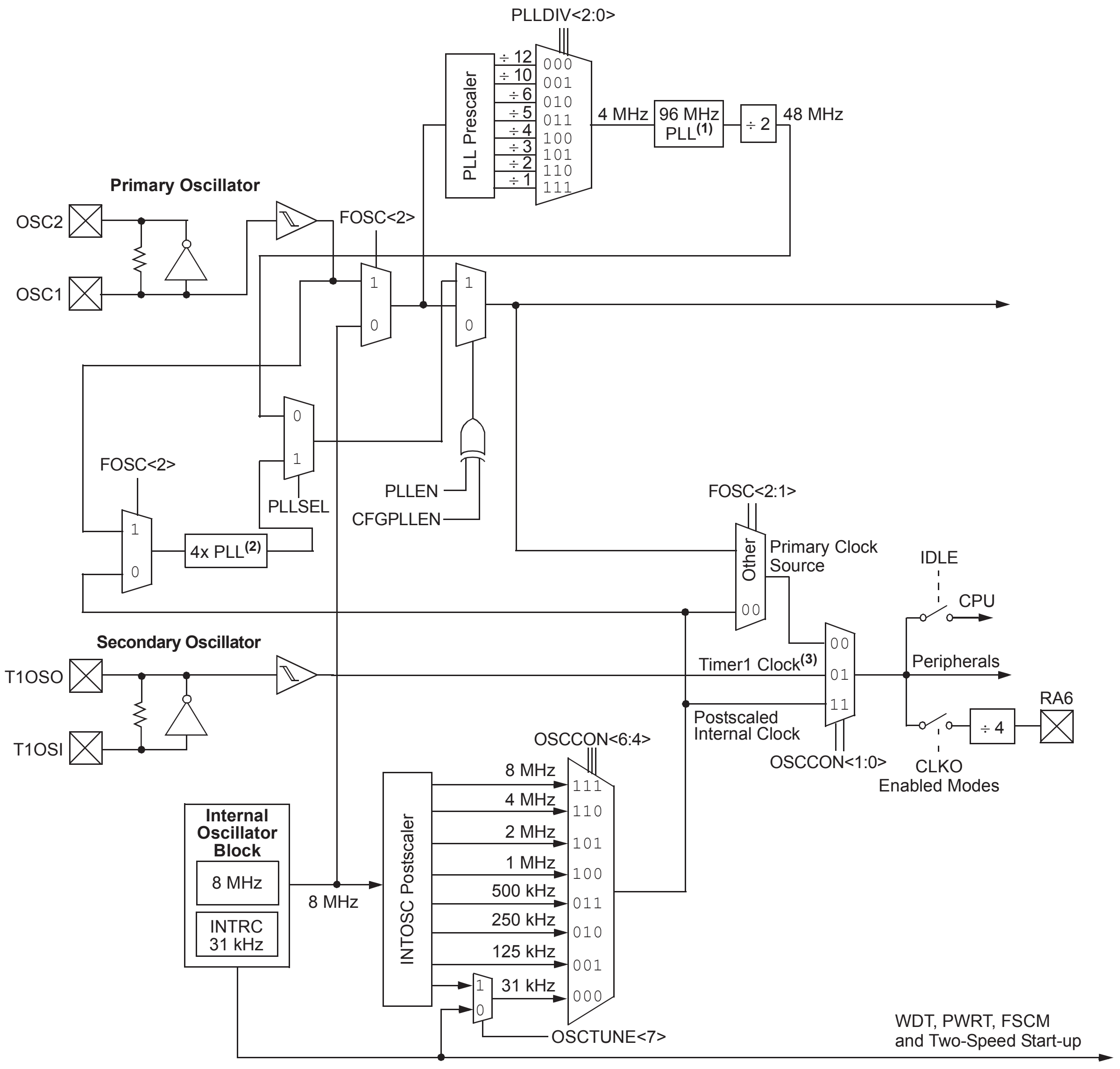
TABLE 3-1: OSCILLATOR MODES

Mode	Description
ECPLL	External Clock Input mode, the PLL can be enabled or disabled in software, CLKO on RA6, apply external clock signal to RA7.
EC	External Clock Input mode, the PLL is always disabled, CLKO on RA6, apply external clock signal to RA7.
HSPLL	High-Speed Crystal/Resonator mode, PLL can be enabled or disabled in software, crystal/resonator connected between RA6 and RA7.
HS	High-Speed Crystal/Resonator mode, PLL always disabled, crystal/resonator connected between RA6 and RA7.
INTOSCPLLO	Internal Oscillator mode, PLL can be enabled or disabled in software, CLKO on RA6, port function on RA7, the internal oscillator block is used to derive both the primary clock source and the postscaled internal clock.
INTOSCPLL	Internal Oscillator mode, PLL can be enabled or disabled in software, port function on RA6 and RA7, the internal oscillator block is used to derive both the primary clock source and the postscaled internal clock.
INTOSCO	Internal Oscillator mode, PLL is always disabled, CLKO on RA6, port function on RA7, the output of the INTOSC postscaler serves as both the postscaled internal clock and the primary clock source.
INTOSC	Internal Oscillator mode, PLL is always disabled, port function on RA6 and RA7, the output of the INTOSC postscaler serves as both the postscaled internal clock and the primary clock source.

3.2.1 OSCILLATOR MODES

A network of MUXes, clock dividers and two PLL circuits have been provided, which can be used to derive various microcontroller frequencies. Figure 3-1 helps in understanding the oscillator structure of the PIC18F47J13 Family of devices.

FIGURE 3-1: PIC18F47J13 FAMILY CLOCK DIAGRAM


PLLDIV<2:0>
PLL Prescaler
÷ 12
÷ 10
÷ 6
÷ 5
÷ 4
÷ 3
÷ 2
÷ 1
000
001
010
011
100
101
110
111
4 MHz
96 MHz PLL(1)
÷ 2
48 MHz
Primary Oscillator
OSC2
OSC1
FOSC<2>
1
0
1
0
0
1
PLLSEL
PLLEN
CFGPLLEN
FOSC<2>
1
0
4x PLL(2)
FOSC<2:1>
Other
00
Primary Clock Source
IDLE
CPU
Peripherals
RA6
÷ 4
CLKO Enabled Modes
Secondary Oscillator
T1OSO
T1OSI
Timer1 Clock(3)
00
01
11
Postscaled Internal Clock
OSCCON<1:0>
OSCCON<6:4>
Internal Oscillator Block
8 MHz
INTRC 31 kHz
8 MHz
INTOSC Postscaler
8 MHz
4 MHz
2 MHz
1 MHz
500 kHz
250 kHz
125 kHz
31 kHz
1
0
111
110
101
100
011
010
001
000
OSCTUNE<7>
WDT, PWRT, FSCM and Two-Speed Start-up

Note 1: The 96 MHz PLL requires a 4 MHz input and it produces a 96 MHz output. The 96 MHz PLL prescaler enables source clocks of 4, 8, 12, 16, 20, 24, 40 or 48 MHz to provide the 4 MHz input.

2: The 4x PLL requires an input clock source between 4 and 12 MHz. When using INTOSC to provide the 4x PLL input, the INTOSC postscaler must be set to either 8 MHz or 4 MHz. Selecting other INTOSC postscaler settings will operate the PLL outside of the specification.

3: Selecting the Timer1 clock or postscaled internal clock will turn off the primary oscillator (unless required by the reference clock of **Section 3.4 “Reference Clock Output”**) and PLL.

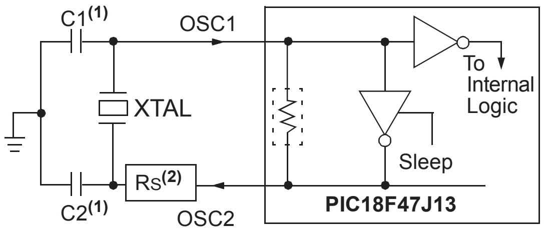
3.2.2 CRYSTAL OSCILLATOR/CERAMIC RESONATORS

In HS and HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 3-2 displays the pin connections.

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

Note: Use of a series resonant crystal may give a frequency out of the crystal manufacturer's specifications.

FIGURE 3-2: 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: RS may be required to avoid overdriving crystals with low drive level specifications.

TABLE 3-2: CAPACITOR SELECTION FOR CERAMIC RESONATORS

Typical Capacitor Values Used:			
Mode	**Freq**	**OSC1**	**OSC2**
HS	8.0 MHz 16.0 MHz	27 pF 22 pF	27 pF 22 pF

Capacitor values are for design guidance only.

These capacitors were tested with the resonators listed below for basic start-up and operation. **These values are not optimized**.

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.

See the notes following Table 3-3 for additional information.

Resonators Used:
4.0 MHz
8.0 MHz
16.0 MHz

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
	16 MHz	18 pF	18 pF

Capacitor values are for design guidance only.

These capacitors were tested with the crystals listed below for basic start-up and operation. **These values are not optimized.**

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.

See the notes following this table for additional information.

Crystals Used:
4 MHz
8 MHz
16 MHz

Note 1: Higher capacitance not only increases the stability of oscillator, but also increases the start-up time.

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

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

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

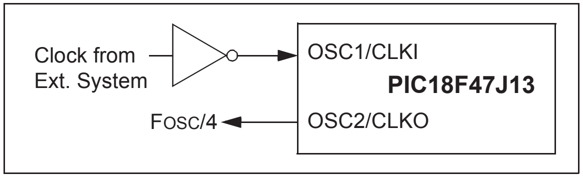
An internal postscaler allows users to select a clock frequency other than that of the crystal or resonator. Frequency division is determined by the CPDIV Configuration bits. Users may select a clock frequency of the oscillator frequency, or 1/2, 1/3 or 1/6 of the frequency.

3.2.3 EXTERNAL CLOCK INPUT

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 (POR) or after an exit from Sleep mode.

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

FIGURE 3-3: EXTERNAL CLOCK INPUT OPERATION (EC AND ECPLL CONFIGURATION)



3.2.4 PLL FREQUENCY MULTIPLIER

A Phase Locked Loop (PLL) circuit provides an option for users who want to use a low-frequency oscillator circuit or 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 the internal oscillator. PIC18F47J13 Family devices include two PLL circuits: a 4x PLL and a 96 MHz PLL.

Either PLL can be enabled in HSPLL, ECPLL, INTOSCPLL and INTOSCPLLO Oscillator modes by setting the PLLEN bit (OSCTUNE<6>) or by clearing the CFGPLLEN bit (CONFIG1L<4>).

The 96 MHz PLL is designed to produce a fixed 96 MHz reference clock from a fixed 4 MHz input. The output can then be divided and used for producing a 48 MHz microcontroller core clock. Because the PLL has a fixed frequency input and output, there are eight prescaling options to match the oscillator input frequency to the PLL. This prescaler allows the PLL to be used with crystals, resonators and external clocks, which are integer multiple frequencies of 4 MHz. For example, a 12 MHz crystal could be used in a prescaler Divide-by-Three mode to drive the 96 MHz PLL.

The 4x PLL is designed to produce variable reference clocks of 16 to 48 MHz from any input clock between 4 and 12 MHz. The PLLSEL (CONFIG3H<2>) Configuration bit is used to select which PLL circuit will be used by the application.

3.2.5 INTERNAL OSCILLATOR BLOCK

The PIC18F47J13 Family devices include an internal oscillator block which generates two different clock signals; either can be used as the microcontroller's clock source. The internal oscillator may eliminate the need for external oscillator circuits on the OSC1 and/or OSC2 pins.

The main output (INTOSC) is an 8 MHz clock source which can be used to directly drive the device clock. It also drives the INTOSC postscaler which can provide a range of clock frequencies from 31 kHz to 8 MHz. Additionally, the INTOSC may be used in conjunction with a PLL to generate clock frequencies up to 48 MHz.

The other clock source is the internal RC oscillator (INTRC) which provides a nominal 31 kHz output. INTRC is enabled if it is selected as the device clock source. It is also enabled automatically when any of the following are enabled:

- Power-up Timer
- Fail-Safe Clock Monitor
- Watchdog Timer
- Two-Speed Start-up

These features are discussed in larger detail in **Section 27.0 "Special Features of the CPU"**.

The clock source frequency (INTOSC direct, INTRC direct or INTOSC postscaler) is selected by configuring the IRCF bits of the OSCCON register (page 39).

3.2.5.1 OSCTUNE Register

The internal oscillator's output has been calibrated at the factory but can be adjusted in the user's application. This is done by writing to the OSCTUNE register (Register 3-1).

When the OSCTUNE register is modified, the INTOSC frequency will begin shifting to the new frequency. The INTOSC clock will typically stabilize within 1 μs. Code execution continues during this shift. There is no indication that the shift has occurred.

The OSCTUNE register also contains the INTSRC bit. The INTSRC bit allows users to select which internal oscillator provides the clock source when the 31 kHz frequency option is selected. This is covered in larger detail in **Section 3.3.1 "Oscillator Control Register"**.

The PLLEN bit, contained in the OSCTUNE register, can be used to enable or disable the internal PLL when running in one of the PLL type oscillator modes (e.g., INTOSCPLL). Oscillator modes that do not contain "PLL" in their name cannot be used with the PLL. In these modes, the PLL is always disabled regardless of the setting of the PLLEN bit.

When configured for one of the PLL enabled modes, setting the PLLEN bit does not immediately switch the device clock to the PLL output. The PLL requires up to electrical parameter, $t_{rc,}$ to start-up and lock, during which time, the device continues to be clocked. Once the PLL output is ready, the microcontroller core will automatically switch to the PLL derived frequency.

3.2.5.2 Internal Oscillator Output Frequency and Drift

The internal oscillator block is calibrated at the factory to produce an INTOSC output frequency of 8.0 MHz. However, this frequency may drift as VDD or temperature changes, which can affect the controller operation in a variety of ways.

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

3.2.5.3 Compensating for INTOSC Drift

It is possible to adjust the INTOSC frequency by modifying the value in the OSCTUNE register. This has no effect on the INTRC clock source frequency.

Tuning the INTOSC source requires knowing when to make the adjustment, in which direction it should be made, and in some cases, how large a change is needed. When using the EUSART, for example, an adjustment may be required when it 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.

It is also possible to verify device clock speed against a reference clock. Two timers may be used: one timer is clocked by the peripheral clock, while the other is clocked by a fixed reference source, such as the Timer1 oscillator. Both timers are cleared, but the timer clocked by the reference generates interrupts. When an interrupt occurs, the internally clocked timer is read and both timers are cleared. If the internally clocked timer value is greater than expected, then the internal oscillator block is running too fast. To adjust for this, decrement the OSCTUNE register.

Finally, an ECCP 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 greater than the calculated time, the internal oscillator block is running too fast; to compensate, decrement the OSCTUNE register. If the measured time is less than the calculated time, the internal oscillator block is running too slow; to compensate, increment the OSCTUNE register.

REGISTER 3-1: OSCTUNE: OSCILLATOR TUNING REGISTER (ACCESS F9Bh)

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(1)	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0
bit 7							bit 0

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

bit 7 **INTSRC:** Internal Oscillator Low-Frequency Source Select bit

`1` = 31.25 kHz device clock derived from 8 MHz INTOSC source (divide-by-256 enabled)
`0` = 31 kHz device clock derived directly from INTRC internal oscillator

bit 6 **PLLEN:** Frequency Multiplier Enable bit(1)

`1` = PLL is enabled
`0` = PLL is disabled

bit 5-0 **TUN<5:0>:** Frequency Tuning bits

`011111` = Maximum frequency
`011110`
•
•
•
`000001`
`000000` = Center frequency; oscillator module is running at the calibrated frequency
`111111`
•
•
•
`100000` = Minimum frequency

Note 1: When the CFGPLLEN Configuration bit is used to enable the PLL, clearing OSCTUNE<6> will not disable the PLL.

3.3 Clock Sources and Oscillator Switching

Like previous PIC18 enhanced devices, the PIC18F47J13 Family includes a feature that allows the device clock source to be switched from the main oscillator to an alternate, low-frequency clock source. PIC18F47J13 Family devices offer two alternate clock sources. When an alternate clock source is enabled, the various power-managed operating modes are available.

Essentially, there are three clock sources for these devices:

- Primary Oscillators
- Secondary Oscillators
- Internal Oscillator Block

The **Primary Oscillators** include the External Crystal and Resonator modes, the External Clock modes and the internal oscillator block. The particular mode is defined by the FOSC<2:0> Configuration bits. The details of these modes are covered earlier in this chapter.

The **Secondary Oscillators** are external sources that are not connected to the OSC1 or OSC2 pins. These sources may continue to operate even after the controller is placed in a power-managed mode.

PIC18F47J13 Family devices offer the Timer1 oscillator as a secondary oscillator. This oscillator, in all power-managed modes, is often the time base for functions such as a Real-Time Clock (RTC). Most often, a 32.768 kHz watch crystal is connected between the RC0/T1OSO/T1CKI/RP11 and RC1/CCP8/T1OSI/RP12 pins. Like the HS Oscillator mode circuits, loading capacitors are also connected from each pin to ground. The Timer1 oscillator is discussed in larger detail in **Section 13.5 “Timer1 Oscillator”**.

In addition to being a primary clock source, the **postscaled internal clock** is available as a power-managed mode clock source. The INTRC source is also used as the clock source for several special features, such as the WDT and Fail-Safe Clock Monitor (FSCM).

3.3.1 OSCILLATOR CONTROL REGISTER

The OSCCON register (Register 3-2) controls several aspects of the device clock’s operation, both in full-power operation and in power-managed modes.

The System Clock Select bits, SCS<1:0>, select the clock source. The available clock sources are the primary clock (defined by the FOSC<2:0> Configuration bits), the secondary clock (Timer1 oscillator) and the postscaled internal clock.The clock source changes immediately, after one or more of the bits is written to, following a brief clock transition interval. The SCS bits are cleared on all forms of Reset.

The Internal Oscillator Frequency Select bits, IRCF<2:0>, select the frequency output provided on the postscaled internal clock line. The choices are the INTRC source, the INTOSC source (8 MHz) or one of the frequencies derived from the INTOSC postscaler (31 kHz to 4 MHz). If the postscaled internal clock is supplying the device clock, changing the states of these bits will have an immediate change on the internal oscillator’s output. On device Resets, the default output frequency of the INTOSC postscaler is set at 4 MHz.

When an output frequency of 31 kHz is selected (IRCF<2:0> = `000`), users may choose the internal oscillator, which acts as the source. This is done with the INTSRC bit in the OSCTUNE register (OSCTUNE<7>). Setting this bit selects INTOSC as a 31.25 kHz clock source by enabling the divide-by-256 output of the INTOSC postscaler. Clearing INTSRC selects INTRC (nominally 31 kHz) as the clock source.

This option allows users to select the tunable and more precise INTOSC as a clock source, while maintaining power savings with a very low clock speed. Regardless of the setting of INTSRC, INTRC always remains the clock source for features such as the WDT and the FSCM.

The OSTS and SOSCRUN 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 (OSCCON2<6>) indicates when the Timer1 oscillator is providing the device clock in secondary clock modes. In power-managed modes, only one of these bits will be set at any time. If none of these bits are set, the INTRC is providing the clock or the internal oscillator block has just started and is not yet stable.

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

The use of the flag and control bits in the OSCCON register is discussed in more detail in **Section 4.0 "Low-Power Modes"**.

Note 1:	The Timer1 crystal driver is enabled by setting the T1OSCEN bit in the Timer1 Control register (T1CON<3>). If the Timer1 oscillator is not enabled, then any attempt to select the Timer1 clock source will be ignored, unless the CONFIG2L register's SOSCSEL<1:0> bits are set to Digital mode.
2:	If Timer1 is driving a crystal, it is recommended that the Timer1 oscillator be operating and stable prior to switching to it as the clock source; otherwise, a very long delay may occur while the Timer1 oscillator starts.

3.3.2 OSCILLATOR TRANSITIONS

PIC18F47J13 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 more detail in **Section 4.1.2 "Entering Power-Managed Modes"**.

REGISTER 3-2: OSCCON: OSCILLATOR CONTROL REGISTER (ACCESS FD3h)

R/W-0	R/W-1	R/W-1	R/W-0	R-1[1]	U-1	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS	—	SCS1	SCS0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit	
-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 Idle mode on `SLEEP` instruction `0` = Device enters Sleep mode on `SLEEP` instruction
bit 6-4	**IRCF<2:0>:** Internal Oscillator Frequency Select bits When using INTOSC to drive the 4x PLL, select 8 MHz or 4 MHz only to avoid operating the 4x PLL outside of specification. `111` = 8 MHz (INTOSC drives clock directly) `110` = 4 MHz[2] `101` = 2 MHz `100` = 1 MHz `011` = 500 kHz `010` = 250 kHz `001` = 125 kHz `000` = 31 kHz (from either INTOSC/256 or INTRC directly)[3]
bit 3	**OSTS:** Oscillator Start-up Time-out Status bit[1] `1` = Oscillator Start-up Timer time-out has expired; primary oscillator is running `0` = Oscillator Start-up Timer time-out is running; primary oscillator is not ready
bit 2	**Unimplemented:** Read as '`1`'
bit 1-0	**SCS<1:0>:** System Clock Select bits `11` = Postscaled internal clock (INTRC/INTOSC derived) `10` = Reserved `01` = Timer1 oscillator `00` = Primary clock source (INTOSC postscaler output when FOSC<2:0> = `001` or `000`) `00` = Primary clock source (CPU divider output for other values of FOSC<2:0>)

Note 1: Reset value is '`0`' when Two-Speed Start-up is enabled and '`1`' if disabled.
2: Default output frequency of INTOSC on Reset (4 MHz).
3: Source selected by the INTSRC bit (OSCTUNE<7>).

REGISTER 3-3: OSCCON2: OSCILLATOR CONTROL REGISTER 2 (ACCESS F87h)

U-0	R-0(2)	U-0	R/W-1	R/W-0(2)	R/W-1	U-0	U-0
—	SOSCRUN	—	SOSCDRV	SOSCGO(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 **Unimplemented:** Read as ‘0’

bit 6 **SOSCRUN:** SOSC Run Status bit
1 = System clock comes from secondary SOSC
0 = System clock comes from an oscillator other than SOSC

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **SOSCDRV:** SOSC Drive Control bit
1 = T1OSC/SOSC oscillator drive circuit is selected by Configuration bits, CONFIG2L<4:3>
0 = Low-power T1OSC/SOSC circuit is selected

bit 3 **SOSCGO:** Oscillator Start Control bit[3]
1 = Turns on the oscillator, even if no peripherals are requesting it
0 = Oscillator is shut off unless peripherals are requesting it

bit 2 **Reserved:** Maintain as ‘1’

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

Note 1: Reset value is ‘0’ when Two-Speed Start-up is enabled and ‘1’ if disabled.

2: Default output frequency of INTOSC on Reset (4 MHz).

3: When the SOSC is selected to run from a digital clock input, rather than an external crystal, this bit has no effect.

3.4 Reference Clock Output

In addition to the peripheral clock/4 output in certain oscillator modes, the device clock in the PIC18F47J13 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 (RB2) 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 is 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 RB2 when the device is in Sleep mode.

To use the reference clock output in Sleep mode, both the ROSSLP and ROSEL bits must be set. The device clock must also be configured for an EC or HS mode; otherwise, the oscillator on OSC1 and OSC2 will be powered down when the device enters Sleep mode. Clearing the ROSEL bit allows the reference output frequency to change as the system clock changes during any clock switches.

REGISTER 3-4: REFOCON: REFERENCE OSCILLATOR CONTROL REGISTER (BANKED F3Dh)

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	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 is enabled on REFO pin
`0` = Reference oscillator 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` = Primary oscillator crystal/resonator is used as the base clock(1)
`0` = System clock (FOSC) is used as the base clock; the 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: The crystal oscillator must be enabled using the FOSC<2:0> bits; the crystal maintains the operation in Sleep mode.

3.5 Effects of Power-Managed Modes on Various Clock Sources

When the 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.

In secondary clock modes (SEC_RUN and SEC_IDLE), the Timer1 oscillator is operating and providing the device clock. The Timer1 oscillator may also run in all power-managed modes if required to clock Timer1, Timer3 or Timer5.

In internal oscillator modes (RC_RUN and RC_IDLE), the internal oscillator block provides the device clock source. The 31 kHz INTRC output can be used directly to provide the clock and may be enabled to support various special features regardless of the power-managed mode (see **Section 27.2 "Watchdog Timer (WDT)"**, **Section 27.4 "Two-Speed Start-up"** and **Section 27.5 "Fail-Safe Clock Monitor"** for more information on WDT, FSCM and Two-Speed Start-up). The INTOSC output at 8 MHz may be used directly to clock the device or may be divided down by the postscaler. The INTOSC output is disabled if the clock is provided directly from the INTRC output.

If Sleep mode is selected, all clock sources which are no longer required are stopped. Since all the transistor switching currents have been stopped, Sleep mode achieves the lowest current consumption of the device (only leakage currents) outside of Deep Sleep.

Enabling any on-chip feature that will operate during Sleep mode increases the current consumed during Sleep mode. The INTRC is required to support WDT operation. The Timer1 oscillator may be operating to support an RTC. Other features may be operating that do not require a device clock source (i.e., MSSP slave, PMP, INTx pins, etc.). Peripherals that may add significant current consumption are listed in **Section 30.2 "DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial)"**.

3.6 Power-up Delays

Power-up delays are controlled by two timers so that no external Reset circuitry is required for most applications. The delays ensure that the device is kept in Reset until the device power supply is stable under normal circumstances and the primary clock is operating and stable. For additional information on power-up delays, see **Section 5.6 "Power-up Timer (PWRT)"**.

The first timer is the Power-up Timer (PWRT), which provides a fixed delay on power-up (parameter 33, Table 30-14).

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

There is a delay of interval, T_{CSD} (parameter 38, Table 30-14), following POR, while the controller becomes ready to execute instructions. This delay runs concurrently with any other delays. This may be the only delay that occurs when any of the internal oscillator or EC modes are used as the primary clock source.

4.0 LOW-POWER MODES

The PIC18F47J13 Family devices can manage power consumption through clocking to the CPU and the peripherals. In general, reducing the clock frequency and number of circuits being clocked reduces power consumption.

For managing power in an application, the primary modes of operation are:

- Run Mode
- Idle Mode
- Sleep Mode
- Deep Sleep Mode

Additionally, there is an Ultra Low-Power Wake-up (ULPWU) mode for generating an interrupt-on-change on RA0.

These modes define which portions of the device are clocked and at what speed.

- The Run and Idle modes can use any of the three available clock sources (primary, secondary or internal oscillator blocks).
- The Sleep mode does not use a clock source.

The ULPWU mode on RA0 allows a slow falling voltage to generate an interrupt-on-change on RA0 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, such as clock switching, ULPWU and Sleep mode. In addition, the PIC18F47J13 Family devices have added a new power-managed Deep Sleep mode.

4.1 Selecting Power-Managed Modes

Selecting a power-managed mode requires these decisions:

- Will the CPU be clocked?
- If so, which clock source will be used?

The IDLEN bit (OSCCON<7>) controls CPU clocking and the SCS<1:0> bits (OSCCON<1:0>) select the clock source. The individual modes, bit settings, clock sources and affected modules are summarized in Table 4-1.

4.1.1 CLOCK SOURCES

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

- Primary clock source – Defined by the FOSC<2:0> Configuration bits
- Timer1 clock – Provided by the secondary oscillator
- Postscaled internal clock – Derived from the internal oscillator block

4.1.2 ENTERING POWER-MANAGED MODES

Switching from one clock source to another begins by loading the OSCCON register. The SCS<1:0> bits select the clock source.

Changing these bits causes an immediate switch to the new clock source, assuming that it is running. The switch also may be subject to clock transition delays. These delays are discussed in **Section 4.1.3 "Clock Transitions and Status Indicators"** and subsequent sections.

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

Depending on the current mode and the mode being switched to, a change to a power-managed mode does not always require setting all of these bits. Many transitions may be done by changing the oscillator select bits, the IDLEN bit or the DSEN bit prior to issuing a `SLEEP` instruction.

If the IDLEN and DSEN bits are already configured correctly, it may only be necessary to perform a `SLEEP` instruction to switch to the desired mode.

TABLE 4-1: LOW-POWER MODES

Mode	DSCONH<7>	OSCCON<7,1:0>		Module Clocking		Available Clock and Oscillator Source
	DSEN[1]	IDLEN[1]	SCS<1:0>	CPU	Peripherals	
Sleep	`0`	`0`	N/A	Off	Off	Timer1 oscillator and/or RTCC may optionally be enabled
Deep Sleep[3]	`1`	`0`	N/A	Powered off[2]	Powered off	RTCC can run uninterrupted using the Timer1 or internal low-power RC oscillator
PRI_RUN	`0`	N/A	`00`	Clocked	Clocked	The normal, full-power execution mode; primary clock source (defined by FOSC<2:0>)
SEC_RUN	`0`	N/A	`01`	Clocked	Clocked	Secondary – Timer1 oscillator
RC_RUN	`0`	N/A	`11`	Clocked	Clocked	Postscaled internal clock
PRI_IDLE	`0`	`1`	`00`	Off	Clocked	Primary clock source (defined by FOSC<2:0>)
SEC_IDLE	`0`	`1`	`01`	Off	Clocked	Secondary – Timer1 oscillator
RC_IDLE	`0`	`1`	`11`	Off	Clocked	Postscaled internal clock

Note 1: IDLEN and DSEN reflect their values when the `SLEEP` instruction is executed.

2: Deep Sleep turns off the internal core voltage regulator to power down core logic. See **Section 4.6 "Deep Sleep Mode"** for more information.

3: Deep Sleep mode is only available on "F" devices, not "LF" devices.

4.1.3 CLOCK TRANSITIONS AND STATUS INDICATORS

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

Two bits indicate the current clock source and its status: OSTS (OSCCON<3>) and SOSCRUN (OSCCON2<6>). In general, only one of these bits will be set in a given power-managed mode. When the OSTS bit is set, the primary clock would be providing the device clock. When the SOSCRUN bit is set, the Timer1 oscillator would be providing the clock. If neither of these bits is set, INTRC would be clocking the device.

Note: Executing a `SLEEP` instruction does not necessarily place the device into Sleep mode. It acts as the trigger to place the controller into either the Sleep or Deep 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 and DSEN bits at the time the instruction is executed. If another `SLEEP` instruction is executed, the device will enter the power-managed mode specified by IDLEN and DSEN at that time. If IDLEN or DSEN have changed, the device will enter the new power-managed mode specified by the new setting.

4.2 Run Modes

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

4.2.1 PRI_RUN MODE

The PRI_RUN mode is the normal, full-power execution mode of the microcontroller. This is also the default mode upon a device Reset unless Two-Speed Start-up is enabled (see **Section 27.4 "Two-Speed Start-up"** for details). In this mode, the OSTS bit is set (see **Section 3.3.1 "Oscillator Control Register"**).

4.2.2 SEC_RUN MODE

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

SEC_RUN mode is entered by setting the SCS<1:0> bits to '`01`'. The device clock source is switched to the Timer1 oscillator (see Figure 4-1), the primary oscillator is shut down, the SOSCRUN bit (OSCCON2<6>) is set and the OSTS bit is cleared.

Note:	The Timer1 oscillator should already be running prior to entering SEC_RUN mode. If the T1OSCEN bit is not set when the SCS<1:0> bits are set to '`01`', entry to SEC_RUN mode will not occur. If the Timer1 oscillator is enabled, but not yet running, device clocks will be delayed until the oscillator has started. In such situations, initial oscillator operation is far from stable and unpredictable operation may result.

On transitions from SEC_RUN mode to PRI_RUN mode, the peripherals and CPU continue to be clocked from the Timer1 oscillator while the primary clock is started. When the primary clock becomes ready, a clock switch back to the primary clock occurs (see Figure 4-2). When the clock switch is complete, the SOSCRUN bit is cleared, the OSTS bit is set and the primary clock would be providing the clock. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run.

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

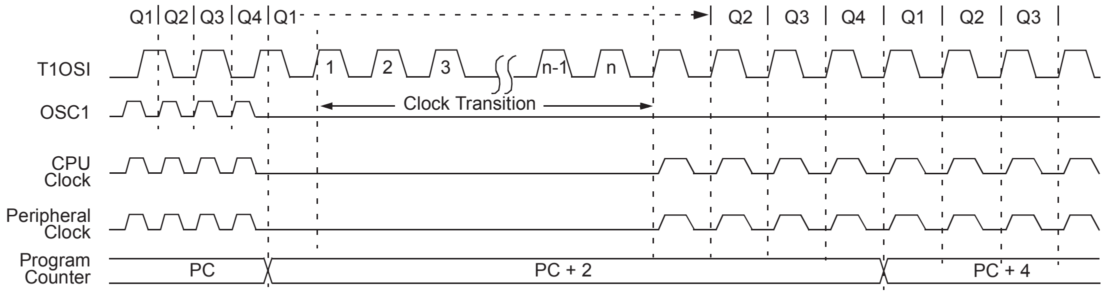
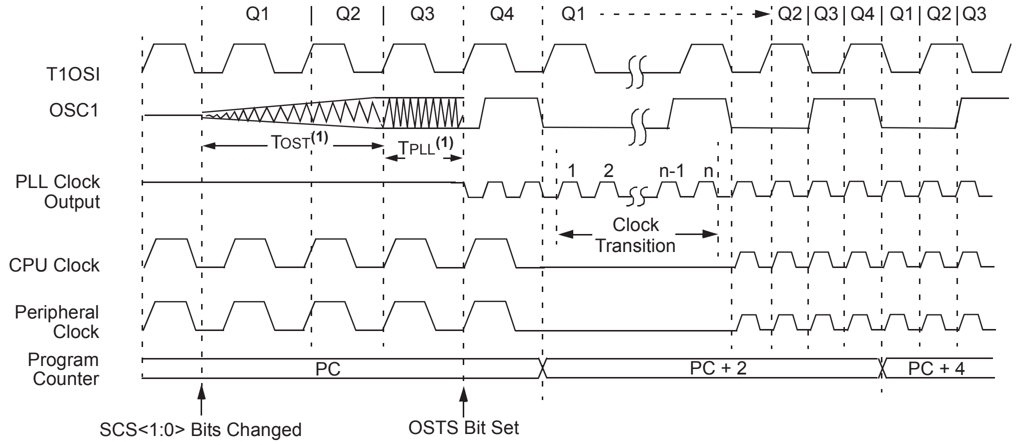

Q1
Q2
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
Q3
T1OSI
1
2
3
n-1
n
OSC1
Clock Transition
CPU Clock
Peripheral Clock
Program Counter
PC
PC + 2
PC + 4

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


Q1
Q2
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
Q3
T1OSI
OSC1
TOST(1)
TPLL(1)
PLL Clock Output
1
2
n-1
n
Clock Transition
CPU Clock
Peripheral Clock
Program Counter
PC
PC + 2
PC + 4
SCS<1:0> Bits Changed
OSTS Bit Set

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

4.2.3 RC_RUN MODE

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

This mode is entered by setting the SCS<1:0> bits (OSCCON<1:0>) to `11`. When the clock source is switched to the internal oscillator block (see Figure 4-3), the primary oscillator is shut down and the OSTS bit is cleared.

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

FIGURE 4-3: TRANSITION TIMING TO RC_RUN MODE

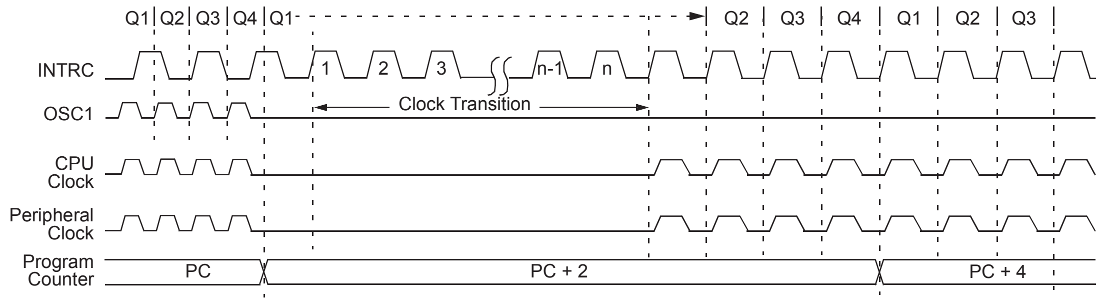
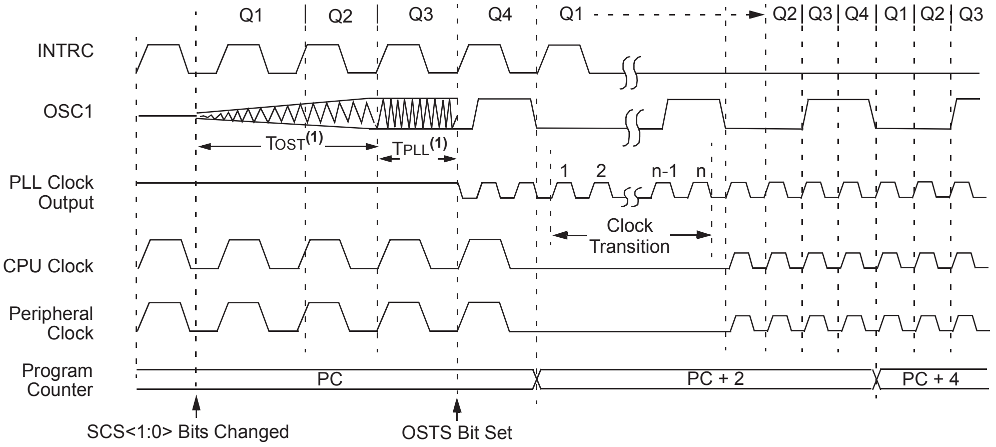

Q1
Q2
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
Q3
INTRC
OSC1
CPU Clock
Peripheral Clock
Program Counter
1
2
3
n-1
n
Clock Transition
PC
PC + 2
PC + 4

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


Q1
Q2
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
Q3
INTRC
OSC1
PLL Clock Output
CPU Clock
Peripheral Clock
Program Counter
TOST(1)
TPLL(1)
1
2
n-1
n
Clock Transition
PC
PC + 2
PC + 4
SCS<1:0> Bits Changed
OSTS Bit Set

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

4.3 Sleep Mode

The power-managed Sleep mode is identical to the legacy Sleep mode offered in all other PIC devices. It is entered by clearing the IDLEN bit (the default state on device Reset) and executing the `SLEEP` instruction. This shuts down the selected oscillator (Figure 4-5). All clock source status bits are cleared.

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

When a wake event occurs in Sleep mode (by interrupt, Reset or WDT time-out), the device will not be clocked until the clock source selected by the SCS<1:0> bits becomes ready (see Figure 4-6), or it will be clocked from the internal oscillator if either the Two-Speed Start-up or the FSCM is enabled (see **Section 27.0 "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.

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

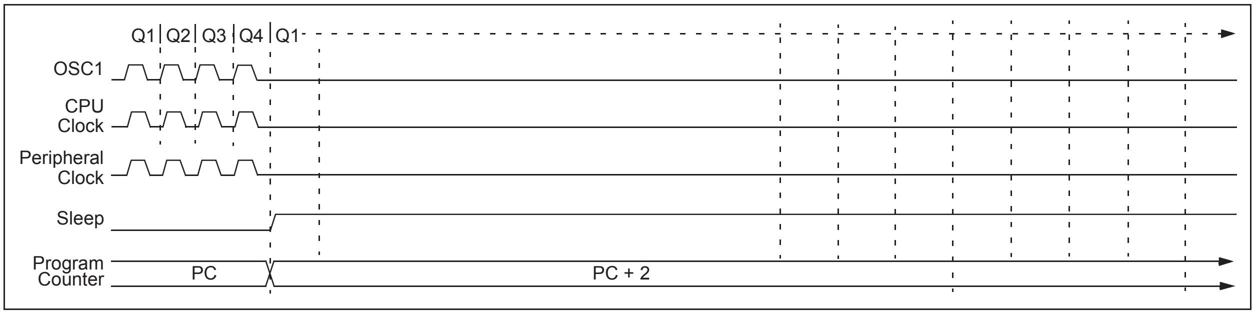
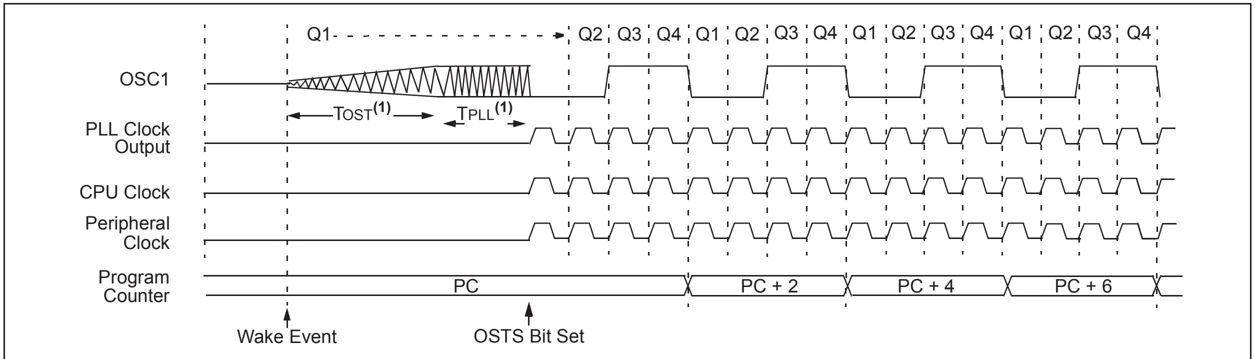

Q1
Q2
Q3
Q4
Q1
OSC1
CPU Clock
Peripheral Clock
Sleep
Program Counter
PC
PC + 2

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


Q1
Q2
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
Q3
Q4
OSC1
TOST(1)
TPLL(1)
PLL Clock Output
CPU Clock
Peripheral Clock
Program Counter
PC
PC + 2
PC + 4
PC + 6
Wake Event
OSTS Bit Set

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

4.4 Idle Modes

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

If the IDLEN bit is set to '`1`' when a `SLEEP` instruction is executed, the peripherals will be clocked from the clock source selected using the SCS<1:0> bits; however, the CPU will not be clocked. The clock source status bits are not affected. Setting IDLEN and executing a `SLEEP` instruction provides a quick method of switching from a given Run mode to its corresponding Idle mode.

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

Since the CPU is not executing instructions, the only exits from any of the Idle modes are by interrupt, WDT time-out or a Reset. When a wake event occurs, CPU execution is delayed by an interval of TCSD (parameter 38, Table 30-14) 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 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.

4.4.1 PRI_IDLE MODE

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

PRI_IDLE mode is entered from PRI_RUN mode by setting the IDLEN bit and executing a `SLEEP` instruction. If the device is in another Run mode, set IDLEN first, then set the SCS bits to '`00`' and execute `SLEEP`. Although the CPU is disabled, the peripherals continue to be clocked from the primary clock source specified by the FOSC<1:0> Configuration bits. The OSTS bit remains set (see Figure 4-7).

When a wake event occurs, the CPU is clocked from the primary clock source. A delay of interval, TCSD, is required between the wake event and when code execution starts. This is required to allow the CPU to become ready to execute instructions. After the wake-up, the OSTS bit remains set. The IDLEN and 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 Timer1 oscillator. This mode is entered from SEC_RUN by setting the IDLEN bit and executing a `SLEEP` instruction. If the device is in another Run mode, set IDLEN first, then set SCS<1:0> to '`01`' and execute `SLEEP`. When the clock source is switched to the Timer1 oscillator, the primary oscillator is shut down, the OSTS bit is cleared and the SOSCRUN bit is set.

When a wake event occurs, the peripherals continue to be clocked from the Timer1 oscillator. After an interval of TCSD following the wake event, the CPU begins executing code being clocked by the Timer1 oscillator. The IDLEN and SCS bits are not affected by the wake-up; the Timer1 oscillator continues to run (see Figure 4-8).

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

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

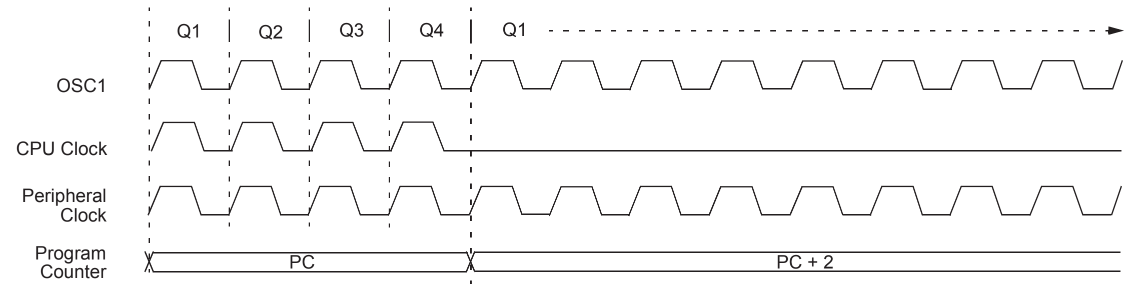
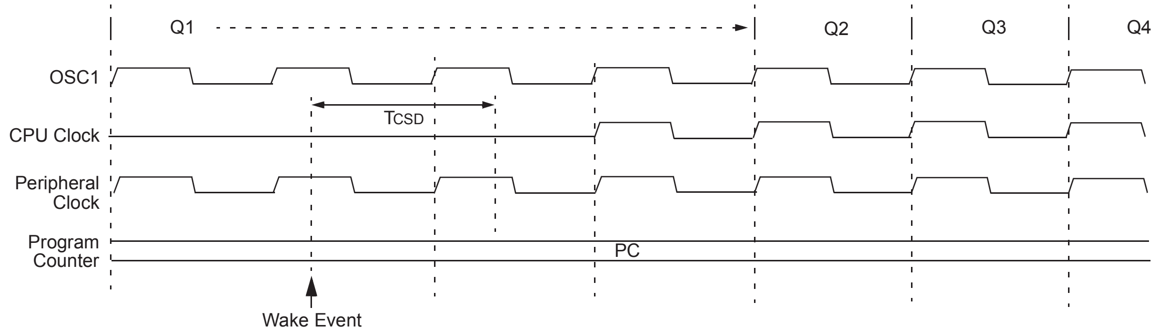

Q1
Q2
Q3
Q4
Q1
OSC1
CPU Clock
Peripheral Clock
Program Counter
PC
PC + 2

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


Q1
Q2
Q3
Q4
OSC1
TCSD
CPU Clock
Peripheral Clock
Program Counter
PC
Wake Event

4.4.3 RC_IDLE MODE

In RC_IDLE mode, the CPU is disabled but the peripherals continue to be clocked from the internal oscillator block. This mode allows for controllable power conservation during Idle periods.

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

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

4.5 Exiting Idle and Sleep Modes

An exit from Sleep mode, or any of the Idle modes, is triggered by an interrupt, a Reset or a WDT time-out. This section discusses the triggers that cause exits from power-managed modes. The clocking subsystem actions are discussed in each of the power-managed modes sections (see **Section 4.2 “Run Modes”**, **Section 4.3 “Sleep Mode”** and **Section 4.4 “Idle Modes”**).

4.5.1 EXIT BY INTERRUPT

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

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

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

4.5.2 EXIT BY WDT TIME-OUT

A WDT time-out will cause different actions depending on which power-managed mode the device is, 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 27.2 “Watchdog Timer (WDT)”**).

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

- Executing a `SLEEP` or `CLRWDT` instruction
- The loss of a currently selected clock source (if the FSCM is enabled)

4.5.3 EXIT BY RESET

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

4.5.4 EXIT WITHOUT AN OSCILLATOR START-UP DELAY

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

- PRI_IDLE mode (where the primary clock source is not stopped) and the primary clock source is the EC mode
- PRI_IDLE mode and the primary clock source is the ECPLL mode

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

4.6 Deep Sleep Mode

Deep Sleep mode brings the device into its lowest power consumption state without requiring the use of external switches to remove power from the device. During Deep Sleep, the on-chip V_{DDCORE} voltage regulator is powered down, effectively disconnecting power to the core logic of the microcontroller.

Note: Since Deep Sleep mode powers down the microcontroller by turning off the on-chip V_{DDCORE} voltage regulator, Deep Sleep capability is available only on PIC18FXXJ members in the device family. The on-chip voltage regulator is not available in PIC18LFXXJ members of the device family, and therefore, they do not support Deep Sleep.

On devices that support it, the Deep Sleep mode is entered by:

- Setting the REGSLP (WDTCON<7>) bit
- Clearing the IDLEN bit
- Clearing the GIE bit
- Setting the DSEN bit (DSCONH<7>)
- Executing one single-cycle `NOP` instruction
- Executing the `SLEEP` instruction immediately after setting DSEN and the single `NOP` (no interrupts in between)

In order to minimize the possibility of inadvertently entering Deep Sleep, the DSEN bit is cleared in hardware two instruction cycles after having been set. Therefore, in order to enter Deep Sleep, the `SLEEP` instruction must be executed in the immediate instruction cycle after setting DSEN. If DSEN is not set when `SLEEP` is executed, the device will enter conventional Sleep mode instead.

During Deep Sleep, the core logic circuitry of the microcontroller is powered down to reduce leakage current. Therefore, most peripherals and functions of the microcontroller become unavailable during Deep Sleep. However, a few specific peripherals and functions are powered directly from the V_{DD} supply rail of the microcontroller, and therefore, can continue to function in Deep Sleep.

Entering Deep Sleep mode clears the DSWAKEL register. However, if the Real-Time Clock and Calendar (RTCC) is enabled prior to entering Deep Sleep, it will continue to operate uninterrupted.

The device has a dedicated Brown-out Reset (DSBOR) and Watchdog Timer Reset (DSWDT) for monitoring voltage and time-out events in Deep Sleep. The DSBOR and DSWDT are independent of the standard BOR and WDT used with other power-managed modes (Run, Idle and Sleep).

When a wake event occurs in Deep Sleep mode (by $\overline{\text{MCLR}}$ Reset, RTCC alarm, INT0 interrupt, ULPWU or DSWDT), the device will exit Deep Sleep mode and perform a Power-on Reset (POR). When the device is released from Reset, code execution will resume at the device's Reset vector.

4.6.1 PREPARING FOR DEEP SLEEP

Because V_{DDCORE} could fall below the SRAM retention voltage while in Deep Sleep mode, SRAM data could be lost in Deep Sleep. Exiting Deep Sleep mode causes a POR; as a result, most Special Function Registers will reset to their default POR values.

Applications needing to save a small amount of data throughout a Deep Sleep cycle can save the data to the general purpose DSGPR0 and DSGPR1 registers. The contents of these registers are preserved while the device is in Deep Sleep, and will remain valid throughout an entire Deep Sleep entry and wake-up sequence.

4.6.2 I/O PINS DURING DEEP SLEEP

During Deep Sleep, the general purpose I/O pins will retain their previous states.

Pins that are configured as inputs (TRISx bit set) prior to entry into Deep Sleep will remain high-impedance during Deep Sleep.

Pins that are configured as outputs (TRISx bit clear) prior to entry into Deep Sleep will remain as output pins during Deep Sleep. While in this mode, they will drive the output level determined by their corresponding LAT bit at the time of entry into Deep Sleep.

When the device wakes back up, the I/O pin behavior depends on the type of wake up source.

If the device wakes back up by an RTCC alarm, INT0 interrupt, DSWDT or ULPWU event, all I/O pins will continue to maintain their previous states, even after the device has finished the POR sequence and is executing application code again. Pins configured as inputs during Deep Sleep will remain high-impedance, and pins configured as outputs will continue to drive their previous value.

After waking up, the TRIS and LAT registers will be reset, but the I/O pins will still maintain their previous states. If firmware modifies the TRIS and LAT values for the I/O pins, they will not immediately go to the newly configured states. Once the firmware clears the RELEASE bit (DSCONL<0>), the I/O pins will be "released". This causes the I/O pins to take the states configured by their respective TRIS and LAT bit values.

If the Deep Sleep BOR (DSBOR) circuit is enabled, and VDD drops below the DSBOR and VDD rail POR thresholds, the I/O pins will be immediately released similar to clearing the RELEASE bit. All previous state information will be lost, including the general purpose DSGPR0 and DSGPR1 contents. See **Section 4.6.5 "Deep Sleep Brown-out Reset (DSBOR)"** for additional details regarding this scenario.

If a $\overline{\text{MCLR}}$ Reset event occurs during Deep Sleep, the I/O pins will also be released automatically, but in this case, the DSGPR0 and DSGPR1 contents will remain valid.

In all other Deep Sleep wake-up cases, application firmware needs to clear the RELEASE bit in order to reconfigure the I/O pins.

4.6.3 DEEP SLEEP WAKE-UP SOURCES

The device can be awakened from Deep Sleep mode by a $\overline{\text{MCLR}}$, POR, RTCC, INT0 I/O pin interrupt, DSWDT or ULPWU event. After waking, the device performs a POR. When the device is released from Reset, code execution will begin at the device's Reset vector.

The software can determine if the wake-up was caused from an exit from Deep Sleep mode by reading the DS bit (WDTCON<3>). If this bit is set, the POR was caused by a Deep Sleep exit. The DS bit must be manually cleared by the software.

The software can determine the wake event source by reading the DSWAKEH and DSWAKEL registers. When the application firmware is done using the DSWAKEH and DSWAKEL status registers, individual bits do not need to be manually cleared before entering Deep Sleep again. When entering Deep Sleep mode, these registers are automatically cleared.

4.6.3.1 Wake-up Event Considerations

Deep Sleep wake-up events are only monitored while the processor is fully in Deep Sleep mode. If a wake-up event occurs before Deep Sleep mode is entered, the event status will not be reflected in the DSWAKE registers. If the wake-up source asserts prior to entering Deep Sleep, the CPU will either go to the interrupt vector (if the wake source has an interrupt bit and the interrupt is fully enabled) or will abort the Deep Sleep entry sequence by executing past the `SLEEP` instruction, if the interrupt was not enabled. In this case, a wake-up event handler should be placed after the `SLEEP` instruction to process the event and re-attempt entry into Deep Sleep if desired.

When the device is in Deep Sleep with more than one wake-up source simultaneously enabled, only the first wake-up source to assert will be detected and logged in the DSWAKEH/DSWAKEL Status registers.

4.6.4 DEEP SLEEP WATCHDOG TIMER (DSWDT)

Deep Sleep has its own dedicated WDT (DSWDT) with a postscaler for time-outs of 2.1 ms to 25.7 days, configurable through the bits, DSWDTPS<3:0>.

The DSWDT can be clocked from either the INTRC or the T1OSC/T1CKI input. If the T1OSC/T1CKI source will be used with a crystal, the T1OSCEN bit in the T1CON register needs to be set prior to entering Deep Sleep. The reference clock source is configured through the DSWDTOSC bit.

DSWDT is enabled through the DSWDTEN bit. Entering Deep Sleep mode automatically clears the DSWDT. See **Section 27.0 "Special Features of the CPU"** for more information.

4.6.5 DEEP SLEEP BROWN-OUT RESET (DSBOR)

The Deep Sleep module contains a dedicated Deep Sleep BOR (DSBOR) circuit. This circuit may be optionally enabled through the DSBOREN Configuration bit.

The DSBOR circuit monitors the VDD supply rail voltage. The behavior of the DSBOR circuit is described in **Section 5.4 “Brown-out Reset (BOR)”**.

4.6.6 RTCC PERIPHERAL AND DEEP SLEEP

The RTCC can operate uninterrupted during Deep Sleep mode. It can wake the device from Deep Sleep by configuring an alarm.

The RTCC clock source is configured with the RTCOSC bit (CONFIG3L<1>). The available reference clock sources are the INTRC and T1OSC/T1CKI. If the INTRC is used, the RTCC accuracy will directly depend on the INTRC tolerance. For more information on configuring the RTCC peripheral, see **Section 17.0 “Real-Time Clock and Calendar (RTCC)”**.

4.6.7 TYPICAL DEEP SLEEP SEQUENCE

This section gives the typical sequence for using the Deep Sleep mode. Optional steps are indicated and additional information is given in notes at the end of the procedure.

1. Enable DSWDT (optional).(1)
2. Configure the DSWDT clock source (optional).(2)
3. Enable DSBOR (optional).(1)
4. Enable RTCC (optional).(3)
5. Configure the RTCC peripheral (optional).(3)
6. Configure the ULPWU peripheral (optional).(4)
7. Enable the INT0 Interrupt (optional).
8. Context save SRAM data by writing to the DSGPR0 and DSGPR1 registers (optional).
9. Set the REGSLP bit (WDTCON<7>) and clear the IDLEN bit (OSCCON<7>).
10. If using an RTCC alarm for wake-up, wait until the RTCSYNC bit (RTCCFG<4>) is clear.
11. Enter Deep Sleep mode by setting the DSEN bit (DSCONH<7>), execute a `NOP` and issue a `SLEEP` instruction. The `SLEEP` instruction must be executed within 2 instruction cycles of setting the DSEN bit.
12. Once a wake-up event occurs, the device will perform a POR Reset sequence. Code execution resumes at the device’s Reset vector.
13. Determine if the device exited Deep Sleep by reading the Deep Sleep bit, DS (WDTCON<3>). This bit will be set if there was an exit from Deep Sleep mode.
14. Clear the Deep Sleep bit, DS (WDTCON<3>).
15. Determine the wake-up source by reading the DSWAKEH and DSWAKEL registers.
16. Determine if a DSBOR event occurred during Deep Sleep mode by reading the DSBOR bit (DSCONL<1>).
17. Read the DSGPR0 and DSGPR1 context save registers (optional).
18. Clear the RELEASE bit (DSCONL<0>).

Note 1: DSWDT and DSBOR are enabled through the devices’ Configuration bits. For more information, see **Section 27.1 “Configuration Bits”**.

2: The DSWDT and RTCC clock sources are selected through the devices’ Configuration bits. For more information, see **Section 27.1 “Configuration Bits”**.

3: For more information, see **Section 17.0 “Real-Time Clock and Calendar (RTCC)”**.

4: For more information on configuring this peripheral, see **Section 4.7 “Ultra Low-Power Wake-up”**.

4.6.8 DEEP SLEEP FAULT DETECTION

If during Deep Sleep, the device is subjected to unusual operating conditions, such as an Electrostatic Discharge (ESD) event, it is possible that internal circuit states used by the Deep Sleep module could become corrupted. If this were to happen, the device may exhibit unexpected behavior, such as a failure to wake back up.

In order to prevent this type of scenario from occurring, the Deep Sleep module includes automatic self-monitoring capability. During Deep Sleep, critical internal nodes are continuously monitored in order to detect possible Fault conditions (which would not ordinarily occur). If a Fault condition is detected, the circuitry will set the DSFLT status bit (DSWAKEL<7>) and automatically wake the microcontroller from Deep Sleep, causing a POR Reset.

During Deep Sleep, the Fault detection circuitry is always enabled and does not require any specific configuration prior to entering Deep Sleep.

4.6.9 DEEP SLEEP MODE REGISTERS

Deep Sleep mode registers are provided in Register 4-1 through Register 4-6.

REGISTER 4-1: DSCONH: DEEP SLEEP CONTROL HIGH BYTE REGISTER (BANKED F4Dh)

R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
DSEN[(1)]	—	—	—	—	r	DSULPEN	RTCWDIS
bit 7							bit 0

Legend:	r = Reserved 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 **DSEN:** Deep Sleep Enable bit[(1)]

`1` = Deep Sleep mode is entered on a `SLEEP` command
`0` = Sleep mode is entered on a `SLEEP` command

bit 6-3 **Unimplemented:** Read as ‘`0`’

bit 2 **Reserved:** Maintain as ‘`0`’

bit 1 **DSULPEN:** Ultra Low-Power Wake-up Module Enable bit

`1` = ULPWU module is enabled in Deep Sleep
`0` = ULPWU module is disabled in Deep Sleep

bit 0 **RTCWDIS:** RTCC Wake-up Disable bit

`1` = Wake-up from RTCC is disabled
`0` = Wake-up from RTCC is enabled

Note 1: In order to enter Deep Sleep, Sleep must be executed within 2 instruction cycles after setting DSEN.

REGISTER 4-2: DSCONL: DEEP SLEEP LOW BYTE CONTROL REGISTER (BANKED F4Ch)

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0[(1)]	R/W-0[(1)]
—	—	—	—	—	ULPWDIS	DSBOR	RELEASE
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 **ULPWDIS:** Ultra Low-Power Wake-up Disable bit

`1` = ULPWU wake-up source is disabled
`0` = ULPWU wake-up source is enabled (must also set DSULPEN = `1`)

bit 1 **DSBOR:** Deep Sleep BOR Event Status bit

`1` = DSBOREN was enabled and V_{DD} dropped below the DSBOR arming voltage during Deep Sleep, but did not fall below V_{DSBOR}
`0` = DSBOREN was disabled or V_{DD} did not drop below the DSBOR arming voltage during Deep Sleep

bit 0 **RELEASE:** I/O Pin State Release bit

Upon waking from Deep Sleep, the I/O pins maintain their previous states. Clearing this bit will release the I/O pins and allow their respective TRIS and LAT bits to control their states.

Note 1: This is the value when V_{DD} is initially applied.

REGISTER 4-3: DSGPR0: DEEP SLEEP PERSISTENT GENERAL PURPOSE REGISTER 0 (BANKED F4Eh)

R/W-xxxx(1)	
Deep Sleep Persistent General Purpose bits	
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 Deep Sleep Persistent General Purpose bits
Contents are retained even in Deep Sleep mode.

Note 1: All register bits are maintained unless VDDCORE drops below the normal BOR threshold outside of Deep Sleep, or the device is in Deep Sleep and the dedicated DSBOR is enabled and VDD drops below the DSBOR threshold, or DSBOR is enabled or disabled, but VDD is hard cycled to near VSS.

REGISTER 4-4: DSGPR1: DEEP SLEEP PERSISTENT GENERAL PURPOSE REGISTER 1 (BANKED F4Fh)

R/W-xxxx(1)	
Deep Sleep Persistent General Purpose bits	
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 Deep Sleep Persistent General Purpose bits
Contents are retained even in Deep Sleep mode.

Note 1: All register bits are maintained unless VDDCORE drops below the normal BOR threshold outside of Deep Sleep, or the device is in Deep Sleep and the dedicated DSBOR is enabled and VDD drops below the DSBOR threshold, or DSBOR is enabled or disabled, but VDD is hard cycled to near VSS.

REGISTER 4-5: DSWAKEH: DEEP SLEEP WAKE HIGH BYTE REGISTER (BANKED F4Bh)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	DSINT0
bit 7							bit 0

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

bit 7-1 **Unimplemented:** Read as ‘`0`’

bit 0 **DSINT0:** Interrupt-on-Change bit
`1` = Interrupt-on-change was asserted during Deep Sleep
`0` = Interrupt-on-change was not asserted during Deep Sleep

REGISTER 4-6: DSWAKEL: DEEP SLEEP WAKE LOW BYTE REGISTER (BANKED F4Ah)

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-1
DSFLT	—	DSULP	DSWDT	DSRTC	DSMCLR	—	DSPOR
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	**DSFLT:** Deep Sleep Fault Detected bit 1 = A Deep Sleep Fault was detected during Deep Sleep 0 = A Deep Sleep Fault was not detected during Deep Sleep
bit 6	**Unimplemented:** Read as ‘0’
bit 5	**DSULP:** Ultra Low-Power Wake-up Status bit 1 = An ultra low-power wake-up event occurred during Deep Sleep 0 = An ultra low-power wake-up event did not occur during Deep Sleep
bit 4	**DSWDT:** Deep Sleep Watchdog Timer Time-out bit 1 = The Deep Sleep Watchdog Timer timed out during Deep Sleep 0 = The Deep Sleep Watchdog Timer did not time out during Deep Sleep
bit 3	**DSRTC:** Real-Time Clock and Calendar Alarm bit 1 = The Real-Time Clock/Calendar triggered an alarm during Deep Sleep 0 = The Real-Time Clock /Calendar did not trigger an alarm during Deep Sleep
bit 2	**DSMCLR:** $\overline{\text{MCLR}}$ Event bit 1 = The $\overline{\text{MCLR}}$ pin was asserted during Deep Sleep 0 = The $\overline{\text{MCLR}}$ pin was not asserted during Deep Sleep
bit 1	**Unimplemented:** Read as ‘0’
bit 0	**DSPOR:** Power-on Reset Event bit 1 = The VDD supply POR circuit was active and a POR event was detected[(1)] 0 = The VDD supply POR circuit was not active, or was active, but did not detect a POR event

Note 1: Unlike the other bits in this register, this bit can be set outside of Deep Sleep.

4.7 Ultra Low-Power Wake-up

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

Follow these steps to use this feature:

1. Configure a remappable output pin to output the ULPOUT signal.
2. Map an INTx interrupt-on-change input function to the same pin as used for the ULPOUT output function. Alternatively, in Step 1, configure ULPOUT to output onto a PORTB interrupt-on-change pin.
3. Charge the capacitor on RA0 by configuring the RA0 pin to an output and setting it to ‘`1`’.
4. Enable interrupt-on-change (PIE bit) for the corresponding pin selected in Step 2.
5. Stop charging the capacitor by configuring RA0 as an input.
6. Discharge the capacitor by setting the ULPEN and ULPSINK bits in the WDTCON register.
7. Configure Sleep mode.
8. Enter Sleep mode.

When the voltage on RA0 drops below V_{IL}, an interrupt will be generated, which will cause the device to wake-up and execute 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 causes the device to wake-up from Sleep mode, the ULPLVL (WDTCON<5>) bit is set. When the ULPWU module causes the device to wake-up from Deep Sleep, the DSULP (DSWAKEL<5>) bit is set. Software can check these bits upon wake-up to determine the wake-up source. Also in Sleep mode, only the remappable output function, ULPWU, will output this bit value to an RPn pin for externally detecting wake-up events.

See Example 4-1 for initializing the ULPWU module.

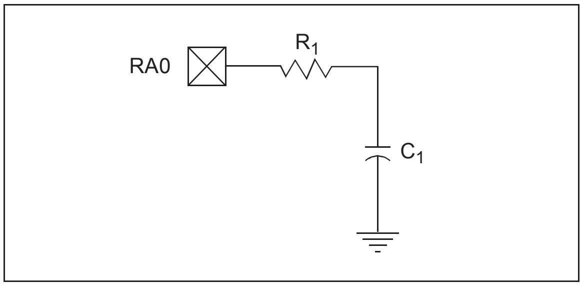
Note:	For module related bit definitions, see the WDTCON register in **Section 27.2 “Watchdog Timer (WDT)”** and the DSWAKEL register (Register 4-6).

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

```
//********************************************************************************
//Configure a remappable output pin with interrupt capability
//for ULPWU function (RP21 => RD4/INT1 in this example)
//********************************************************************************
RPOR21 = 13;// ULPWU function mapped to RP21/RD4
RPINR1 = 21;// INT1 mapped to RP21 (RD4)

//***************************
//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;

//******************************************
//Enable Interrupt for ULPW
//******************************************
//For Sleep
//(assign the ULPOUT signal in the PPS module to a pin
//which has also been assigned an interrupt capability,
//such as INT1)
INTCON3bits.INT1IF = 0;
INTCON3bits.INT1IE = 1;

//********************
//Configure Sleep Mode
//********************
//For Sleep
OSCCONbits.IDLEN = 0;

//For Deep Sleep
OSCCONbits.IDLEN = 0;  // enable deep sleep
DSCONHbits.DSEN = 1;   // Note: must be set just before executing Sleep();
//****************
//Enter Sleep Mode
//****************
Sleep();
// for sleep, execution will resume here
// for deep sleep, execution will restart at reset vector (use WDTCONbits.DS to detect)
```

A series resistor between RA0 and the external capacitor provides overcurrent protection for the RA0/AN0/C1INA/ULPWU/RP0 pin and can allow for software calibration of the time-out (see Figure 4-9).

FIGURE 4-9: SERIAL RESISTOR



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 interrupt delay. This technique will compensate for the affects of temperature, voltage and component accuracy. The ULPWU peripheral can also be configured as a simple Programmable Low-Voltage Detect (LVD) or temperature sensor.

> **Note:** For more information, refer to *AN879, Using the Microchip Ultra Low-Power Wake-up Module* application note (DS00879).

4.8 Peripheral Module Disable

All peripheral modules (except for I/O ports) also have a second control bit that can disable their functionality. These bits, known as the Peripheral Module Disable (PMD) bits, are generically named “xxxMD” (using “xxx” as the mnemonic version of the module’s name).

These bits are located in the PMDISx special function registers. In contrast to the module enable bits (generically named “xxxEN” and located in bit position seven of the control registers), the PMD bits must be set (= `1`) to disable the modules.

While the PMD and module enable bits both disable a peripheral’s functionality, the PMD bit completely shuts down the peripheral, effectively powering down all circuits and removing all clock sources. This has the additional effect of making any of the module’s control and buffer registers, mapped in the SFR space, unavailable for operations. Essentially, the peripheral ceases to exist until the PMD bit is cleared.

This differs from using the module enable bit, which allows the peripheral to be reconfigured and buffer registers preloaded, even when the peripheral’s operations are disabled.

The PMD bits are most useful in highly power-sensitive applications. In these cases, the bits can be set before the main body of the application to remove peripherals that will not be needed at all.

TABLE 4-2: LOW-POWER MODE REGISTERS

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
PMDIS3	CCP10MD	CCP9MD	CCP8MD	CCP7MD	CCP6MD	CCP5MD	CCP4MD	—	`0000 000-`
PMDIS2	—	TMR8MD	—	TMR6MD	TMR5MD	CMP3MD	CMP2MD	CMP1MD	`-0-0 0000`
PMDIS1	PSPMD[1]	CTMUMD	RTCCMD[2]	TMR4MD	TMR3MD	TMR2MD	TMR1MD	—	`0000 000-`
PMDIS0	ECCP3MD	ECCP2MD	ECCP1MD	UART2MD	UART1MD	SPI2MD	SP11MD	ADCMD	`0000 0000`

Note 1: Not implemented on 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).

2: To prevent accidental RTCC changes, the RTCCMD bit is normally locked. Use the following unlock sequence (with interrupts disabled) to successfully modify the RTCCMD bit:

1. Write 55h to EECON2.
2. Write 0AAh to EECON2.
3. Immediately write the modified RTCCMD setting to PMDIS1.

5.0 RESET

The PIC18F47J13 Family of devices differentiates among various kinds of Reset:

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

This section discusses Resets generated by MCLR, POR and BOR, and covers the operation of the various start-up timers.

For information on WDT Resets, see **Section 27.2 "Watchdog Timer (WDT)"**. For Stack Reset events, see **Section 6.1.4.4 "Stack Full and Underflow Resets"** and for Deep Sleep mode, see **Section 4.6 "Deep Sleep Mode"**.

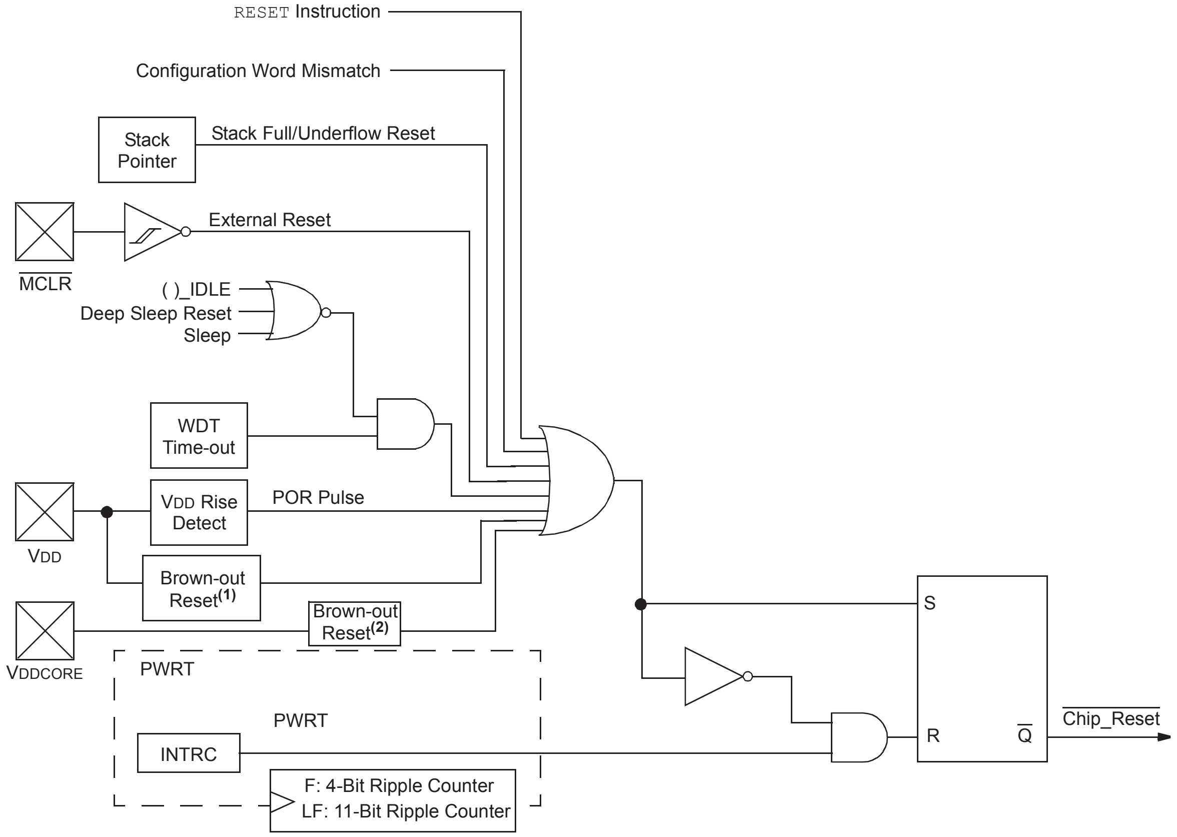
Figure 5-1 provides a simplified block diagram of the on-chip Reset circuit.

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 9.0 "Interrupts"**.

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


RESET Instruction
Configuration Word Mismatch
Stack Pointer
Stack Full/Underflow Reset
External Reset
MCLR
( )_IDLE
Deep Sleep Reset
Sleep
WDT Time-out
VDD Rise Detect
POR Pulse
VDD
Brown-out Reset(1)
Brown-out Reset(2)
VDDCORE
PWRT
PWRT
INTRC
F: 4-Bit Ripple Counter
LF: 11-Bit Ripple Counter
S
R
Q
Chip_Reset

Note 1: The VDD monitoring BOR circuit can be enabled or disabled on "LF" devices based on the CONFIG3L<DSBOREN> Configuration bit. On "F" devices, the VDD monitoring BOR circuit is only enabled during Deep Sleep mode by CONFIG3L<DSBOREN>.

2: The VDDCORE monitoring BOR circuit is only implemented on "F" devices. It is always used, except while in Deep Sleep mode. The VDDCORE monitoring BOR circuit has a trip point threshold of VBOR (parameter D005).

REGISTER 5-1: RCON: RESET CONTROL REGISTER (ACCESS FD0h)

R/W-0	U-0	R/W-1	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	$\overline{\text{CM}}$	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
bit 7							bit 0

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

bit 7 **IPEN:** Interrupt Priority Enable bit

`1` = Enable priority levels on interrupts
`0` = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

bit 6 **Unimplemented:** Read as ‘`0`’

bit 5 **$\overline{\text{CM}}$:** Configuration Mismatch Flag bit

`1` = A Configuration Mismatch Reset has not occurred
`0` = A Configuration Mismatch Reset has occurred (must be set in software after a Configuration Mismatch Reset occurs)

bit 4 **$\overline{\text{RI}}$:** `RESET` Instruction Flag bit

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

bit 3 **$\overline{\text{TO}}$:** Watchdog Time-out Flag bit

`1` = Set by power-up, `CLRWDT` instruction or `SLEEP` instruction
`0` = A WDT time-out occurred

bit 2 **$\overline{\text{PD}}$:** Power-Down Detection Flag bit

`1` = Set by power-up or by the `CLRWDT` instruction
`0` = Set by execution of the `SLEEP` instruction

bit 1 **$\overline{\text{POR}}$:** Power-on Reset Status bit

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

bit 0 **$\overline{\text{BOR}}$:** Brown-out Reset Status bit

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

Note 1: It is recommended that the $\overline{\text{POR}}$ bit be set after a Power-on Reset has been detected, so that subsequent Power-on Resets may be detected.

2: If the on-chip voltage regulator is disabled, $\overline{\text{BOR}}$ remains ‘`0`’ at all times. See **Section 5.4.1 “Detecting BOR”** for more information.

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

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

The Master Clear Reset ($\overline{\text{MCLR}}$) pin provides a method for triggering a hard external Reset of the device. A Reset is generated by holding the pin low. PIC18 extended microcontroller devices have a noise filter in the $\overline{\text{MCLR}}$ Reset path, which detects and ignores small pulses.

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

5.3 Power-on Reset (POR)

A POR 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 $\overline{\text{MCLR}}$ pin through a resistor (1 kΩ to 10 kΩ) to VDD. This will eliminate external RC components usually needed to create a POR delay.

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

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

5.4 Brown-out Reset (BOR)

The "F" devices in the PIC18F47J13 Family incorporate two types of BOR circuits: one which monitors VDDCORE and one which monitors VDD. Only one BOR circuit can be active at a time. When in normal Run mode, Idle or normal Sleep modes, the BOR circuit that monitors VDDCORE is active and will cause the device to be held in BOR if VDDCORE drops below VBOR (parameter D005). Once VDDCORE rises back above VBOR, the device will be held in Reset until the expiration of the Power-up Timer, with period, TPWRT (parameter 33).

During Deep Sleep operation, the on-chip core voltage regulator is disabled and VDDCORE is allowed to drop to VSS. If the Deep Sleep BOR circuit is enabled by the DSBOREN bit (CONFIG3L<2> = `1`), it will monitor VDD. If VDD drops below the VDSBOR threshold, the device will be held in a Reset state similar to POR. All registers will be set back to their POR Reset values and the contents of the DSGPR0 and DSGPR1 holding registers will be lost. Additionally, if any I/O pins had been configured as outputs during Deep Sleep, these pins will be tri-stated and the device will no longer be held in Deep Sleep. Once the VDD voltage recovers back above the VDSBOR threshold, and once the core voltage regulator achieves a VDDCORE voltage above VBOR, the device will begin executing code again normally, but the DS bit in the WDTCON register will not be set. The device behavior will be similar to hard cycling all power to the device.

On "LF" devices (ex: PIC18LF47J13), the VDDCORE BOR circuit is always disabled because the internal core voltage regulator is disabled. Instead of monitoring VDDCORE, PIC18LF devices in this family can still use the VDD BOR circuit to monitor VDD excursions below the VDSBOR threshold. The VDD BOR circuit can be disabled by setting the DSBOREN bit = `0`.

The VDD BOR circuit is enabled when DSBOREN = `1` on "LF" devices, or on "F" devices while in Deep Sleep with DSBOREN = `1`. When enabled, the VDD BOR circuit is extremely low power (typ. 200nA) during normal operation above ~2.3V on VDD. If VDD drops below this DSBOR arming level when the VDD BOR circuit is enabled, the device may begin to consume additional current (typ. 50 μA) as internal features of the circuit power-up. The higher current is necessary to achieve more accurate sensing of the VDD level. However, the device will not enter Reset until VDD falls below the VDSBOR threshold.

5.4.1 DETECTING BOR

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

If the voltage regulator is disabled (LF device), the VDDCORE BOR functionality is disabled. In this case, the $\overline{\text{BOR}}$ bit cannot be used to determine a Brown-out Reset event. The $\overline{\text{BOR}}$ bit is still cleared by a Power-on Reset event.

5.5 Configuration Mismatch (CM)

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

In PIC18FXXJ Flash devices, the device Configuration registers (located in the configuration memory space) are continuously monitored during operation by comparing their values to complimentary shadow registers. If a mismatch is detected between the two sets of registers, a CM Reset automatically occurs. These events are captured by the $\overline{\text{CM}}$ bit (RCON<5>). The state of the bit is set to '`0`' whenever a CM event occurs; it does not change for any other Reset event.

A CM Reset behaves similarly to $\overline{\text{MCLR}}$, `RESET` instruction, WDT time-out or Stack Event Resets. As with all hard and power Reset events, the device Configuration Words are reloaded from the Flash Configuration Words in program memory as the device restarts.

5.6 Power-up Timer (PWRT)

PIC18F47J13 Family devices incorporate an on-chip PWRT to help regulate the POR process. The PWRT is always enabled. The main function is to ensure that the device voltage is stable before code is executed.

The Power-up Timer (PWRT) of the PIC18F47J13 Family devices is a counter which uses the INTRC source as the clock input. While the PWRT is counting, the device is held in Reset.

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

5.6.1 TIME-OUT SEQUENCE

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 5-2, Figure 5-3, Figure 5-4 and Figure 5-5 all depict time-out sequences on power-up with the PWRT.

Since the time-outs occur from the POR pulse, if $\overline{\text{MCLR}}$ is kept low long enough, the PWRT will expire. Bringing $\overline{\text{MCLR}}$ high will begin execution immediately if a clock source is available (Figure 5-4). This is useful for testing purposes or to synchronize more than one PIC18F device operating in parallel.

FIGURE 5-2: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO VDD, VDD RISE < TPWRT)

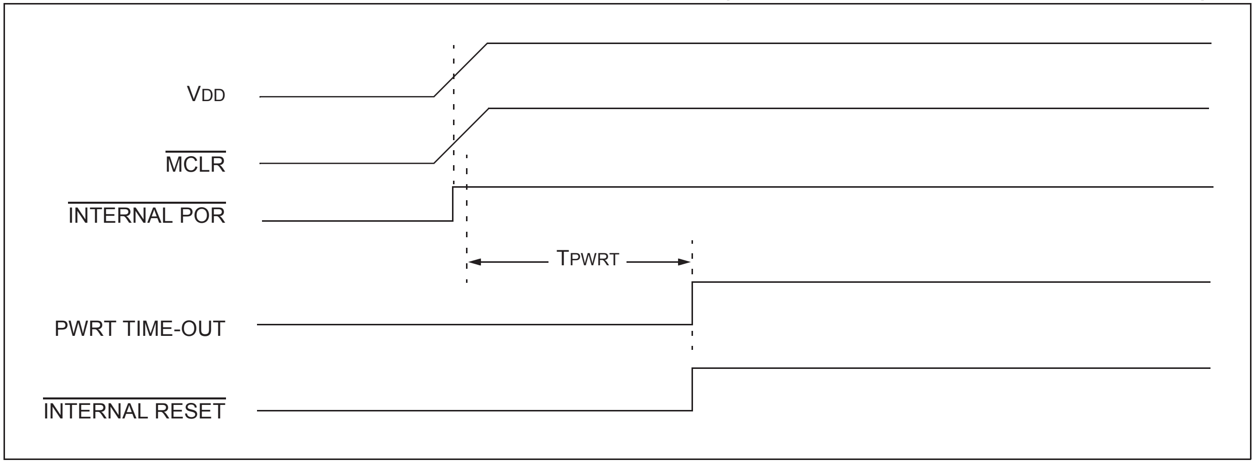

VDD
$\overline{\text{MCLR}}$
$\overline{\text{INTERNAL POR}}$
TPWRT
PWRT TIME-OUT
$\overline{\text{INTERNAL RESET}}$

FIGURE 5-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO V_{DD}): CASE 1

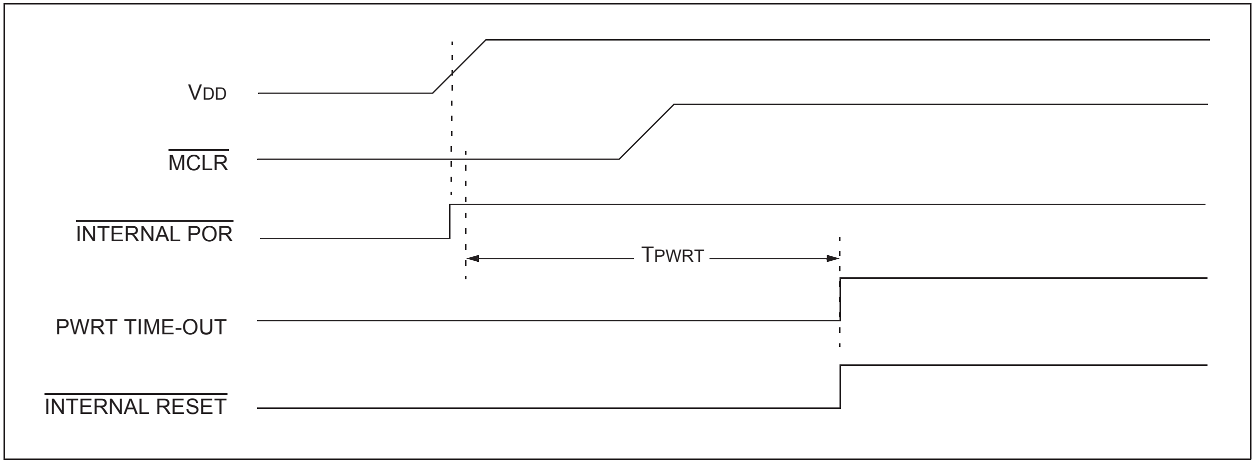

VDD
MCLR
INTERNAL POR
TPWRT
PWRT TIME-OUT
INTERNAL RESET

FIGURE 5-4: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO V_{DD}): CASE 2

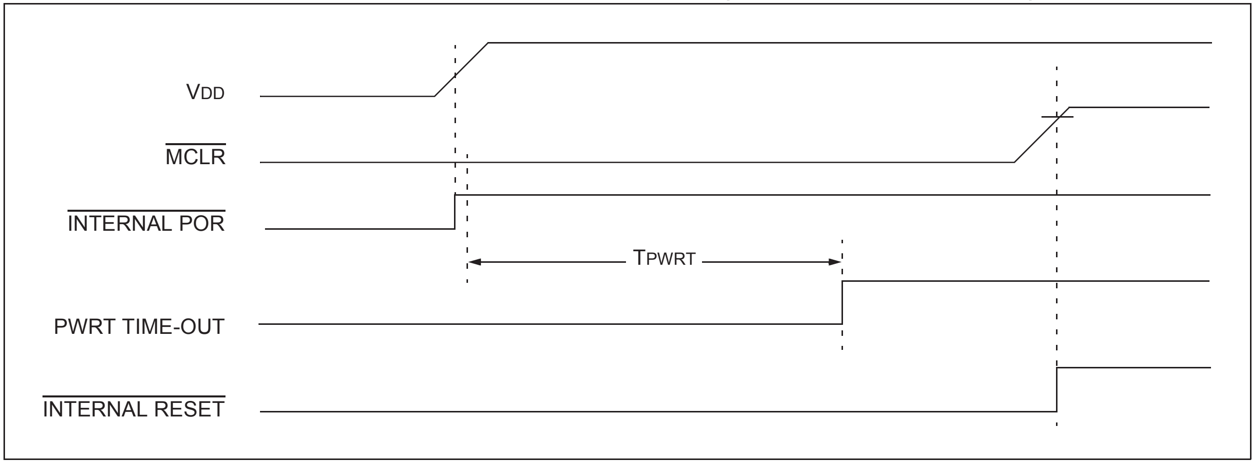
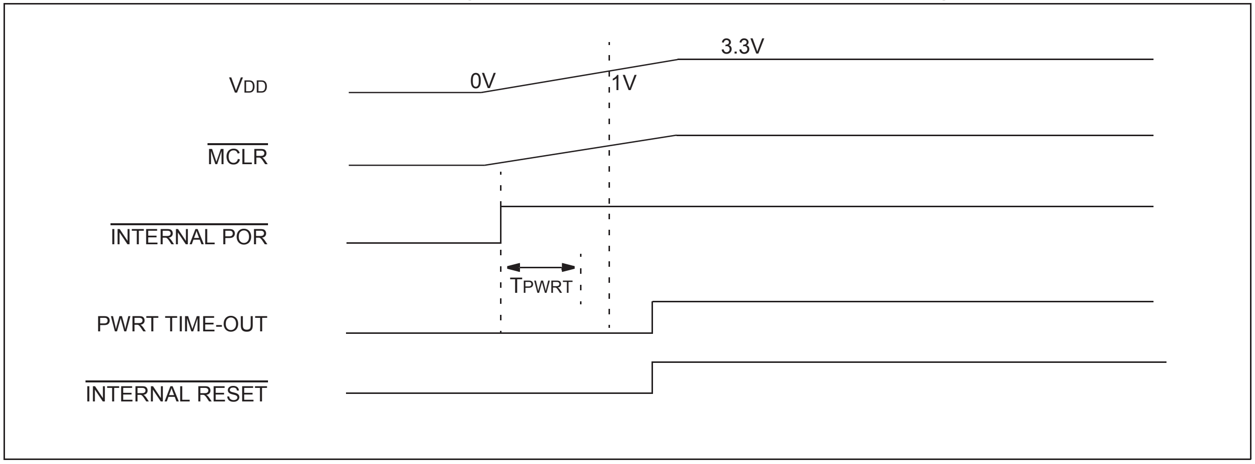

VDD
MCLR
INTERNAL POR
TPWRT
PWRT TIME-OUT
INTERNAL RESET

FIGURE 5-5: SLOW RISE TIME ($\overline{\text{MCLR}}$ TIED TO V_{DD}, V_{DD} RISE > T_{PWRT})


VDD
0V
1V
3.3V
MCLR
INTERNAL POR
TPWRT
PWRT TIME-OUT
INTERNAL RESET

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 ($\overline{\text{CM}}$, $\overline{\text{RI}}$, $\overline{\text{TO}}$, $\overline{\text{PD}}$, $\overline{\text{POR}}$ and $\overline{\text{BOR}}$) 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 POR and BOR, $\overline{\text{MCLR}}$ 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**	
		$\overline{\text{CM}}$	**$\overline{\text{RI}}$**	**$\overline{\text{TO}}$**	**$\overline{\text{PD}}$**	**$\overline{\text{POR}}$**	**$\overline{\text{BOR}}$**	**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
$\overline{\text{MCLR}}$ Reset during power-managed Run modes	0000h	u	u	1	u	u	u	u	u
$\overline{\text{MCLR}}$ Reset during power-managed Idle modes and Sleep mode	0000h	u	u	1	0	u	u	u	u
$\overline{\text{MCLR}}$ Reset during full-power execution	0000h	u	u	u	u	u	u	u	u
Stack Full Reset (STVREN = 1)	0000h	u	u	u	u	u	u	1	u
Stack Underflow Reset (STVREN = 1)	0000h	u	u	u	u	u	u	u	1
Stack Underflow Error (not an actual Reset, STVREN = 0)	0000h	u	u	u	u	u	u	u	1
WDT time-out during full-power or power-managed Run modes	0000h	u	u	0	u	u	u	u	u
WDT time-out during power-managed Idle or Sleep modes	PC + 2	u	u	0	0	u	u	u	u
Interrupt exit from power-managed modes	PC + 2	u	u	u	0	u	u	u	u

Legend: u = unchanged

Note 1: When the wake-up is due to an interrupt and the GIEH or GIEL bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

TABLE 5-2: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset `RESET` Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
TOSU	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`(1)
TOSH	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`(1)
TOSL	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`(1)
STKPTR	PIC18F2XJ13	PIC18F4XJ13	`00-0 0000`	`uu-0 0000`	`uu-u uuuu`(1)
PCLATU	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
PCLATH	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
PCL	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	PC + 2(2)
TBLPTRU	PIC18F2XJ13	PIC18F4XJ13	`--00 0000`	`--00 0000`	`--uu uuuu`
TBLPTRH	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
TBLPTRL	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
TABLAT	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
PRODH	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PRODL	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
INTCON	PIC18F2XJ13	PIC18F4XJ13	`0000 000x`	`0000 000u`	`uuuu uuuu`(3)
INTCON2	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`(3)
INTCON3	PIC18F2XJ13	PIC18F4XJ13	`1100 0000`	`1100 0000`	`uuuu uuuu`(3)
INDF0	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
POSTINC0	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
POSTDEC0	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
PREINC0	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
PLUSW0	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
FSR0H	PIC18F2XJ13	PIC18F4XJ13	`---- 0000`	`---- 0000`	`---- uuuu`
FSR0L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
WREG	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
INDF1	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
POSTINC1	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
POSTDEC1	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
PREINC1	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
PLUSW1	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
FSR1H	PIC18F2XJ13	PIC18F4XJ13	`---- 0000`	`---- 0000`	`---- uuuu`
FSR1L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
BSR	PIC18F2XJ13	PIC18F4XJ13	`---- 0000`	`---- 0000`	`---- uuuu`

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

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

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 the Reset value for a specific condition.

5: Not implemented on PIC18F2XJ13 devices.

6: Not implemented on "LF" devices.

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

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
INDF2	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
POSTINC2	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
POSTDEC2	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
PREINC2	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
PLUSW2	PIC18F2XJ13	PIC18F4XJ13	N/A	N/A	N/A
FSR2H	PIC18F2XJ13	PIC18F4XJ13	`---- 0000`	`---- 0000`	`---- uuuu`
FSR2L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
STATUS	PIC18F2XJ13	PIC18F4XJ13	`---x xxxx`	`---u uuuu`	`---u uuuu`
TMR0H	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
TMR0L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
T0CON	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
OSCCON	PIC18F2XJ13	PIC18F4XJ13	`0110 q100`	`0110 q100`	`0110 q1uu`
CM1CON	PIC18F2XJ13	PIC18F4XJ13	`0001 1111`	`0001 1111`	`uuuu uuuu`
CM2CON	PIC18F2XJ13	PIC18F4XJ13	`0001 1111`	`0001 1111`	`uuuu uuuu`
RCON(4)	PIC18F2XJ13	PIC18F4XJ13	`0-11 11qq`	`0-qq qquu`	`u-qq qquu`
TMR1H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
TMR1L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
T1CON	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`uuuu uuuu`	`uuuu uuuu`
TMR2	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
PR2	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
T2CON	PIC18F2XJ13	PIC18F4XJ13	`-000 0000`	`-000 0000`	`-uuu uuuu`
SSP1BUF	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
SSP1ADD	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
SSP1MSK	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
SSP1STAT	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
SSP1CON1	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
SSP1CON2	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
ADRESH	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
ADRESL	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
ADCON0	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
ADCON1	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
WDTCON	PIC18F2XJ13	PIC18F4XJ13	`1qq0 q000`	`1qq0 0000`	`uqqu uuuu`

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

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

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 the Reset value for a specific condition.

5: Not implemented on PIC18F2XJ13 devices.

6: Not implemented on “LF” devices.

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

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
PSTR1CON	PIC18F2XJ13	PIC18F4XJ13	00-0 0001	00-0 0001	uu-u uuuu
ECCP1AS	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
ECCP1DEL	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
CCPR1H	PIC18F2XJ13	PIC18F4XJ13	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	PIC18F2XJ13	PIC18F4XJ13	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PSTR2CON	PIC18F2XJ13	PIC18F4XJ13	00-0 0001	00-0 0001	uu-u uuuu
ECCP2AS	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
ECCP2DEL	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
CCPR2H	PIC18F2XJ13	PIC18F4XJ13	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	PIC18F2XJ13	PIC18F4XJ13	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
CTMUCONH	PIC18F2XJ13	PIC18F4XJ13	0-00 0000	0-00 0000	u-uu uuuu
CTMUCONL	PIC18F2XJ13	PIC18F4XJ13	0000 00xx	0000 00xx	uuuu uuuu
CTMUICON	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
SPBRG1	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
RCREG1	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
TXREG1	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
TXSTA1	PIC18F2XJ13	PIC18F4XJ13	0000 0010	0000 0010	uuuu uuuu
RCSTA1	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
SPBRG2	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
RCREG2	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
TXREG2	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
TXSTA2	PIC18F2XJ13	PIC18F4XJ13	0000 0010	0000 0010	uuuu uuuu
EECON2	PIC18F2XJ13	PIC18F4XJ13	---- ----	---- ----	---- ----
EECON1	PIC18F2XJ13	PIC18F4XJ13	--00 x00-	--00 q00-	--00 u00-
IPR3	PIC18F2XJ13	PIC18F4XJ13	1111 1111	1111 1111	uuuu uuuu
PIR3	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu$^{(3)}$
PIE3	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
IPR2	PIC18F2XJ13	PIC18F4XJ13	111- 1111	111- 1111	uuu- uuuu
PIR2	PIC18F2XJ13	PIC18F4XJ13	000- 0000	000- 0000	uuu- uuuu$^{(3)}$
PIE2	PIC18F2XJ13	PIC18F4XJ13	000- 0000	000- 0000	uuu- uuuu

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

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

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 the Reset value for a specific condition.

5: Not implemented on PIC18F2XJ13 devices.

6: Not implemented on "LF" devices.

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

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset `RESET` Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
IPR1	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
PIR1	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`(3)
PIE1	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
RCSTA2	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
OSCTUNE	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
T1GCON	PIC18F2XJ13	PIC18F4XJ13	`0000 0x00`	`0000 0x00`	`uuuu uxuu`
T3GCON	PIC18F2XJ13	PIC18F4XJ13	`0000 0x00`	`uuuu uxuu`	`uuuu uxuu`
TRISE(5)	—	PIC18F4XJ13	`00-- -111`	`00-- -111`	`uu-- -uuu`
TRISD(5)	—	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
TRISC	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
TRISB	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
TRISA	PIC18F2XJ13	PIC18F4XJ13	`111- 1111`	`111- 1111`	`uuu- uuuu`
PIE5	PIC18F2XJ13	PIC18F4XJ13	`--00 0000`	`--00 0000`	`--uu uuuu`
IPR4	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
PIR4	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
PIE4	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
LATE(5)	—	PIC18F4XJ13	`---- -xxx`	`---- -uuu`	`---- -uuu`
LATD(5)	—	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
LATC	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
LATB	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
LATA	PIC18F2XJ13	PIC18F4XJ13	`xxx- xxxx`	`uuu- uuuu`	`uuu- uuuu`
DMACON1	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
OSCCON2	PIC18F2XJ13	PIC18F4XJ13	`-0-1 01--`	`-0-1 u1--`	`-u-u uu--`
DMACON2	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
HLVDCON	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
PORTE(5)	—	PIC18F4XJ13	`---- -xxx`	`---- -uuu`	`---- -uuu`
PORTD(5)	—	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PORTC	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PORTB	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PORTA	PIC18F2XJ13	PIC18F4XJ13	`xxx- xxxx`	`uuu- uuuu`	`uuu- uuuu`
SPBRGH1	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
BAUDCON1	PIC18F2XJ13	PIC18F4XJ13	`0100 0-00`	`0100 0-00`	`uuuu u-uu`

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

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

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 the Reset value for a specific condition.

5: Not implemented on PIC18F2XJ13 devices.

6: Not implemented on "LF" devices.

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

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset `RESET` Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
SPBRGH2	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
BAUDCON2	PIC18F2XJ13	PIC18F4XJ13	0100 0-00	0100 0-00	uuuu u-uu
TMR3H	PIC18F2XJ13	PIC18F4XJ13	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	PIC18F2XJ13	PIC18F4XJ13	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	PIC18F2XJ13	PIC18F4XJ13	0000 0000	uuuu uuuu	uuuu uuuu
TMR4	PIC18F2XJ13	PIC18F4XJ13	0000 0000	uuuu uuuu	uuuu uuuu
PR4	PIC18F2XJ13	PIC18F4XJ13	1111 1111	1111 1111	uuuu uuuu
T4CON	PIC18F2XJ13	PIC18F4XJ13	-000 0000	-000 0000	-uuu uuuu
SSP2BUF	PIC18F2XJ13	PIC18F4XJ13	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSP2ADD	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
SSP2MSK	PIC18F2XJ13	PIC18F4XJ13	1111 1111	1111 1111	uuuu uuuu
SSP2STAT	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
SSP2CON1	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
SSP2CON2	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
CMSTAT	PIC18F2XJ13	PIC18F4XJ13	---- -111	---- -111	---- -uuu
PMADDRH(5)	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMDOUT1H(5)	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMADDRL	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMDOUT1L	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMDIN1H	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMDIN1L	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
TXADDRL	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
TXADDRH	PIC18F2XJ13	PIC18F4XJ13	---- 0000	---- 0000	---- uuuu
RXADDRL	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
RXADDRH	PIC18F2XJ13	PIC18F4XJ13	---- 0000	---- 0000	---- uuuu
DMABCL	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
DMABCH	PIC18F2XJ13	PIC18F4XJ13	---- --00	---- --00	---- --uu
PMCONH	PIC18F2XJ13	PIC18F4XJ13	0--0 0000	0--0 0000	u--u uuuu
PMCONL	—	PIC18F4XJ13	000- 0000	000- 0000	uuu- uuuu
PMMODEH	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMMODEL	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMDOUT2H	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu

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

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

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 the Reset value for a specific condition.

5: Not implemented on PIC18F2XJ13 devices.

6: Not implemented on “LF” devices.

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

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset `RESET` Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
PMDOUT2L	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMDIN2H	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMDIN2L	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMEH	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMEL	—	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
PMSTATH	—	PIC18F4XJ13	00-- 0000	00-- 0000	uu-- uuuu
PMSTATL	—	PIC18F4XJ13	10-- 1111	10-- 1111	uu-- uuuu
CVRCON	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
CCPTMRS0	PIC18F2XJ13	PIC18F4XJ13	0000 0000	uuuu uuuu	uuuu uuuu
CCPTMRS1	PIC18F2XJ13	PIC18F4XJ13	00-0 -000	uu-u -uuu	uu-u -uuu
CCPTMRS2	PIC18F2XJ13	PIC18F4XJ13	---0 -000	---u -uuu	---u -uuu
DSGPR1[6]	PIC18F2XJ13	PIC18F4XJ13	uuuu uuuu	uuuu uuuu	uuuu uuuu
DSGPR0[6]	PIC18F2XJ13	PIC18F4XJ13	uuuu uuuu	uuuu uuuu	uuuu uuuu
DSCONH[6]	PIC18F2XJ13	PIC18F4XJ13	0--- -000	0--- -uuu	u--- -uuu
DSCONL[6]	PIC18F2XJ13	PIC18F4XJ13	---- -000	---- -u00	---- -uuu
DSWAKEH[6]	PIC18F2XJ13	PIC18F4XJ13	---- ---0	---- ---0	---- ---u
DSWAKEL[6]	PIC18F2XJ13	PIC18F4XJ13	0-00 00-1	0-00 00-0	u-uu uu-u
ANCON1	PIC18F2XJ13	PIC18F4XJ13	00-0 0000	00-0 0000	uu-u uuuu
ANCON0	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
ALRMCFG	PIC18F2XJ13	PIC18F4XJ13	0000 0000	uuuu uuuu	uuuu uuuu
ALRMRPT	PIC18F2XJ13	PIC18F4XJ13	0000 0000	uuuu uuuu	uuuu uuuu
ALRMVALH	PIC18F2XJ13	PIC18F4XJ13	xxxx xxxx	uuuu uuuu	uuuu uuuu
ALRMVALL	PIC18F2XJ13	PIC18F4XJ13	xxxx xxxx	uuuu uuuu	uuuu uuuu
ODCON1	PIC18F2XJ13	PIC18F4XJ13	0000 0000	0000 0000	uuuu uuuu
ODCON2	PIC18F2XJ13	PIC18F4XJ13	---- 0000	---- 0000	---- uuuu
ODCON3	PIC18F2XJ13	PIC18F4XJ13	0--- --00	0--- --00	u--- --uu
RTCCFG	PIC18F2XJ13	PIC18F4XJ13	0-00 0000	u-uu uuuu	u-uu uuuu
RTCCAL	PIC18F2XJ13	PIC18F4XJ13	0000 0000	uuuu uuuu	uuuu uuuu
REFOCON	PIC18F2XJ13	PIC18F4XJ13	0-00 0000	0-00 0000	u-uu uuuu
PADCFG1	PIC18F2XJ13	PIC18F4XJ13	---- -000	---- -000	---- -uuu
RTCVALH	PIC18F2XJ13	PIC18F4XJ13	0xxx xxxx	0uuu uuuu	0uuu uuuu
RTCVALL	PIC18F2XJ13	PIC18F4XJ13	0xxx xxxx	0uuu uuuu	0uuu uuuu

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

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

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 the Reset value for a specific condition.

5: Not implemented on PIC18F2XJ13 devices.

6: Not implemented on "LF" devices.

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

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset `RESET` Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
CM3CON	PIC18F2XJ13	PIC18F4XJ13	`0001 1111`	`0001 1111`	`uuuu uuuu`
TMR5H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
TMR5L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
T5CON	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`uuuu uuuu`	`uuuu uuuu`
T5GCON	PIC18F2XJ13	PIC18F4XJ13	`0000 0x00`	`uuuu uquu`	`uuuu uquu`
TMR6	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
PR6	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
T6CON	PIC18F2XJ13	PIC18F4XJ13	`-000 0000`	`-000 0000`	`-uuu uuuu`
TMR8	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
PR8	PIC18F2XJ13	PIC18F4XJ13	`1111 1111`	`1111 1111`	`uuuu uuuu`
T8CON	PIC18F2XJ13	PIC18F4XJ13	`-000 0000`	`-000 0000`	`-uuu uuuu`
PSTR3CON	PIC18F2XJ13	PIC18F4XJ13	`00-0 0001`	`00-0 0001`	`uu-u uuuu`
ECCP3AS	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
ECCP3DEL	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
CCPR3H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR3L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP3CON	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
CCPR4H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR4L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP4CON	PIC18F2XJ13	PIC18F4XJ13	`--00 0000`	`--00 0000`	`--uu uuuu`
CCPR5H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR5L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP5CON	PIC18F2XJ13	PIC18F4XJ13	`--00 0000`	`--00 0000`	`--uu uuuu`
CCPR6H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR6L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP6CON	PIC18F2XJ13	PIC18F4XJ13	`--00 0000`	`--00 0000`	`--uu uuuu`
CCPR7H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR7L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP7CON	PIC18F2XJ13	PIC18F4XJ13	`--00 0000`	`--00 0000`	`--00 0000`
CCPR8H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR8L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP8CON	PIC18F2XJ13	PIC18F4XJ13	`--00 0000`	`--00 0000`	`--uu uuuu`

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

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

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 the Reset value for a specific condition.

5: Not implemented on PIC18F2XJ13 devices.

6: Not implemented on “LF” devices.

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

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset `RESET` Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
CCPR9H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR9L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP9CON	PIC18F2XJ13	PIC18F4XJ13	`--00 0000`	`--00 0000`	`--uu uuuu`
CCPR10H	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCPR10L	PIC18F2XJ13	PIC18F4XJ13	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
CCP10CON	PIC18F2XJ13	PIC18F4XJ13	`--00 0000`	`--00 0000`	`--uu uuuu`
RPINR24	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR23	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR22	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR21	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR17	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR16	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR14	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR13	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR12	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR9	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR8	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR7	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR15	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR6	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR4	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR3	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR2	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPINR1	PIC18F2XJ13	PIC18F4XJ13	`---1 1111`	`---1 1111`	`---u uuuu`
RPOR24	—	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR23	—	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR22	—	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR21	—	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR20	—	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR19	—	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR18	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR17	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`

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

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

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 the Reset value for a specific condition.

5: Not implemented on PIC18F2XJ13 devices.

6: Not implemented on “LF” devices.

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

Register	Applicable Devices		Power-on Reset, Brown-out Reset, Wake From Deep Sleep	$\overline{\text{MCLR}}$ Resets WDT Reset `RESET` Instruction Stack Resets CM Resets	Wake-up via WDT or Interrupt
RPOR16	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR15	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR14	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR13	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR12	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR11	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR10	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR9	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR8	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR7	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR6	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR5	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR4	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR3	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR2	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR1	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
RPOR0	PIC18F2XJ13	PIC18F4XJ13	`---0 0000`	`---0 0000`	`---u uuuu`
PPSCON	PIC18F2XJ13	PIC18F4XJ13	`---- ---0`	`---- ---0`	`---- ---u`
PMDIS3	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
PMDIS2	PIC18F2XJ13	PIC18F4XJ13	`-0-0 0000`	`-0-0 0000`	`-u-u uuuu`
PMDIS1	PIC18F2XJ13	PIC18F4XJ13	`0000 000-`	`0000 000-`	`uuuu uuu-`
PMDIS0	PIC18F2XJ13	PIC18F4XJ13	`0000 0000`	`0000 0000`	`uuuu uuuu`
ADCTRIG	PIC18F2XJ13	PIC18F4XJ13	`---- --00`	`---- --00`	`---- --uu`

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

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

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 the Reset value for a specific condition.

5: Not implemented on PIC18F2XJ13 devices.

6: Not implemented on "LF" devices.

6.0 MEMORY ORGANIZATION

There are two types of memory in PIC18 Flash microcontrollers:

- Program Memory
- Data RAM

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

Section 7.0 "Flash Program Memory" provides additional information on the operation of the Flash program memory.

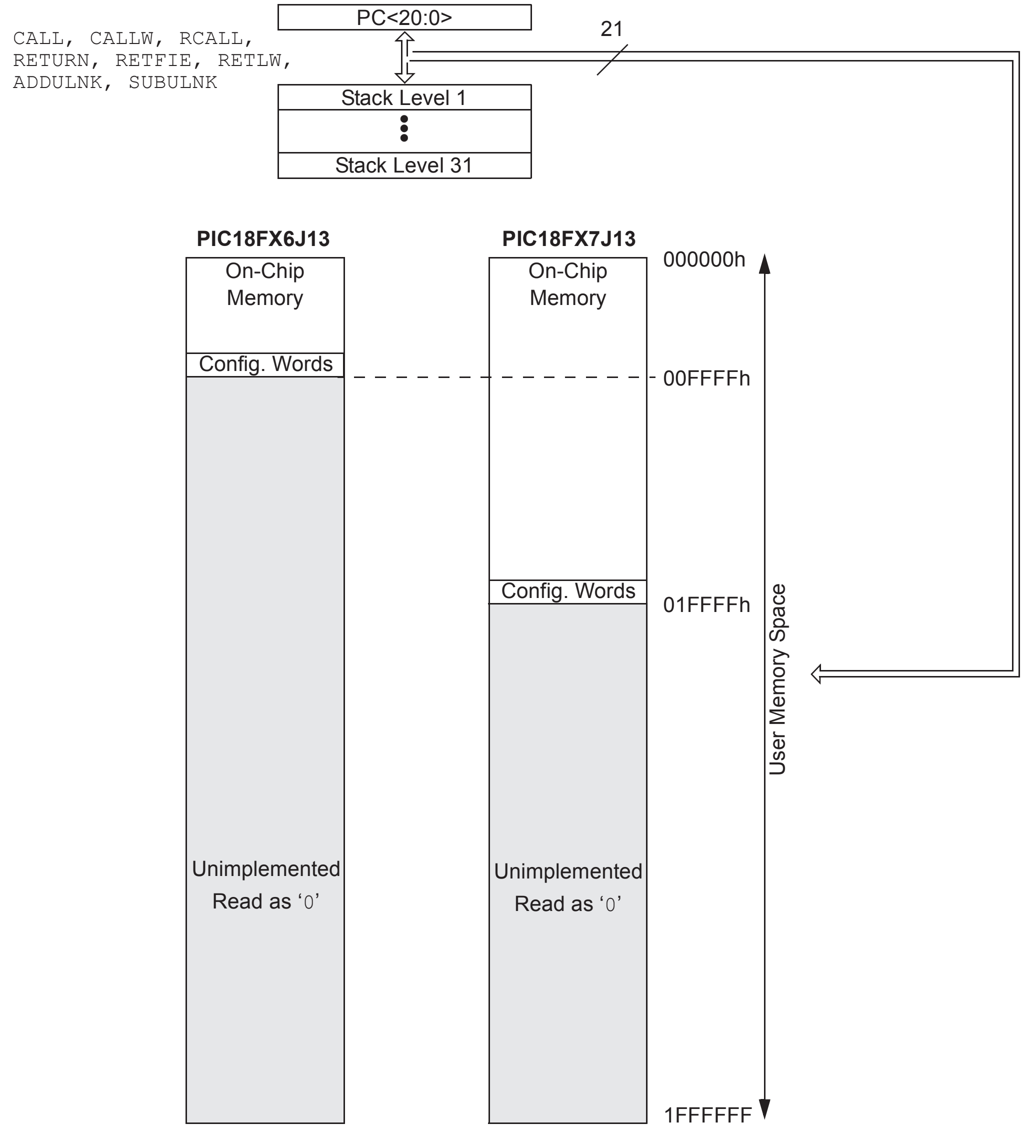
6.1 Program Memory Organization

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

The PIC18F47J13 Family offers a range of on-chip Flash program memory sizes, from 64 Kbytes (up to 32,768 single-word instructions) to 128 Kbytes (65,536 single-word instructions).

Figure 6-1 provides the program memory maps for individual family devices.

FIGURE 6-1: MEMORY MAPS FOR PIC18F47J13 FAMILY DEVICES


PC<20:0>
CALL, CALLW, RCALL,
RETURN, RETFIE, RETLW,
ADDULNK, SUBULNK
21
Stack Level 1
Stack Level 31
PIC18FX6J13
On-Chip Memory
Config. Words
Unimplemented
Read as '0'
PIC18FX7J13
On-Chip Memory
Config. Words
Unimplemented
Read as '0'
000000h
00FFFFh
01FFFFh
1FFFFF
User Memory Space

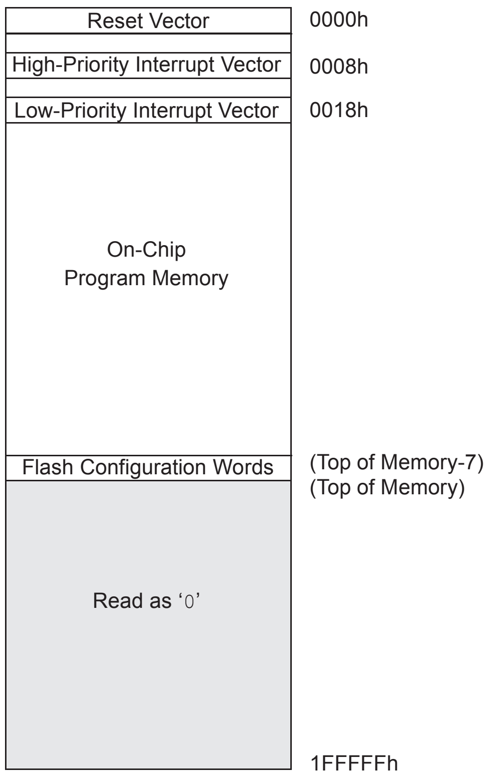
Note: Sizes of memory areas are not to scale. Sizes of program memory areas are enhanced to show detail.

6.1.1 HARD MEMORY VECTORS

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

PIC18 devices also have two interrupt vector addresses for handling high-priority and low-priority interrupts. The high-priority interrupt vector is located at 0008h and the low-priority interrupt vector at 0018h. Figure 6-2 provides their locations in relation to the program memory map.

FIGURE 6-2: HARD VECTOR AND CONFIGURATION WORD LOCATIONS FOR PIC18F47J13 FAMILY DEVICES



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

6.1.2 FLASH CONFIGURATION WORDS

Because PIC18F47J13 Family devices do not have persistent configuration memory, the top four words of on-chip program memory are reserved for configuration information. On Reset, the configuration information is copied into the Configuration registers.

The Configuration Words are stored in their program memory location in numerical order, starting with the lower byte of CONFIG1 at the lowest address and ending with the upper byte of CONFIG4.

Table 6-1 provides the actual addresses of the Flash Configuration Word for devices in the PIC18F47J13 Family. Figure 6-2 displays their location in the memory map with other memory vectors.

Additional details on the device Configuration Words are provided in **Section 27.1 "Configuration Bits"**.

TABLE 6-1: FLASH CONFIGURATION WORD FOR PIC18F47J13 FAMILY DEVICES

Device	Program Memory (Kbytes)	Configuration Word Addresses
PIC18F26J13	64	FFF8h to FFFFh
PIC18F46J13		
PIC18F27J13	128	1FFF8h to 1FFFFh
PIC18F47J13		

6.1.3 PROGRAM COUNTER

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

The contents of PCLATH and PCLATU are transferred to the program counter by any operation that writes to PCL. Similarly, the upper 2 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 6.1.6.1 "Computed GOTO"**).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit (LSb) 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.

6.1.4 RETURN ADDRESS STACK

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

The stack operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer (SP), 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 (SFRs). 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.

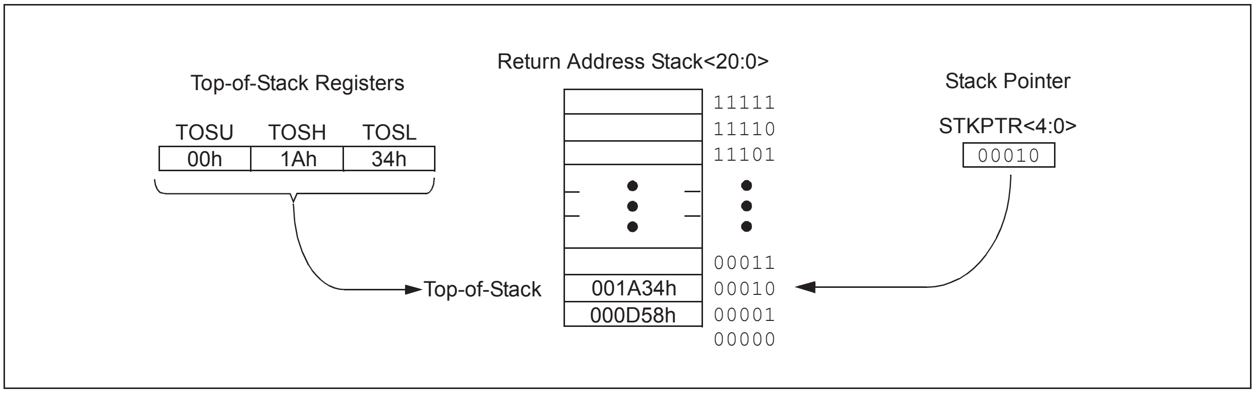
The Stack Pointer is initialized to '`00000`' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '`00000`'; this is only a Reset value. Status bits indicate if the stack is full, has overflowed or has underflowed.

6.1.4.1 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 (and `ADDULNK` and `SUBULNK` instructions if the extended instruction set is enabled), the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user-defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

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

FIGURE 6-3: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



6.1.4.2 Return Stack Pointer (STKPTR)

The STKPTR register (Register 6-1) contains the Stack Pointer value, the STKFUL (Stack Full) 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 of 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 Power-on Reset (POR).

The action that takes place when the stack becomes full depends on the state of the Stack Overflow Reset Enable (STVREN) Configuration bit.

Refer to **Section 27.1 “Configuration Bits”** for the device Configuration bits’ description.

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 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.

6.1.4.3 `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 necessary. 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 (ACCESS FFCh)

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 became full or overflowed
0 = Stack has not become full or overflowed

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

bit 5 **Unimplemented**: Read as ‘0’

bit 4-0 **SP<4:0>:** Stack Pointer Location bits

Note 1: Bits 7 and 6 are cleared by user software or by a POR.

6.1.4.4 Stack Full and Underflow Resets

Device Resets on stack overflow and stack underflow conditions are enabled by setting the STVREN bit in Configuration Register 1L. When STVREN is set, a full or underflow condition sets the appropriate STKFUL or STKUNF bit and then causes a device Reset. When STVREN is cleared, a full or underflow condition sets the appropriate STKFUL or STKUNF bit, but does not cause a device Reset. The STKFUL or STKUNF bits are cleared by the user software or a POR.

6.1.5 FAST REGISTER STACK (FRS)

A Fast Register Stack (FRS) 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 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-priority 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 FRS for returns from interrupt. If no interrupts are used, the FRS 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 FRS.

Example 6-1 provides a source code example that uses the FRS 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
```

6.1.6 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

6.1.6.1 Computed `GOTO`

A computed `GOTO` is accomplished by adding an offset to the PC. 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 executed instruction 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 PC should advance and should be multiples of 2 (LSb = `0`).

In this method, only one byte may be stored in each instruction location, but 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
        .
        .
        .
```

6.1.6.2 Table Reads

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

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

Table read operation is discussed further in **Section 7.1 “Table Reads and Table Writes”**.

6.2 PIC18 Instruction Cycle

6.2.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the PC is incremented on every Q1; the instruction is fetched from the program memory and latched into the Instruction Register (IR) during Q4. The instruction is decoded and executed during the following Q1 through Q4. Figure 6-4 illustrates the clocks and instruction execution flow.

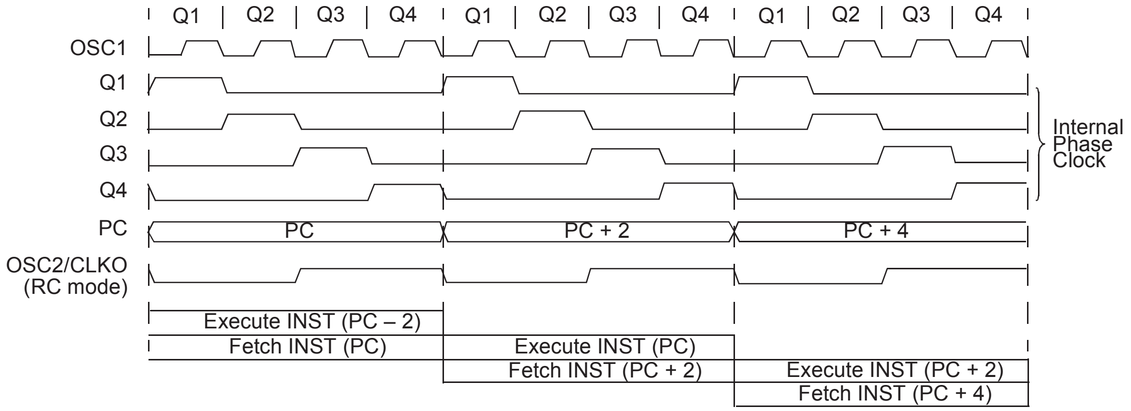
6.2.2 INSTRUCTION FLOW/PIPELINING

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 causes the PC to change (e.g., `GOTO`), then two cycles are required to complete the instruction (Example 6-3).

A fetch cycle begins with the PC incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the 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

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

Note: All instructions are single-cycle, except for any program branches. These take two cycles since the fetch instruction is “flushed” from the pipeline while the new instruction is being fetched and then executed.

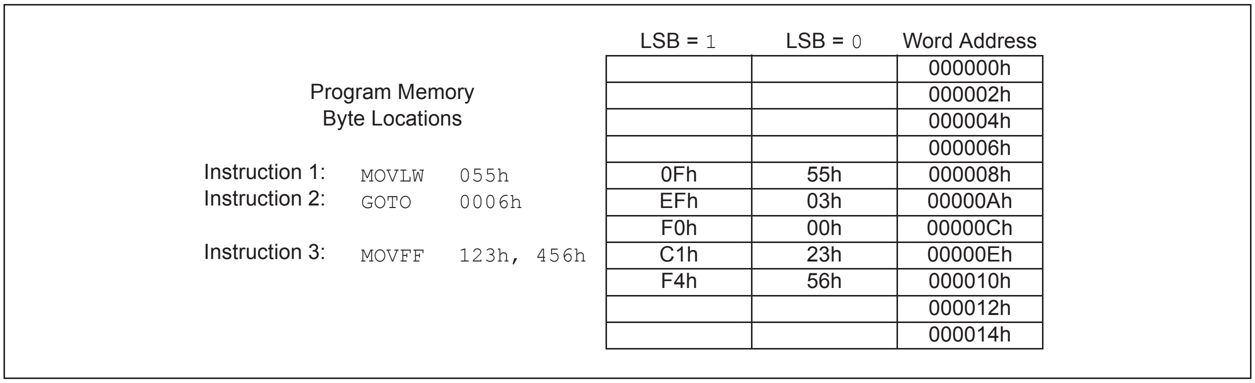
6.2.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as 2 bytes or 4 bytes in program memory. The Least Significant Byte (LSB) of an instruction word is always stored in a program memory location with an even address (LSB = `0`). To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read ‘`0`’ (see **Section 6.1.3 “Program Counter”**).

Figure 6-5 provides 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 displays how the instruction, `GOTO 0006h`, is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 28.0 “Instruction Set Summary”** provides further details of the instruction set.

FIGURE 6-5: INSTRUCTIONS IN PROGRAM MEMORY



Program Memory
Byte Locations

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

6.2.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four, two-word instructions: `CALL`, `MOVFF`, `GOTO` and `LSFR`. In all cases, the second word of the instructions always has ‘`1111`’ as its four Most Significant bits (MSbs); 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 illustrates how this works.

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

EXAMPLE 6-4: TWO-WORD INSTRUCTIONS

CASE 1:

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

CASE 2:

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

6.3 Data Memory Organization

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

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. The PIC18F47J13 Family implements all available banks and provides 3.8 Kbytes of data memory available to the user. Figure 6-6 provides the data memory organization for the devices.

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

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

To ensure that commonly used registers (select SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a 256-byte memory space that provides fast access to select SFRs and the lower portion of GPR Bank 0 without using the BSR. **Section 6.3.2 "Access Bank"** provides a detailed description of the Access RAM.

6.3.1 BANK SELECT REGISTER

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

Most instructions in the PIC18 instruction set make use of the Bank Pointer, known as the Bank Select Register (BSR). This SFR holds the 4 MSbs of a location's address; the instruction itself includes the 8 LSbs. Only the four lower bits of the BSR are implemented (BSR<3:0>). The upper four bits are unused; they will always read '`0`' and cannot be written to. The BSR can be loaded directly by using the `MOVLB` instruction.

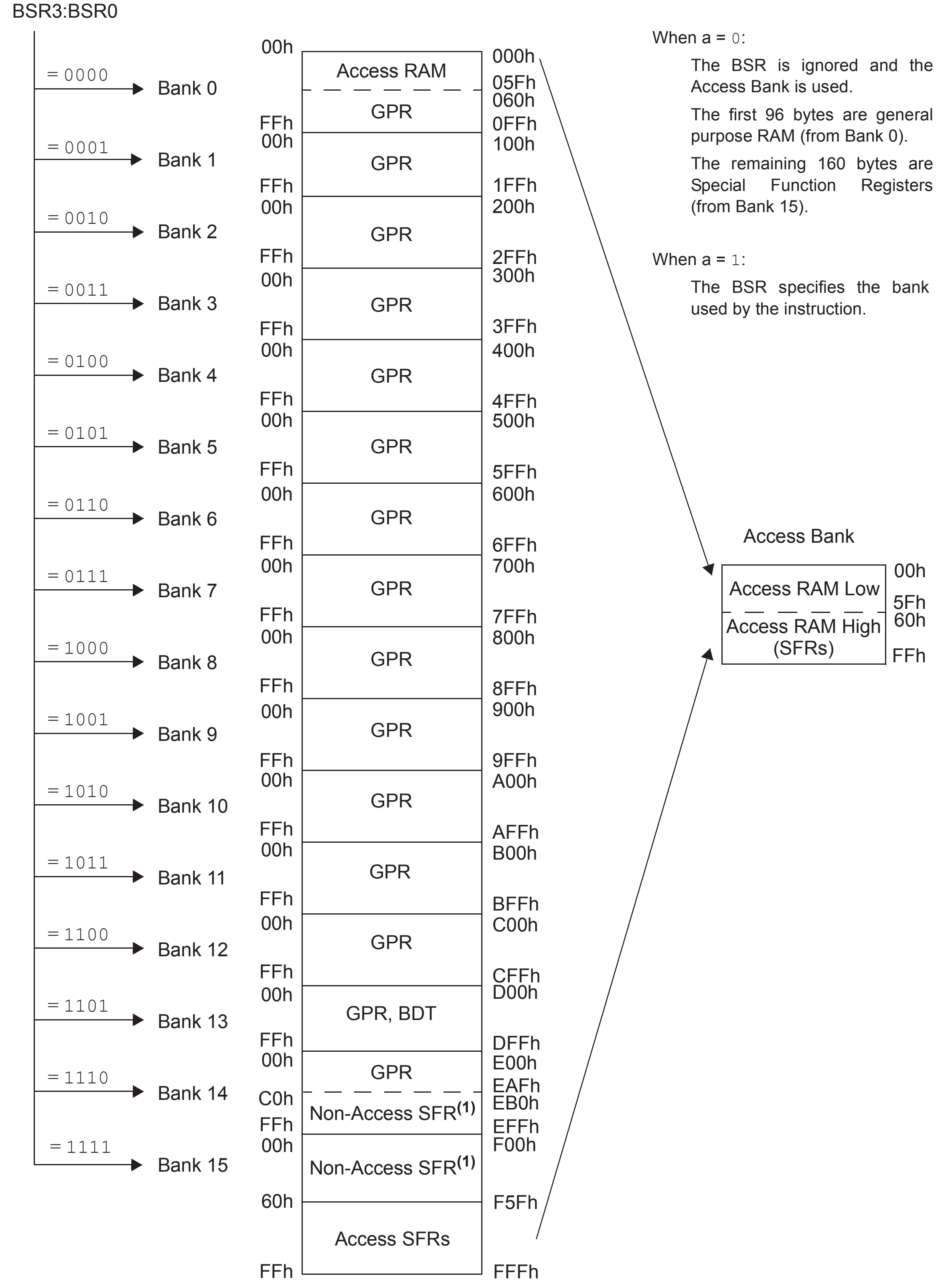
The value of the BSR indicates the bank in data memory. The 8 bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is illustrated in Figure 6-7.

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

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. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the Access Bank to locate their target registers.

FIGURE 6-6: DATA MEMORY MAP FOR PIC18F47J13 FAMILY DEVICES


BSR3:BSR0
= 0000
Bank 0
= 0001
Bank 1
= 0010
Bank 2
= 0011
Bank 3
= 0100
Bank 4
= 0101
Bank 5
= 0110
Bank 6
= 0111
Bank 7
= 1000
Bank 8
= 1001
Bank 9
= 1010
Bank 10
= 1011
Bank 11
= 1100
Bank 12
= 1101
Bank 13
= 1110
Bank 14
= 1111
Bank 15
00h
Access RAM
GPR
FFh
000h
05Fh
060h
0FFh
100h
1FFh
200h
2FFh
300h
3FFh
400h
4FFh
500h
5FFh
600h
6FFh
700h
7FFh
800h
8FFh
900h
9FFh
A00h
AFFh
B00h
BFFh
C00h
CFFh
D00h
GPR, BDT
DFFh
E00h
EAFh
EB0h
C0h
Non-Access SFR(1)
EFFh
F00h
Non-Access SFR(1)
60h
F5Fh
Access SFRs
FFFh
Access Bank
Access RAM Low
Access RAM High
(SFRs)
00h
5Fh
60h
FFh

When a = `0`:

The BSR is ignored and the Access Bank is used.

The first 96 bytes are general purpose RAM (from Bank 0).

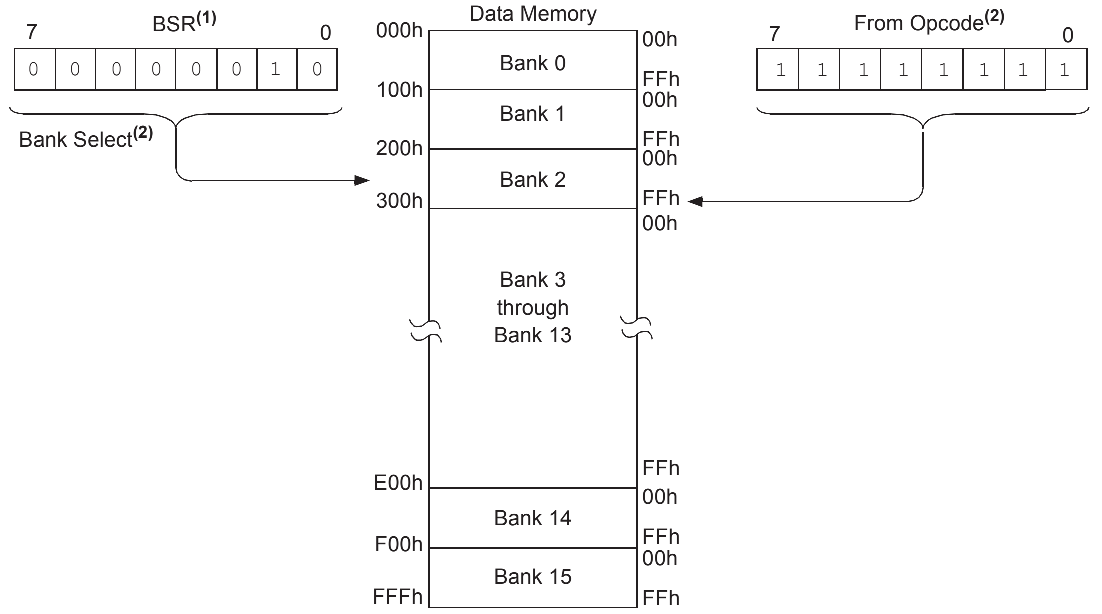
The remaining 160 bytes are Special Function Registers (from Bank 15).

When a = `1`:

The BSR specifies the bank used by the instruction.

Note 1: Addresses, EB0h through F5Fh, are not part of the Access Bank. Either the `BANKED` or the `MOVFF` instruction should be used to access these SFRs.

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 always ensure that the correct bank is selected. Otherwise, data may be read from, or written to, the wrong location. This can be disastrous if a GPR is the intended target of an operation, but an SFR is written to instead. Verifying and/or changing the BSR for each read or write to data memory can become very inefficient.

To streamline access for the most commonly used data memory locations, the data memory is configured with an Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Bank 15. The lower half is known as the Access RAM and is composed of GPRs. The upper half is where the device's SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address (Figure 6-6).

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; 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 upward toward the bottom of the SFR area. GPRs are not initialized by a POR and are unchanged on all other Resets.

6.3.4 SPECIAL FUNCTION REGISTERS

The SFRs are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy more than the top half of Bank 15 (F40h to FFFh). Table 6-2, Table 6-3 and Table 6-4 provide a list of these registers.

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 corresponding chapters, while the ALU's STATUS register is described later in this section. Registers related to the operation of the peripheral features are described in the chapter for that peripheral.

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

Note: The SFRs located between EB0h and F5Fh are not part of the Access Bank. Either `BANKED` instructions (using BSR) or the `MOVFF` instruction should be used to access these locations. When programming in MPLAB® C18, the compiler will automatically use the appropriate addressing mode.

TABLE 6-2: ACCESS BANK SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(1)	FBFh	PSTR1CON	F9Fh	IPR1	F7Fh	SPBRGH1
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	ECCP1AS	F9Eh	PIR1	F7Eh	BAUDCON1
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	ECCP1DEL	F9Dh	PIE1	F7Dh	SPBRGH2
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	CCPR1H	F9Ch	RCSTA2	F7Ch	BAUDCON2
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	CCPR1L	F9Bh	OSCTUNE	F7Bh	TMR3H
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP1CON	F9Ah	T1GCON	F7Ah	TMR3L
FF9h	PCL	FD9h	FSR2L	FB9h	PSTR2CON	F99h	IPR5	F79h	T3CON
FF8h	TBLPTRU	FD8h	STATUS	FB8h	ECCP2AS	F98h	PIR5	F78h	TMR4
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	ECCP2DEL	F97h	T3GCON	F77h	PR4
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	CCPR2H	F96h	TRISE	F76h	T4CON
FF5h	TABLAT	FD5h	T0CON	FB5h	CCPR2L	F95h	TRISD	F75h	SSP2BUF
FF4h	PRODH	FD4h	—(5)	FB4h	CCP2CON	F94h	TRISC	F74h	SSP2ADD(3)
FF3h	PRODL	FD3h	OSCCON	FB3h	CTMUCONH	F93h	TRISB	F73h	SSP2STAT
FF2h	INTCON	FD2h	CM1CON	FB2h	CTMUCONL	F92h	TRISA	F72h	SSP2CON1
FF1h	INTCON2	FD1h	CM2CON	FB1h	CTMUICON	F91h	PIE5	F71h	SSP2CON2
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRG1	F90h	IPR4	F70h	CMSTAT
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	RCREG1	F8Fh	PIR4	F6Fh	PMADDRH(2,4)
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	TXREG1	F8Eh	PIE4	F6Eh	PMADDRL(2,4)
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXSTA1	F8Dh	LATE(2)	F6Dh	PMDIN1H(2)
FECh	PREINC0(1)	FCCh	TMR2	FACh	RCSTA1	F8Ch	LATD(2)	F6Ch	PMDIN1L(2)
FEBh	PLUSW0(1)	FCBh	PR2	FABh	SPBRG2	F8Bh	LATC	F6Bh	TXADDRL
FEAh	FSR0H	FCAh	T2CON	FAAh	RCREG2	F8Ah	LATB	F6Ah	TXADDRH
FE9h	FSR0L	FC9h	SSP1BUF	FA9h	TXREG2	F89h	LATA	F69h	RXADDRL
FE8h	WREG	FC8h	SSP1ADD(3)	FA8h	TXSTA2	F88h	DMACON1	F68h	RXADDRH
FE7h	INDF1(1)	FC7h	SSP1STAT	FA7h	EECON2	F87h	OSCCON2(5)	F67h	DMABCL
FE6h	POSTINC1(1)	FC6h	SSP1CON1	FA6h	EECON1	F86h	DMACON2	F66h	DMABCH
FE5h	POSTDEC1(1)	FC5h	SSP1CON2	FA5h	IPR3	F85h	HLVDCON	F65h	—
FE4h	PREINC1(1)	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE(2)	F64h	—
FE3h	PLUSW1(1)	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD(2)	F63h	—
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC	F62h	—
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB	F61h	—
FE0h	BSR	FC0h	WDTCON	FA0h	PIE2	F80h	PORTA	F60h	—

Note 1: This is not a physical register.
2: This register is not available on 28-pin devices.
3: SSPxADD and SSPxMSK share the same address.
4: PMADDRH and PMDOUTH share the same address, and PMADDRL and PMDOUTL share the same address. PMADDRx is used in Master modes and PMDOUTx is used in Slave modes.
5: Reserved; do not write to this location.

TABLE 6-3: NON-ACCESS BANK SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name	Address	Name	Address	Name
F5Fh	PMCONH	F3Fh	RTCCFG	F1Fh	PR6	EFFh	RPINR24	EDFh	—	EBFh	PPSCON
F5Eh	PMCONL	F3Eh	RTCCAL	F1Eh	T6CON	EFEh	RPINR23	EDEh	—	EBEh	—
F5Dh	PMMODEH	F3Dh	REFOCON	F1Dh	TMR8	EFDh	RPINR22	EDDh	—	EBDh	—
F5Ch	PMMODEL	F3Ch	PADCFG1	F1Ch	PR8	EFCh	RPINR21	EDCh	—	EBCh	PMDIS3
F5Bh	PMDOUT2H	F3Bh	RTCVALH	F1Bh	T8CON	EFBh	—	EDBh	—	EBBh	PMDIS2
F5Ah	PMDOUT2L	F3Ah	RTCVALL	F1Ah	PSTR3CON	EFAh	—	EDAh	—	EBAh	PMDIS1
F59h	PMDIN2H	F39h	—	F19h	ECCP3AS	EF9h	—	ED9h	—	EB9h	PMDIS0
F58h	PMDIN2L	F38h	—	F18h	ECCP3DEL	EF8h	RPINR17	ED8h	RPOR24	EB8h	ADCTRIG
F57h	PMEH	F37h	—	F17h	CCPR3H	EF7h	RPINR16	ED7h	RPOR23	EB7h	—
F56h	PMEL	F36h	—	F16h	CCPR3L	EF6h	—	ED6h	RPOR22	EB6h	—
F55h	PMSTATH	F35h	—	F15h	CCP3CON	EF5h	—	ED5h	RPOR21	EB5h	—
F54h	PMSTATL	F34h	—	F14h	CCPR4H	EF4h	RPINR14	ED4h	RPOR20	EB4h	—
F53h	CVRCON	F33h	—	F13h	CCPR4L	EF3h	RPINR13	ED3h	RPOR19	EB3h	—
F52h	CCPTMRS0	F32h	—	F12h	CCP4CON	EF2h	RPINR12	ED2h	RPOR18	EB2h	—
F51h	CCPTMRS1	F31h	—	F11h	CCPR5H	EF1h	—	ED1h	RPOR17	EB1h	—
F50h	CCPTMRS2	F30h	—	F10h	CCPR5L	EF0h	—	ED0h	RPOR16	EB0h	—
F4Fh	DSGPR1	F2Fh	—	F0Fh	CCP5CON	EEFh	—	ECFh	RPOR15		
F4Eh	DSGPR0	F2Eh	—	F0Eh	CCPR6H	EEEh	—	ECEh	RPOR14		
F4Dh	DSCONH	F2Dh	—	F0Dh	CCPR6L	EEDh	—	ECDh	RPOR13		
F4Ch	DSCONL	F2Ch	—	F0Ch	CCP6CON	EECh	—	ECCh	RPOR12		
F4Bh	DSWAKEH	F2Bh	—	F0Bh	CCPR7H	EEBh	—	ECBh	RPOR11		
F4Ah	DSWAKEL	F2Ah	—	F0Ah	CCPR7L	EEAh	RPINR9	ECAh	RPOR10		
F49h	ANCON1	F29h	—	F09h	CCP7CON	EE9h	RPINR8	EC9h	RPOR9		
F48h	ANCON0	F28h	—	F08h	CCPR8H	EE8h	RPINR7	EC8h	RPOR8		
F47h	ALRMCFG	F27h	—	F07h	CCPR8L	EE7h	RPINR15	EC7h	RPOR7		
F46h	ALRMRPT	F26h	—	F06h	CCP8CON	EE6h	RPINR6	EC6h	RPOR6		
F45h	ALRMVALH	F25h	CM3CON	F05h	CCPR9H	EE5h	—	EC5h	RPOR5		
F44h	ALRMVALL	F24h	TMR5H	F04h	CCPR9L	EE4h	RPINR4	EC4h	RPOR4		
F43h	—	F23h	TMR5L	F03h	CCP9CON	EE3h	RPINR3	EC3h	RPOR3		
F42h	ODCON1	F22h	T5CON	F02h	CCPR10H	EE2h	RPINR2	EC2h	RPOR2		
F41h	ODCON2	F21h	T5GCON	F01h	CCPR10L	EE1h	RPINR1	EC1h	RPOR1		
F40h	ODCON3	F20h	TMR6	F00h	CCP10CON	EE0h	—	EC0h	RPOR0		

6.3.4.1 Context Defined SFRs

There are several registers that share the same address in the SFR space. The register's definition and usage depends on the operating mode of its associated peripheral. These registers are:

- SSPxADD and SSPxMSK: These are two separate hardware registers, accessed through a single SFR address. The operating mode of the MSSP modules determines which register is being accessed. See **Section 20.5.3.4 "7-Bit Address Masking Mode"** for additional details.
- PMADDRH/L and PMDOUT2H/L: In this case, these named buffer pairs are actually the same physical registers. The Parallel Master Port (PMP) module's operating mode determines what function the registers take on. See **Section 11.1.2 "Data Registers"** for additional details.

TABLE 6-4: REGISTER FILE SUMMARY (PIC18F47J13 FAMILY)

Addr.	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	—	SP4	SP3	SP2	SP1	SP0	00-0 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	—	—	—	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)								N/A
FEEh	POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								N/A
FEDh	POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								N/A
FECh	PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								N/A
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								N/A
FEAh	FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte				---- 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)								N/A
FE6h	POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								N/A
FE5h	POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								N/A
FE4h	PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								N/A
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								N/A
FE2h	FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte				---- 0000
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)								N/A
FDEh	POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								N/A
FDDh	POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								N/A
FDCh	PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								N/A

Note 1: Applicable for 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).
2: Applicable for 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: Value on POR, BOR.

TABLE 6-4: REGISTER FILE SUMMARY (PIC18F47J13 FAMILY) (CONTINUED)

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
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								N/A
FDAh	FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte				---- 0000
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	T0PS2	T0PS1	T0PS0	1111 1111
FD3h	OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	—	SCS1	SCS0	0110 q100
FD2h	CM1CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0	0001 1111
FD1h	CM2CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0	0001 1111
FD0h	RCON	IPEN	—	$\overline{\text{CM}}$	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	0-11 11qq
FCFh	TMR1H	Timer1 Register High Byte								xxxx xxxx
FCEh	TMR1L	Timer1 Register Low Bytes								xxxx xxxx
FCDh	T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	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	MSSP1 Receive Buffer/Transmit Register								xxxx xxxx
FC8h	SSP1ADD	MSSP1 Address Register (I^2C Slave mode). MSSP1 Baud Rate Reload Register (I^2C Master mode).								0000 0000
FC8h	SSP1MSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	1111 1111
FC7h	SSP1STAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	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
FC5h	SSP1CON2	GCEN	ACKSTAT	ADMSK5	ADMSK4	ADMSK3	ADMSK2	ADMSK1	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	VCFG1	VCFG0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	0000 0000
FC1h	ADCON1	ADFM	ADCAL	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0000 0000
FC0h	WDTCON	REGSLP	LVDSTAT	ULPLVL	VBGOE	DS	ULPEN	ULPSINK	SWDTEN	1qq0 q000
FBFh	PSTR1CON	CMPL1	CMPL0	—	STRSYNC	STRD	STRC	STRB	STRA	00-0 0001
FBEh	ECCP1AS	ECCP1ASE	ECCP1AS2	ECCP1AS1	ECCP1AS0	PSS1AC1	PSS1AC0	PSS1BD1	PSS1BD0	0000 0000
FBDh	ECCP1DEL	P1RSEN	P1DC6	P1DC5	P1DC4	P1DC3	P1DC2	P1DC1	P1DC0	0000 0000
FBCh	CCPR1H	Capture/Compare/PWM Register 1 High Byte								xxxx xxxx
FBBh	CCPR1L	Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx
FBAh	CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000
FB9h	PSTR2CON	CMPL1	CMPL0	—	STRSYNC	STRD	STRC	STRB	STRA	00-0 0001
FB8h	ECCP2AS	ECCP2ASE	ECCP2AS2	ECCP2AS1	ECCP2AS0	PSS2AC1	PSS2AC0	PSS2BD1	PSS2BD0	0000 0000
FB7h	ECCP2DEL	P2RSEN	P2DC6	P2DC5	P2DC4	P2DC3	P2DC2	P2DC1	P2DC0	0000 0000
FB6h	CCPR2H	Capture/Compare/PWM Register 2 High Byte								xxxx xxxx
FB5h	CCPR2L	Capture/Compare/PWM Register 2 Low Byte								xxxx xxxx
FB4h	CCP2CON	P2M1	P2M0	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	0000 0000
FB3h	CTMUCONH	CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG	0-00 0000
FB2h	CTMUCONL	EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT	0000 00xx
FB1h	CTMUICON	ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0	0000 0000
FB0h	SPBRG1	EUSART1 Baud Rate Generator Register Low Byte								0000 0000
FAFh	RCREG1	EUSART1 Receive Register								0000 0000
FAEh	TXREG1	EUSART1 Transmit Register								0000 0000
FADh	TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010

Note 1: Applicable for 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).
2: Applicable for 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: Value on POR, BOR.

TABLE 6-4: REGISTER FILE SUMMARY (PIC18F47J13 FAMILY) (CONTINUED)

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
FACh	RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 0000
FABh	SPBRG2	EUSART2 Baud Rate Generator Register Low Byte								0000 0000
FAAh	RCREG2	EUSART2 Receive Register								0000 0000
FA9h	TXREG2	EUSART2 Transmit Register								0000 0000
FA8h	TXSTA2	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010
FA7h	EECON2	Flash Self-Program Control Register (not a physical register)								---- ----
FA6h	EECON1	—	—	WPROG	FREE	WRERR	WREN	WR	—	--00 x00-
FA5h	IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP	1111 1111
FA4h	PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF	0000 0000
FA3h	PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE	0000 0000
FA2h	IPR2	OSCFIP	CM2IP	CM1IP	—	BCL1IP	HLVDIP	TMR3IP	CCP2IP	111- 1111
FA1h	PIR2	OSCFIF	CM2IF	CM1IF	—	BCL1IF	HLVDIF	TMR3IF	CCP2IF	000- 0000
FA0h	PIE2	OSCFIE	CM2IE	CM1IE	—	BCL1IE	HLVDIE	TMR3IE	CCP2IE	000- 0000
F9Fh	IPR1	PMPIP(2)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	1111 1111
F9Eh	PIR1	PMPIF(2)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	0000 0000
F9Dh	PIE1	PMPIE(2)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	0000 0000
F9Ch	RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 0000
F9Bh	OSCTUNE	INTSRC	PLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0	0000 0000
F9Ah	T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/$\overline{\text{T1DONE}}$	T1GVAL	T1GSS1	T1GSS0	0000 0x00
F99h	IPR5	—	—	CM3IP	TMR8IP	TMR6IP	TMR5IP	TMR5GIP	TMR1GIP	--11 1111
F98h	PIR5	—	—	CM3IF	TMR8IF	TMR6IF	TMR5IF	TMR5GIF	TMR1GIF	--00 0000
F97h	T3GCON	TMR3GE	T3GPOL	T3GTM	T3GSPM	T3GGO/$\overline{\text{T3DONE}}$	T3GVAL	T3GSS1	T3GSS0	0000 0x00
F96h	TRISE(2)	$\overline{\text{RDPU}}$	$\overline{\text{REPU}}$	—	—	—	TRISE2	TRISE1	TRISE0	00-- -111
F95h	TRISD(2)	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	—	TRISA3	TRISA2	TRISA1	TRISA0	111- 1111
F91h	PIE5	—	—	CM3IE	TMR8IE	TMR6IE	TMR5IE	TMR5GIE	TMR1GIE	--00 0000
F90h	IPR4	CCP10IP	CCP9IP	CCP8IP	CCP7IP	CCP6IP	CCP5IP	CCP4IP	CCP3IP	1111 1111
F8Fh	PIR4	CCP10IF	CCP9IF	CCP8IF	CCP7IF	CCP6IF	CCP5IF	CCP4IF	CCP3IF	0000 0000
F8Eh	PIE4	CCP10IE	CCP9IE	CCP8IE	CCP7IE	CCP6IE	CCP5IE	CCP4IE	CCP3IE	0000 0000
F8Dh	LATE(2)	—	—	—	—	—	LATE2	LATE1	LATE0	---- -xxx
F8Ch	LATD(2)	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	—	LATA3	LATA2	LATA1	LATA0	xxx- xxxx
F88h	DMACON1	SSCON1	SSCON0	TXINC	RXINC	DUPLEX1	DUPLEX0	DLYINTEN	DMAEN	0000 0000
F87h	OSCCON2	—	SOSCRUN	—	SOSCDRV	SOSCGO	PRISD	—	—	-0-1 01--
F86h	DMACON2	DLYCYC3	DLYCYC2	DLYCYC1	DLYCYC0	INTLVL3	INTLVL2	INTLVL1	INTLVL0	0000 0000
F85h	HLVDCON	VDIRMAG	BGVST	IRVST	HLVDEN	HLVDL3	HLVDL2	HLVDL1	HLVDL0	0000 0000
F84h	PORTE(2)	—	—	—	—	—	RE2	RE1	RE0	---- -xxx
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	—	RA3	RA2	RA1	RA0	xxx- xxxx
F7Fh	SPBRGH1	EUSART1 Baud Rate Generator High Byte								0000 0000
F7Eh	BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	0100 0-00

Note 1: Applicable for 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).
2: Applicable for 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: Value on POR, BOR.

TABLE 6-4: REGISTER FILE SUMMARY (PIC18F47J13 FAMILY) (CONTINUED)

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F7Dh	SPBRGH2	EUSART2 Baud Rate Generator High Byte								0000 0000
F7Ch	BAUDCON2	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	0100 0-00
F7Bh	TMR3H	Timer3 Register High Byte								xxxx xxxx
F7Ah	TMR3L	Timer3 Register Low Byte								xxxx xxxx
F79h	T3CON	TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	T3OSCEN	$\overline{\text{T3SYNC}}$	RD16	TMR3ON	0000 0000
F78h	TMR4	Timer4 Register								0000 0000
F77h	PR4	Timer4 Period Register								1111 1111
F76H	T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0	-000 0000
F75h	SSP2BUF	MSSP2 Receive Buffer/Transmit Register								xxxx xxxx
F74h	SSP2ADD	MSSP2 Address Register (I^{2}C Slave mode)/MSSP2 Baud Rate Reload Register (I^{2}C Master mode)								0000 0000
F74h	SSP2MSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	1111 1111
F73h	SSP2STAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000
F72h	SSP2CON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000
F71h	SSP2CON2	GCEN	ACKSTAT	ACKDT ADMSK5	ACKEN ADMSK4	RCEN ADMSK3	PEN ADMSK2	RSEN ADMSK1	SEN	0000 0000
F70h	CMSTAT	—	—	—	—	—	COUT3	COUT2	COUT1	---- -111
F6Fh	PMADDRH/ PMDOUT1H(2)	—	CS1	Parallel Master Port Address High Byte						-000 0000
		Parallel Port Out Data High Byte (Buffer 1)								0000 0000
F6Eh	PMADDRL/ PMDOUT1L(2)	Parallel Master Port Address Low Byte/ Parallel Port Out Data Low Byte (Buffer 1)								0000 0000
F6Dh	PMDIN1H(2)	Parallel Port In Data High Byte (Buffer 1)								0000 0000
F6Ch	PMDIN1L(2)	Parallel Port In Data Low Byte (Buffer 1)								0000 0000
F6Bh	TXADDRL	SPI DMA Transmit Data Pointer Low Byte								0000 0000
F6Ah	TXADDRH	—	—	—	—	SPI DMA Transmit Data Pointer High Byte				---- 0000
F69h	RXADDRL	SPI DMA Receive Data Pointer Low Byte								0000 0000
F68h	RXADDRH	—	—	—	—	SPI DMA Receive Data Pointer High Byte				---- 0000
F67h	DMABCL	SPI DMA Byte Count Low Byte								0000 0000
F66h	DMABCH	—	—	—	—	—	—	SPI DMA Byte Count High Byte		---- --00
F5Fh	PMCONH(2)	PMPEN	—	—	ADRMUX1	ADRMUX0	PTBEEN	PTWREN	PTRDEN	0--0 0000
F5Eh	PMCONL(2)	CSF1	CSF0	ALP	—	CS1P	BEP	WRSP	RDSP	000- 0000
F5Dh	PMMODEH(2)	BUSY	IRQM1	IRQM0	INCM1	INCM0	MODE16	MODE1	MODE0	0000 0000
F5Ch	PMMODEL(2)	WAITB1	WAITB0	WAITM3	WAITM2	WAITM1	WAITM0	WAITE1	WAITE0	0000 0000
F5Bh	PMDOUT2H(2)	Parallel Port Out Data High Byte (Buffer 2)								0000 0000
F5Ah	PMDOUT2L(2)	Parallel Port Out Data Low Byte (Buffer 2)								0000 0000
F59h	PMDIN2H(2)	Parallel Port In Data High Byte (Buffer 2)								0000 0000
F58h	PMDIN2L(2)	Parallel Port In Data Low Byte (Buffer 2)								0000 0000
F57h	PMEH(2)	PTEN15	PTEN14	PTEN13	PTEN12	PTEN11	PTEN10	PTEN9	PTEN8	0000 0000
F56h	PMEL(2)	PTEN7	PTEN6	PTEN5	PTEN4	PTEN3	PTEN2	PTEN1	PTEN0	0000 0000
F55h	PMSTATH(2)	IBF	IBOV	—	—	IB3F	IB2F	IB1F	IB0F	00-- 0000
F54h	PMSTATL(2)	OBE	OBUF	—	—	OB3E	OB2E	OB1E	OB0E	10-- 1111
F53h	CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000
F52h	CCPTMRS0	C3TSEL1	C3TSEL0	C2TSEL2	C2TSEL1	C2TSEL0	C1TSEL2	C1TSEL1	C1TSEL0	0000 0000
F51h	CCPTMRS1	C7TSEL1	C7TSEL0	—	C6TSEL0	—	C5TSEL0	C4TSEL1	C4TSEL0	00-0 -000
F50h	CCPTMRS2	—	—	—	C10TSEL0	—	C9TSEL0	C8TSEL1	C8TSEL0	---0 -000
F4Fh	DSGPR1	Deep Sleep Persistent General Purpose Register (contents retained even in Deep Sleep)								xxxx xxxx
F4Eh	DSGPR0	Deep Sleep Persistent General Purpose Register (contents retained even in Deep Sleep)								xxxx xxxx
F4Dh	DSCONH	DSEN	—	—	—	—	CFGPER2	DSULPEN	RTCWDIS	0--- -000
F4Ch	DSCONL	—	—	—	—	—	ULPWDIS	DSBOR	RELEASE	---- -000

Note 1: Applicable for 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).
2: Applicable for 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: Value on POR, BOR.

TABLE 6-4: REGISTER FILE SUMMARY (PIC18F47J13 FAMILY) (CONTINUED)

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F4Bh	DSWAKEH	—	—	—	—	—	—	—	DSINT0	---- ---0
F4Ah	DSWAKEL	DSFLT	—	DSULP	DSWDT	DSRTC	DSMCLR	—	DSPOR	0-00 00-1
F49h	ANCON1	VBGEN	—	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	0--0 0000
F48h	ANCON0	PCFG7(2)	PCFG6(2)	PCFG5(2)	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000
F47h	ALRMCFG	ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0	0000 0000
F46h	ALRMRPT	ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0	0000 0000
F45h	ALRMVALH	Alarm Value High Register Window based on ALRMPTR<1:0>								xxxx xxxx
F44h	ALRMVALL	Alarm Value Low Register Window based on ALRMPTR<1:0>								xxxx xxxx
F43h	—	—	—	—	—	—	—	—	—	---- ----
F42h	ODCON1	CCP8OD	CCP7OD	CCP6OD	CCP5OD	CCP4OD	ECCP3OD	ECCP2OD	ECCP1OD	0000 0000
F41h	ODCON2	—	—	—	—	CCP10OD	CCP9OD	U2OD	U1OD	---- 0000
F40h	ODCON3	CTMUDS	—	—	—	—	—	SPI2OD	SPI1OD	0--- --00
F3Fh	RTCCFG	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0	0-00 0000
F3Eh	RTCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	0000 0000
F3Dh	REFOCON	ROON	—	ROSSLP	ROSEL	RODIV3	RODIV2	RODIV1	RODIV0	0-00 0000
F3Ch	PADCFG1	—	—	—	—	—	RTSECSEL1	RTSECSEL0	PMPTTL(2)	---- -000
F3Bh	RTCVALH	RTCC Value High Register Window based on RTCPTR<1:0>								0xxx xxxx
F3Ah	RTCVALL	RTCC Value Low Register Window based on RTCPTR<1:0>								0xxx xxxx
F25h	CM3CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0	0001 1111
F24h	TMR5H	Timer5 Register High Byte								xxxx xxxx
F23h	TMR5L	Timer5 Register Low Bytes								xxxx xxxx
F22h	T5CON	TMR5CS1	TMR5CS0	T5CKPS1	T5CKPS0	T5OSCEN	$\overline{\text{T5SYNC}}$	RD16	TMR5ON	0000 0000
F21h	T5GCON	TMR5GE	T5GPOL	T5GTM	T5GSPM	T5GGO/$\overline{\text{T5DONE}}$	T5GVAL	T5GSS1	T5GSS0	0000 0x00
F20h	TMR6	Timer6 Register								0000 0000
F1Fh	PR6	Timer6 Period Register								1111 1111
F1Eh	T6CON	—	T6OUTPS3	T6OUTPS2	T6OUTPS1	T6OUTPS0	TMR6ON	T6CKPS1	T6CKPS0	-000 0000
F1Dh	TMR8	Timer8 Register								0000 0000
F1Ch	PR8	Timer8 Period Register								1111 1111
F1Bh	T8CON	—	T8OUTPS3	T8OUTPS2	T8OUTPS1	T8OUTPS0	TMR8ON	T8CKPS1	T8CKPS0	-000 0000
F1Ah	PSTR3CON	CMPL1	CMPL0	—	STRSYNC	STRD	STRC	STRB	STRA	00-0 0001
F19h	ECCP3AS	ECCP3ASE	ECCP3AS2	ECCP3AS1	ECCP3AS0	PSS3AC1	PSS3AC0	PSS3BD1	PSS3BD0	0000 0000
F18h	ECCP3DEL	P3RSEN	P3DC6	P3DC5	P3DC4	P3DC3	P3DC2	P3DC1	P3DC0	0000 0000
F17h	CCPR3H	Capture/Compare/PWM Register 3 High Byte								xxxx xxxx
F16h	CCPR3L	Capture/Compare/PWM Register 3 Low Byte								xxxx xxxx
F15h	CCP3CON	P3M1	P3M0	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0	0000 0000
F14h	CCPR4H	Capture/Compare/PWM Register 4 High Byte								xxxx xxxx
F13h	CCPR4L	Capture/Compare/PWM Register 4 Low Byte								xxxx xxxx
F12h	CCP4CON	—	—	DC4B1	DC4B0	CCP4M3	CCP4M2	CCP4M1	CCP4M0	--00 0000
F11h	CCPR5H	Capture/Compare/PWM Register 5 High Byte								xxxx xxxx
F10h	CCPR5L	Capture/Compare/PWM Register 5 Low Byte								xxxx xxxx
F0Fh	CCP5CON	—	—	DC5B1	DC5B0	CCP5M3	CCP5M2	CCP5M1	CCP5M0	--00 0000
F0Eh	CCPR6H	Capture/Compare/PWM Register 6 High Byte								xxxx xxxx
F0Dh	CCPR6L	Capture/Compare/PWM Register 6 Low Byte								xxxx xxxx
F0Ch	CCP6CON	—	—	DC6B1	DC6B0	CCP6M3	CCP6M2	CCP6M1	CCP6M0	--00 0000
F0Bh	CCPR7H	Capture/Compare/PWM Register 7 High Byte								xxxx xxxx
F0Ah	CCPR7L	Capture/Compare/PWM Register 7 Low Byte								xxxx xxxx
F09h	CCP7CON	—	—	DC7B1	DC7B0	CCP7M3	CCP7M2	CCP7M1	CCP7M0	--00 0000
F08h	CCPR8H	Capture/Compare/PWM Register 8 High Byte								xxxx xxxx

Note 1: Applicable for 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).
2: Applicable for 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: Value on POR, BOR.

TABLE 6-4: REGISTER FILE SUMMARY (PIC18F47J13 FAMILY) (CONTINUED)

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F07h	CCPR8L	Capture/Compare/PWM Register 8 Low Byte								xxxx xxxx
F06h	CCP8CON	—	—	DC8B1	DC8B0	CCP8M3	CCP8M2	CCP8M1	CCP8M0	--00 0000
F05h	CCPR9H	Capture/Compare/PWM Register 9 High Byte								xxxx xxxx
F04h	CCPR9L	Capture/Compare/PWM Register 9 Low Byte								xxxx xxxx
F03h	CCP9CON	—	—	DC9B1	DC9B0	CCP9M3	CCP9M2	CCP9M1	CCP9M0	--00 0000
F02h	CCPR10H	Capture/Compare/PWM Register 10 High Byte								xxxx xxxx
F01h	CCPR10L	Capture/Compare/PWM Register 10 Low Byte								xxxx xxxx
F00h	CCP10CON	—	—	DC10B1	DC10B0	CCP10M3	CCP10M2	CCP10M1	CCP10M0	--00 0000
EFFh	RPINR24	—	—	—	PWM Fault Input (FLT0) to Input Pin Mapping bits					---1 1111
EFEh	RPINR23	—	—	—	SPI2 Slave Select Input ($\overline{SS2}$) to Input Pin Mapping bits					---1 1111
EFDh	RPINR22	—	—	—	SPI2 Clock Input (SCK2) to Input Pin Mapping bits					---1 1111
EFCh	RPINR21	—	—	—	SPI2 Data Input (SDI2) to Input Pin Mapping bits					---1 1111
EFBh	—	—	—	—	—	—	—	—	—	**(3)**
EFAh	—	—	—	—	—	—	—	—	—	**(3)**
EF9h	—	—	—	—	—	—	—	—	—	**(3)**
EF8h	RPINR17	—	—	—	EUSART2 Clock Input (CK2) to Input Pin Mapping bits					---1 1111
EF7h	RPINR16	—	—	—	EUSART2 RX2/DT2 to Input Pin Mapping bits					---1 1111
EF6h	—	—	—	—	—	—	—	—	—	**(3)**
EF5h	—	—	—	—	—	—	—	—	—	**(3)**
EF4h	RPINR14	—	—	—	Timer5 Gate Input (T5G) to Input Pin Mapping bits					---1 1111
EF3h	RPINR13	—	—	—	Timer3 Gate Input (T3G) to Input Pin Mapping bits					---1 1111
EF2h	RPINR12	—	—	—	Timer1 Gate Input (T1G) to Input Pin Mapping bits					---1 1111
EF1h	—	—	—	—	—	—	—	—	—	**(3)**
EF0h	—	—	—	—	—	—	—	—	—	**(3)**
EEFh	—	—	—	—	—	—	—	—	—	**(3)**
EEEh	—	—	—	—	—	—	—	—	—	**(3)**
EEDh	—	—	—	—	—	—	—	—	—	**(3)**
EECh	—	—	—	—	—	—	—	—	—	**(3)**
EEBh	—	—	—	—	—	—	—	—	—	**(3)**
EEAh	RPINR9	—	—	—	ECCP3 Input Capture (IC3) to Input Pin Mapping bits					---1 1111
EE9h	RPINR8	—	—	—	ECCP2 Input Capture (IC2) to Input Pin Mapping bits					---1 1111
EE8h	RPINR7	—	—	—	ECCP1 Input Capture (IC1) to Input Pin Mapping bits					---1 1111
EE7h	RPINR15	—	—	—	Timer5 External Clock Input (T5CKI) to Input Pin Mapping bits					---1 1111
EE6h	RPINR6	—	—	—	Timer3 External Clock Input (T3CKI) to Input Pin Mapping bits					---1 1111
EE5h	—	—	—	—	—	—	—	—	—	**(3)**
EE4h	RPINR4	—	—	—	Timer0 External Clock Input (T0CKI) to Input Pin Mapping bits					---1 1111
EE3h	RPINR3	—	—	—	External Interrupt (INT3) to Input Pin Mapping bits					---1 1111
EE2h	RPINR2	—	—	—	External Interrupt (INT2) to Input Pin Mapping bits					---1 1111
EE1h	RPINR1	—	—	—	External Interrupt (INT1) to Input Pin Mapping bits					---1 1111
EE0h	—	—	—	—	—	—	—	—	—	**(3)**
EDFh	—	—	—	—	—	—	—	—	—	**(3)**
EDEh	—	—	—	—	—	—	—	—	—	**(3)**
EDDh	—	—	—	—	—	—	—	—	—	**(3)**
EDCh	—	—	—	—	—	—	—	—	—	**(3)**
EDBh	—	—	—	—	—	—	—	—	—	**(3)**
EDAh	—	—	—	—	—	—	—	—	—	**(3)**
ED9h	—	—	—	—	—	—	—	—	—	**(3)**
ED8h	RPOR24(2)	—	—	—	Remappable Pin RP24 Output Signal Select bits					---0 0000
ED7h	RPOR23(2)	—	—	—	Remappable Pin RP23 Output Signal Select bits					---0 0000

Note **1:** Applicable for 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).
2: Applicable for 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: Value on POR, BOR.

TABLE 6-4: REGISTER FILE SUMMARY (PIC18F47J13 FAMILY) (CONTINUED)

Addr.	File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
ED6h	RPOR22[(2)]	—	—	—	Remappable Pin RP22 Output Signal Select bits					---0 0000
ED5h	RPOR21[(2)]	—	—	—	Remappable Pin RP21 Output Signal Select bits					---0 0000
ED4h	RPOR20[(2)]	—	—	—	Remappable Pin RP20 Output Signal Select bits					---0 0000
ED3h	RPOR19[(2)]	—	—	—	Remappable Pin RP19 Output Signal Select bits					---0 0000
ED2h	RPOR18	—	—	—	Remappable Pin RP18 Output Signal Select bits					---0 0000
ED1h	RPOR17	—	—	—	Remappable Pin RP17 Output Signal Select bits					---0 0000
ED0h	RPOR16	—	—	—	Remappable Pin RP16 Output Signal Select bits					---0 0000
ECFh	RPOR15	—	—	—	Remappable Pin RP15 Output Signal Select bits					---0 0000
ECEh	RPOR14	—	—	—	Remappable Pin RP14 Output Signal Select bits					---0 0000
ECDh	RPOR13	—	—	—	Remappable Pin RP13 Output Signal Select bits					---0 0000
ECCh	RPOR12	—	—	—	Remappable Pin RP12 Output Signal Select bits					---0 0000
ECBh	RPOR11	—	—	—	Remappable Pin RP11 Output Signal Select bits					---0 0000
ECAh	RPOR10	—	—	—	Remappable Pin RP10 Output Signal Select bits					---0 0000
EC9h	RPOR9	—	—	—	Remappable Pin RP9 Output Signal Select bits					---0 0000
EC8h	RPOR8	—	—	—	Remappable Pin RP8 Output Signal Select bits					---0 0000
EC7h	RPOR7	—	—	—	Remappable Pin RP7 Output Signal Select bits					---0 0000
EC6h	RPOR6	—	—	—	Remappable Pin RP6 Output Signal Select bits					---0 0000
EC5h	RPOR5	—	—	—	Remappable Pin RP5 Output Signal Select bits					---0 0000
EC4h	RPOR4	—	—	—	Remappable Pin RP4 Output Signal Select bits					---0 0000
EC3h	RPOR3	—	—	—	Remappable Pin RP3 Output Signal Select bits					---0 0000
EC2h	RPOR2	—	—	—	Remappable Pin RP2 Output Signal Select bits					---0 0000
EC1h	RPOR1	—	—	—	Remappable Pin RP1 Output Signal Select bits					---0 0000
EC0h	RPOR0	—	—	—	Remappable Pin RP0 Output Signal Select bits					---0 0000
EBFh	PPSCON	—	—	—	—	—	—	—	IOLOCK	---- ---0
EBEh	—	—	—	—	—	—	—	—	—	**(3)**
EBDh	—	—	—	—	—	—	—	—	—	**(3)**
EBCh	PMDIS3	CCP10MD	CCP9MD	CCP8MD	CCP7MD	CCP6MD	CCP5MD	CCP4MD	—	0000 000-
EBBh	PMDIS2	—	TMR8MD	—	TMR6MD	TMR5MD	CMP3MD	CMP2MD	CMP1MD	-0-0 0000
EBAh	PMDIS1	PSPMD[(2)]	CTMUMD	RTCCMD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	—	0000 000-
EB9h	PMDIS0	ECCP3MD	ECCP2MD	ECCP1MD	UART2MD	UART1MD	SPI2MD	SPI1MD	ADCMD	0000 0000
EB8h	ADCTRIG	—	—	—	—	—	—	SRC1	SRC0	---- --00
EB7h	—	—	—	—	—	—	—	—	—	**(3)**
EB6h	—	—	—	—	—	—	—	—	—	**(3)**
EB5h	—	—	—	—	—	—	—	—	—	**(3)**
EB4h	—	—	—	—	—	—	—	—	—	**(3)**
EB3h	—	—	—	—	—	—	—	—	—	**(3)**
EB2h	—	—	—	—	—	—	—	—	—	**(3)**
EB1h	—	—	—	—	—	—	—	—	—	**(3)**
EB0h	—	—	—	—	—	—	—	—	—	**(3)**

Note 1: Applicable for 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).
2: Applicable for 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).
3: Value on POR, BOR.

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, then the write to these five bits is disabled.

These bits are set or cleared according to the device logic. Therefore, the result of an instruction with the STATUS register as destination may be different than intended. For example, `CLRF STATUS` will set the Z bit but leave the other bits unchanged. The STATUS register then reads back as ‘`000u u1uu`’. It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF`, `MOVFF` and `MOVWF` instructions are used to alter the STATUS register because these instructions do not affect the Z, C, DC, OV or N bits in the STATUS register.

For other instructions not affecting any Status bits, see the instruction set summary in Table 28-2 and Table 28-3.

> **Note:** The C and DC bits operate as $\overline{\text{Borrow}}$ and $\overline{\text{Digit Borrow}}$ bits, respectively in subtraction.

REGISTER 6-2: STATUS REGISTER (ACCESS FDBh)

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC[1]	C[2]
bit 7							bit 0

Legend:

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

bit 7-5	**Unimplemented:** Read as ‘`0`’
bit 4	N: Negative bit This bit is used for signed arithmetic (2’s complement). It indicates whether the result was negative (ALU MSB = `1`). `1` = Result was negative `0` = Result was positive
bit 3	**OV:** Overflow bit This bit is used for signed arithmetic (2’s complement). It indicates an overflow of the 7-bit magnitude, which causes the sign bit (Bit 7) to change state. `1` = Overflow occurred for signed arithmetic (in this arithmetic operation) `0` = No overflow occurred
bit 2	**Z**: Zero bit `1` = The result of an arithmetic or logic operation is zero `0` = The result of an arithmetic or logic operation is not zero
bit 1	**DC:** Digit Carry/$\overline{\text{Digit Borrow}}$ 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 MSb of the result occurred `0` = No carry out from the MSb of the result occurred

Note 1: For $\overline{\text{Digit Borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2’s complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either bit 4 or bit 3 of the source register.

2: For $\overline{\text{Borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2’s complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low-order bit of the source register.

6.4 Data Addressing Modes

> **Note:** The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 6.6 “Data Memory and the Extended Instruction Set”** for more information.

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

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = `1`). Its operation is discussed in more detail in **Section 6.6.1 “Indexed Addressing with Literal Offset”**.

6.4.1 INHERENT AND LITERAL ADDRESSING

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

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

6.4.2 DIRECT ADDRESSING

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

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit literal address as their LSB. This address specifies either a register address in one of the banks of data RAM (**Section 6.3.3 “General Purpose Register File”**) or a location in the Access Bank (**Section 6.3.2 “Access Bank”**) as the data source for the instruction.

The Access RAM bit, ‘a’, determines how the address is interpreted. When ‘a’ is ‘`1`’, the contents of the BSR (**Section 6.3.1 “Bank Select Register”**) are used with the address to determine the complete 12-bit address of the register. When ‘a’ is ‘`0`’, the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

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

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

6.4.3 INDIRECT ADDRESSING

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations to be read or written to. Since the FSRs are themselves located in RAM as SFRs, 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, 0x100 ;
NEXT    CLRF    POSTINC0    ; Clear INDF
                            ; register then
                            ; inc pointer
        BTFSS   FSR0H, 1    ; All done with
                            ; Bank1?
        BRA     NEXT        ; NO, clear next
CONTINUE                    ; YES, continue
```

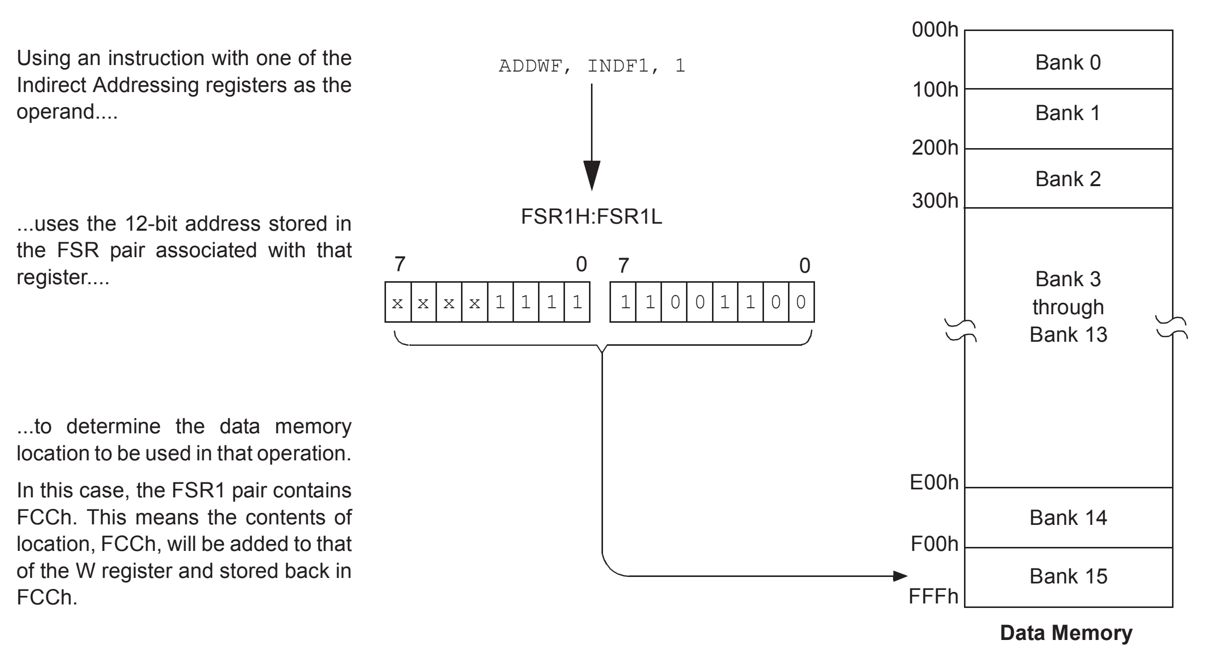
6.4.3.1 FSR Registers and the INDF Operand (INDF)

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

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

FIGURE 6-8: INDIRECT ADDRESSING


Using an instruction with one of the Indirect Addressing registers as the operand....
ADDWF, INDF1, 1
...uses the 12-bit address stored in the FSR pair associated with that register....
FSR1H:FSR1L
7
0
7
0
x x x x 1 1 1 1
1 1 0 0 1 1 0 0
...to determine the data memory location to be used in that operation.
In this case, the FSR1 pair contains FCCh. This means the contents of location, FCCh, will be added to that of the W register and stored back in FCCh.
000h
Bank 0
100h
Bank 1
200h
Bank 2
300h
Bank 3 through Bank 13
E00h
Bank 14
F00h
Bank 15
FFFh
Data Memory

6.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

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

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

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

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

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

6.4.3.3 Operations by FSRs on FSRs

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

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

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

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

6.5 Program Memory and the Extended Instruction Set

The operation of program memory is unaffected by the use of the extended instruction set.

Enabling the extended instruction set adds five additional two-word commands to the existing PIC18 instruction set: `ADDFSR`, `CALLW`, `MOVSF`, `MOVSS` and `SUBFSR`. These instructions are executed as described in **Section 6.2.4 “Two-Word Instructions”**.

6.6 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (XINST Configuration bit = `1`) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different. This is due to the introduction of a new addressing mode for the data memory space. This mode also alters the behavior of Indirect Addressing using FSR2 and its associated operands.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect Addressing mode; inherent and literal instructions do not change at all. Indirect Addressing with FSR0 and FSR1 also remains unchanged.

6.6.1 INDEXED ADDRESSING WITH LITERAL OFFSET

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

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

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

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

6.6.2 INSTRUCTIONS AFFECTED BY INDEXED LITERAL OFFSET MODE

Any of the core PIC18 instructions that can use Direct Addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte 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 and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is ‘`1`’) or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is provided in Figure 6-9.

Those who desire to use byte 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 28.2.1 “Extended Instruction Syntax”**.

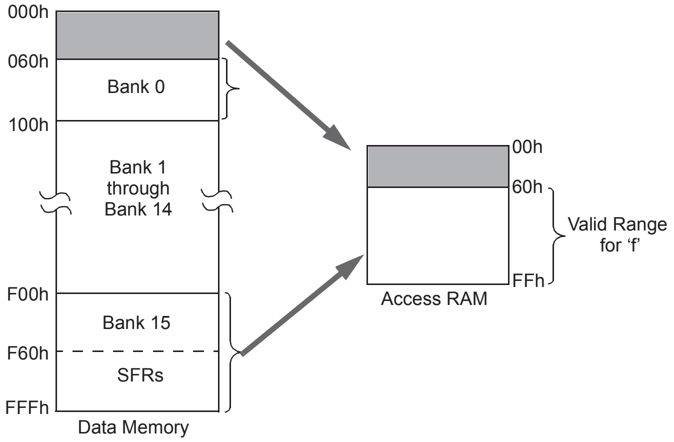
FIGURE 6-9: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

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.


000h
060h
100h
F00h
F60h
FFFh
Bank 0
Bank 1 through Bank 14
Bank 15
SFRs
Data Memory
00h
60h
FFh
Valid Range for ‘f’
Access RAM

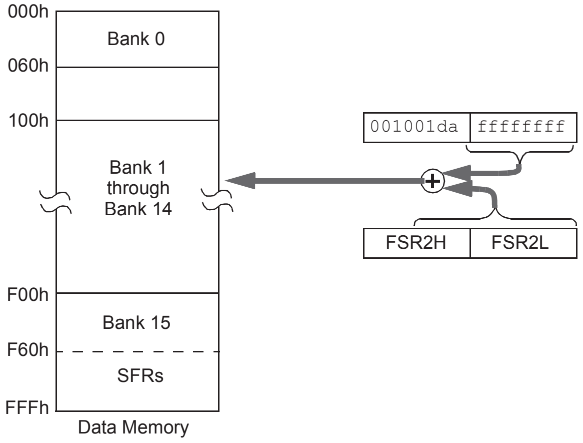
When a = `0` and f ≤ 5Fh:

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:

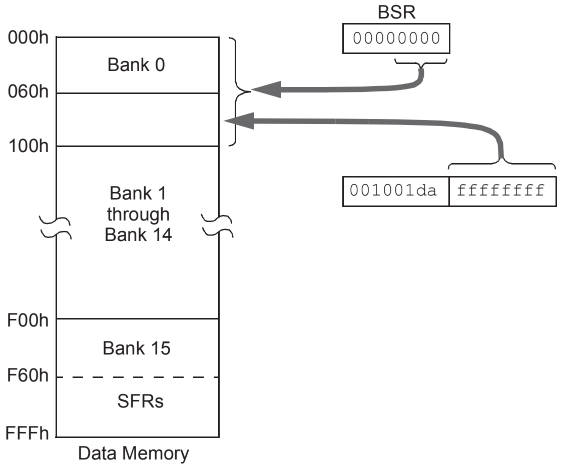
`ADDWF [k], d`

where ‘k’ is the same as ‘f’.


000h
060h
100h
F00h
F60h
FFFh
Bank 0
Bank 1 through Bank 14
Bank 15
SFRs
Data Memory
001001da
ffffffff
FSR2H
FSR2L

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.


BSR
00000000
000h
060h
100h
F00h
F60h
FFFh
Bank 0
Bank 1 through Bank 14
Bank 15
SFRs
Data Memory
001001da
ffffffff

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 to 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”**). Figure 6-10 provides an example of Access Bank remapping in this addressing mode.

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

6.6.4 BSR IN INDEXED LITERAL OFFSET MODE

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

FIGURE 6-10: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING

Example Situation:

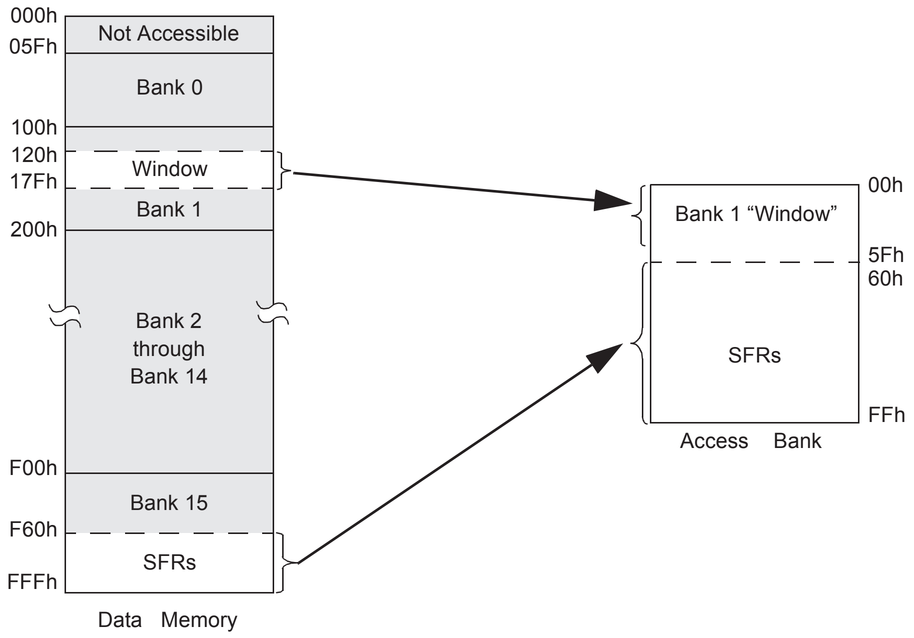
`ADDWF f, d, a`

FSR2H:FSR2L = 120h

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

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

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



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 1 byte at a time. A write to program memory is executed on blocks of 64 bytes at a time or 2 bytes at a time. Program memory is erased in blocks of 1024 bytes at a time. A bulk erase operation may not be issued from user code.

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

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

7.1 Table Reads and Table Writes

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

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

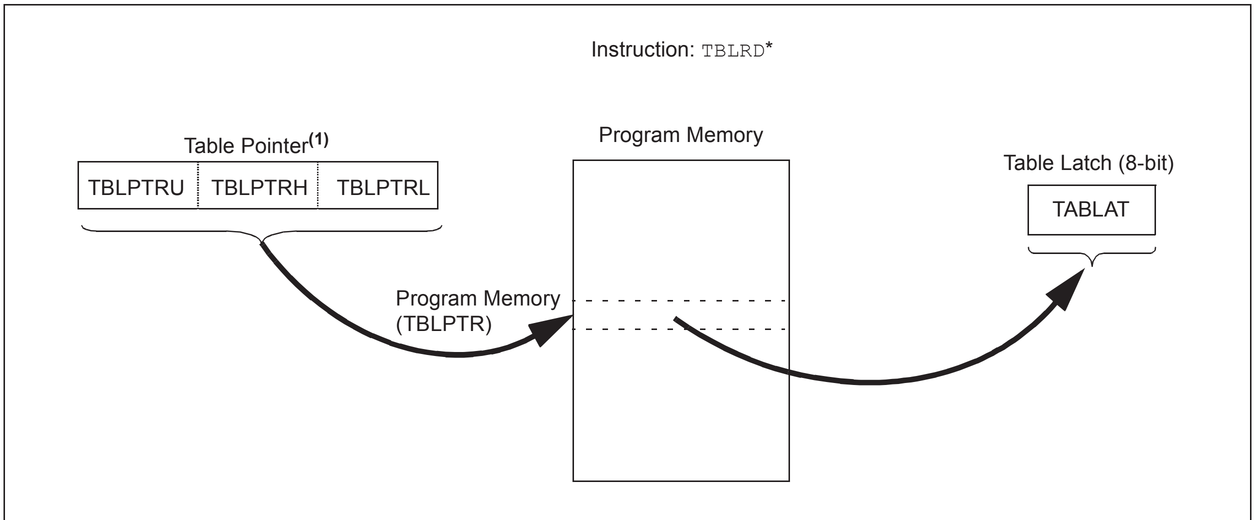
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 7-1 illustrates 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 illustrates the operation of a table write with program memory and data RAM.

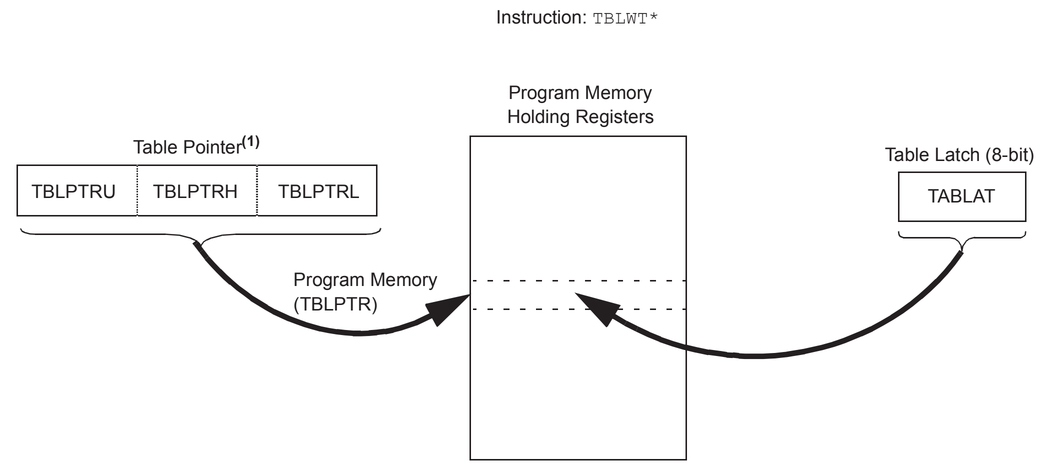
Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word-aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions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

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

7.2.1 EECON1 AND EECON2 REGISTERS

The EECON1 register (Register 7-1) is the control register for memory accesses. The EECON2 register is not a physical register; it is used exclusively in the memory write and erase sequences. Reading EECON2 will read all '`0`'s.

The WPROG bit, when set, will allow programming two bytes per word on the execution of the WR command. If this bit is cleared, the WR command will result in programming on a block of 64 bytes.

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

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

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

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

REGISTER 7-1: EECON1: EEPROM CONTROL REGISTER 1 (ACCESS FA6h)

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

Legend:	S = Settable bit (cannot be cleared in software)		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

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

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

bit 4 **FREE:** Flash Erase Enable bit
1 = Perform an erase operation on the next WR command (cleared by hardware after completion of erase)
0 = Perform write-only

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

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

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

bit 0 **Unimplemented:** Read as ‘0’

7.2.2 TABLE LATCH REGISTER (TABLAT)

The Table Latch (TABLAT) is an 8-bit register mapped into the Special Function Register (SFR) space. The Table Latch register is used to hold 8-bit data during data transfers between program memory and data RAM.

7.2.3 TABLE POINTER REGISTER (TBLPTR)

The Table Pointer (TBLPTR) register addresses a byte within the program memory. The TBLPTR comprises 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.

Table 7-1 displays these operations. On the TBLPTRT, these operations only affect the low-order 21 bits.

7.2.4 TABLE POINTER BOUNDARIES

TBLPTR is used in reads, writes and erases of the Flash program memory.

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

When a `TBLWT` is executed, the seven Least Significant bits (LSbs) of the Table Pointer register (TBLPTR<6:0>) determine which of the 64 program memory holding registers is written to. When the timed write to program memory begins (via the WR bit), the 12 Most Significant bits (MSbs) of the TBLPTR (TBLPTR<21:10>) determine which program memory block of 1024 bytes is written to. For more information, see **Section 7.5 “Writing to Flash Program Memory”**.

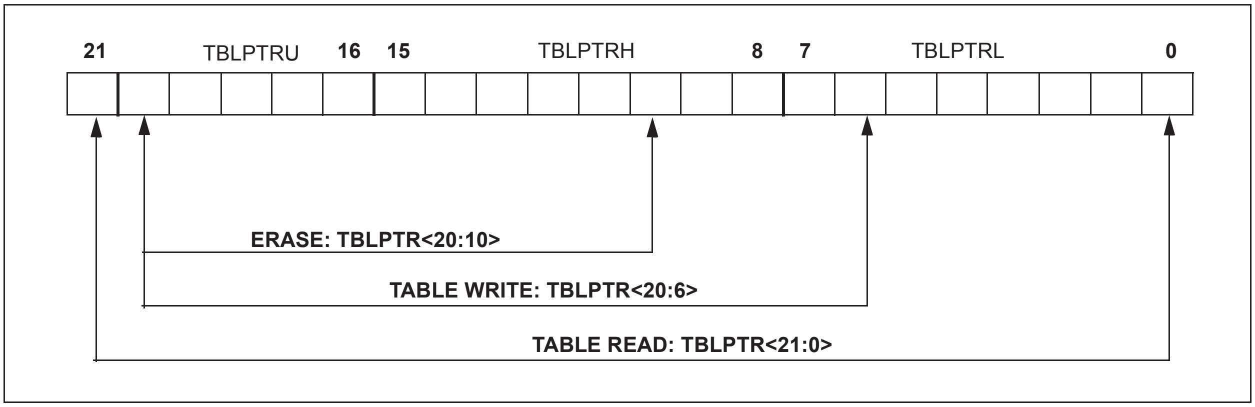
When an erase of program memory is executed, the 12 MSbs of the Table Pointer register point to the 1024-byte block that will be erased. The LSbs are ignored.

Figure 7-3 illustrates 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*` `TBLWT*`	TBLPTR is not modified
`TBLRD*+` `TBLWT*+`	TBLPTR is incremented after the read/write
`TBLRD*-` `TBLWT*-`	TBLPTR is decremented after the read/write
`TBLRD+*` `TBLWT+*`	TBLPTR is incremented before the read/write

FIGURE 7-3: TABLE POINTER BOUNDARIES BASED ON OPERATION



7.3 Reading the Flash Program Memory

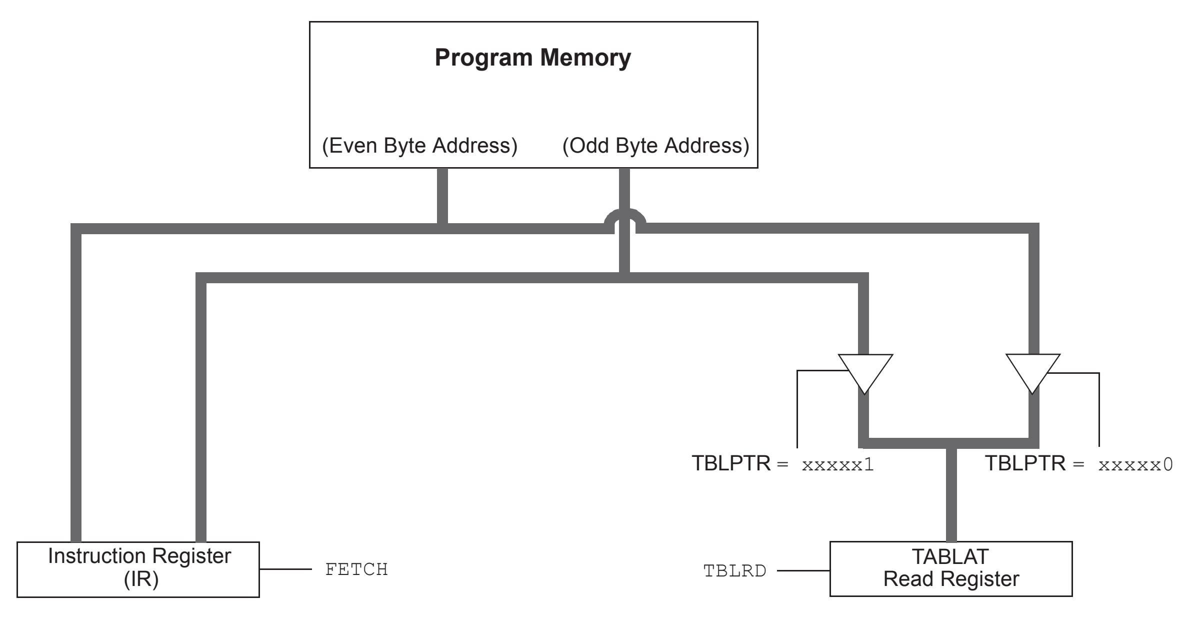
The `TBLRD` instruction is used to retrieve data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

The 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 LSb of the address selects between the high and low bytes of the word.

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

FIGURE 7-4: READS FROM FLASH PROGRAM MEMORY


Program Memory
(Even Byte Address)
(Odd Byte Address)
TBLPTR = xxxxx1
TBLPTR = xxxxx0
Instruction Register (IR)
FETCH
TBLRD
TABLAT Read Register

EXAMPLE 7-1: READING A FLASH PROGRAM MEMORY WORD

```
           MOVLW   CODE_ADDR_UPPER      ; Load TBLPTR with the base
           MOVWF   TBLPTRU              ; address of the word
           MOVLW   CODE_ADDR_HIGH
           MOVWF   TBLPTRH
           MOVLW   CODE_ADDR_LOW
           MOVWF   TBLPTRL
READ_WORD
           TBLRD*+                      ; read into TABLAT and increment
           MOVF    TABLAT, W            ; get data
           MOVWF   WORD_EVEN
           TBLRD*+                      ; read into TABLAT and increment
           MOVF    TABLAT, W            ; get data
           MOVWF   WORD_ODD
```

7.4 Erasing Flash Program Memory

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

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

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

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

7.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

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

1. Load Table Pointer register with address of row being erased.
2. Set the WREN and FREE bits (EECON1<2,4>) to enable the erase operation.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit; this will begin the erase cycle.
7. The CPU will stall for the duration of the erase for TIE (see parameter D133B).
8. Re-enable interrupts.

EXAMPLE 7-2: ERASING FLASH PROGRAM MEMORY

```
                MOVLW   CODE_ADDR_UPPER        ; load TBLPTR with the base
                MOVWF   TBLPTRU                ; address of the memory block
                MOVLW   CODE_ADDR_HIGH
                MOVWF   TBLPTRH
                MOVLW   CODE_ADDR_LOW
                MOVWF   TBLPTRL
     ERASE_ROW
                BSF     EECON1, WREN           ; enable write to memory
                BSF     EECON1, FREE           ; enable 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
```

7.5 Writing to Flash Program Memory

The programming block is 32 words or 64 bytes. Programming one word or 2 bytes at a time is also supported.

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

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

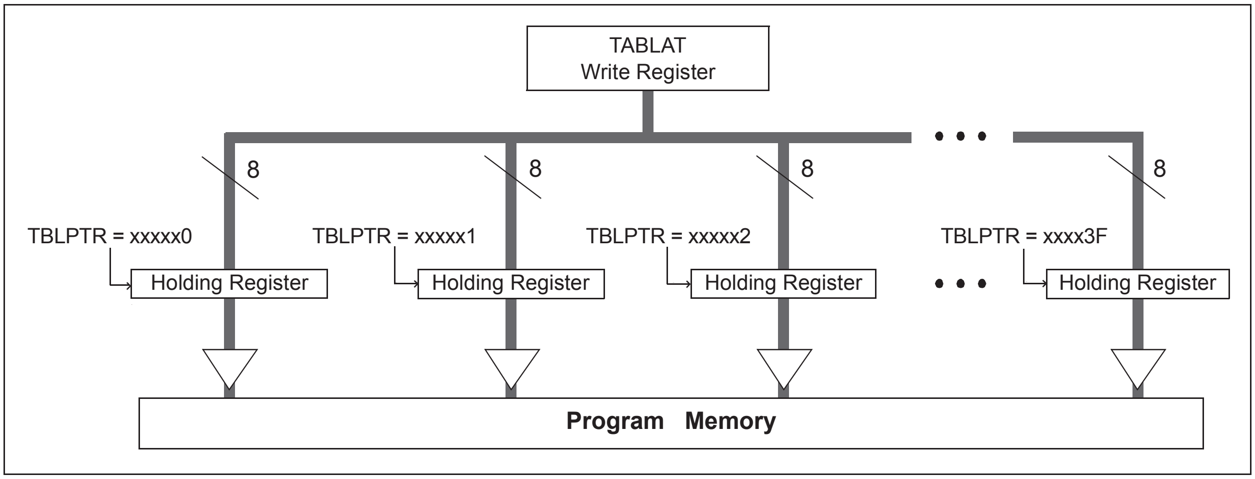
The long write is necessary for programming the internal Flash. Instruction execution is Halted while in a long write cycle. The long write will be terminated by the internal programming timer.

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

Note 1: Unlike previous PIC® devices, devices of the PIC18F47J13 Family do not reset the holding registers after a write occurs. The holding registers must be cleared or overwritten before a programming sequence.

2: To maintain the endurance of the program memory cells, each Flash byte should not be programmed more than once between erase operations. Before attempting to modify the contents of the target cell a second time, an erase of the target page, or a bulk erase of the entire memory, must be performed.

FIGURE 7-5: TABLE WRITES TO FLASH PROGRAM MEMORY



7.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

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

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

An example of the required code is provided in Example 7-3 on the following page.

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

EXAMPLE 7-3: WRITING TO FLASH PROGRAM MEMORY

```
                   MOVLW  CODE_ADDR_UPPER        ; Load TBLPTR with the base address
                   MOVWF  TBLPTRU                ; of the memory block, minus 1
                   MOVLW  CODE_ADDR_HIGH
                   MOVWF  TBLPTRH
                   MOVLW  CODE_ADDR_LOW
                   MOVWF  TBLPTRL
ERASE_BLOCK
                   BSF    EECON1, WREN           ; enable write to memory
                   BSF    EECON1, FREE           ; enable Erase operation
                   BCF    INTCON, GIE            ; disable interrupts
                   MOVLW  0x55
                   MOVWF  EECON2                 ; write 55h
                   MOVLW  0xAA
                   MOVWF  EECON2                 ; write 0AAh
                   BSF    EECON1, WR             ; start erase (CPU stall)
                   BSF    INTCON, GIE            ; re-enable interrupts
                   MOVLW  D'16'
                   MOVWF  WRITE_COUNTER          ; Need to write 16 blocks of 64 to write
                                                 ; one erase block of 1024
RESTART_BUFFER
                   MOVLW  D'64'
                   MOVWF  COUNTER
                   MOVLW  BUFFER_ADDR_HIGH       ; point to buffer
                   MOVWF  FSR0H
                   MOVLW  BUFFER_ADDR_LOW
                   MOVWF  FSR0L
FILL_BUFFER
                   ...                           ; read the new data from I2C, SPI,
                                                 ; PSP, USART, etc.
WRITE_BUFFER
                   MOVLW  D'64'                  ; number of bytes in holding register
                   MOVWF  COUNTER
WRITE_BYTE_TO_HREGS
                   MOVFF  POSTINC0, WREG         ; get low byte of buffer data
                   MOVWF  TABLAT                 ; present data to table latch
                   TBLWT+*                       ; write data, perform a short write
                                                 ; to internal TBLWT holding register.
                   DECFSZ COUNTER                ; loop until buffers are full
                   BRA    WRITE_BYTE_TO_HREGS
PROGRAM_MEMORY
                   BSF    EECON1, WREN           ; enable write to memory
                   BCF    INTCON, GIE            ; disable interrupts
                   MOVLW  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

                   DECFSZ WRITE_COUNTER          ; done with one write cycle
                   BRA    RESTART_BUFFER         ; if not done replacing the erase block
```

7.5.2 FLASH PROGRAM MEMORY WRITE SEQUENCE (WORD PROGRAMMING)

The PIC18F47J13 Family of devices has a feature that allows programming a single word (two bytes). This feature is enabled when the WPROG bit is set. If the memory location is already erased, the following sequence is required to enable this feature:

1. Load the Table Pointer register with the address of the data to be written. (It must be an even address.)
2. Write the 2 bytes into the holding registers by performing table writes. (Do not post-increment on the second table write.)
3. Set the WREN bit (EECON1<2>) to enable writes and the WPROG bit (EECON1<5>) to select Word Write mode.
4. Disable interrupts.
5. Write 55h to EECON2.
6. Write 0AAh to EECON2.
7. Set the WR bit; this will begin the write cycle.
8. The CPU will stall for the duration of the write for TIW (see parameter D133A).
9. Re-enable interrupts.

EXAMPLE 7-4: SINGLE-WORD WRITE TO FLASH PROGRAM MEMORY

```
               MOVLW   CODE_ADDR_UPPER       ; Load TBLPTR with the base address
               MOVWF   TBLPTRU
               MOVLW   CODE_ADDR_HIGH
               MOVWF   TBLPTRH
               MOVLW   CODE_ADDR_LOW         ; The table pointer must be loaded with an even
                                               address
               MOVWF   TBLPTRL

               MOVLW   DATA0                 ; LSB of word to be written
               MOVWF   TABLAT
               TBLWT*+
               MOVLW   DATA1                 ; MSB of word to be written
               MOVWF   TABLAT
               TBLWT*                        ; The last table write must not increment the table
                                               pointer! The table pointer needs to point to the
                                               MSB before starting the write operation.


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

7.5.3 WRITE VERIFY

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

7.5.4 UNEXPECTED TERMINATION OF WRITE OPERATION

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

7.6 Flash Program Operation During Code Protection

See **Section 27.6 “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	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	Program Memory Control Register 2 (not a physical register)							
EECON1	—	—	WPROG	FREE	WRERR	WREN	WR	—

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

8.0 8 x 8 HARDWARE MULTIPLIER

8.1 Introduction

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

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

8.2 Operation

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

Example 8-2 provides the instruction sequence for 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 8-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

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

EXAMPLE 8-2: 8 x 8 SIGNED MULTIPLY ROUTINE

```
MOVF    ARG1, W
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ; PRODH:PRODL
BTFSC   ARG2, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG1
MOVF    ARG2, W
BTFSC   ARG1, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG2
```

TABLE 8-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

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

Example 8-3 provides the instruction sequence for a 16 x 16 unsigned multiplication. Equation 8-1 provides the algorithm that is used. The 32-bit result is stored in four registers (RES3:RES0).

EQUATION 8-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

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

EXAMPLE 8-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;

    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;

    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H->
                            ; PRODH:PRODL
    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 8-4 provides the sequence to do a 16 x 16 signed multiply. Equation 8-2 provides the algorithm used. The 32-bit result is stored in four registers (RES3:RES0). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

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

EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;

    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;

    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H ->
                            ; PRODH:PRODL
    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
   :
```

9.0 INTERRUPTS

Devices of the PIC18F47J13 Family have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high-priority level or a low-priority level. The high-priority interrupt vector is at 0008h and the low-priority interrupt vector is at 0018h. High-priority interrupt events will interrupt any low-priority interrupts that may be in progress.

There are 19 registers, which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3, PIR4, PIR5
- PIE1, PIE2, PIE3, PIE4, PIE5
- IPR1, IPR2, IPR3, IPR4, IPR5

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

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

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

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

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

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

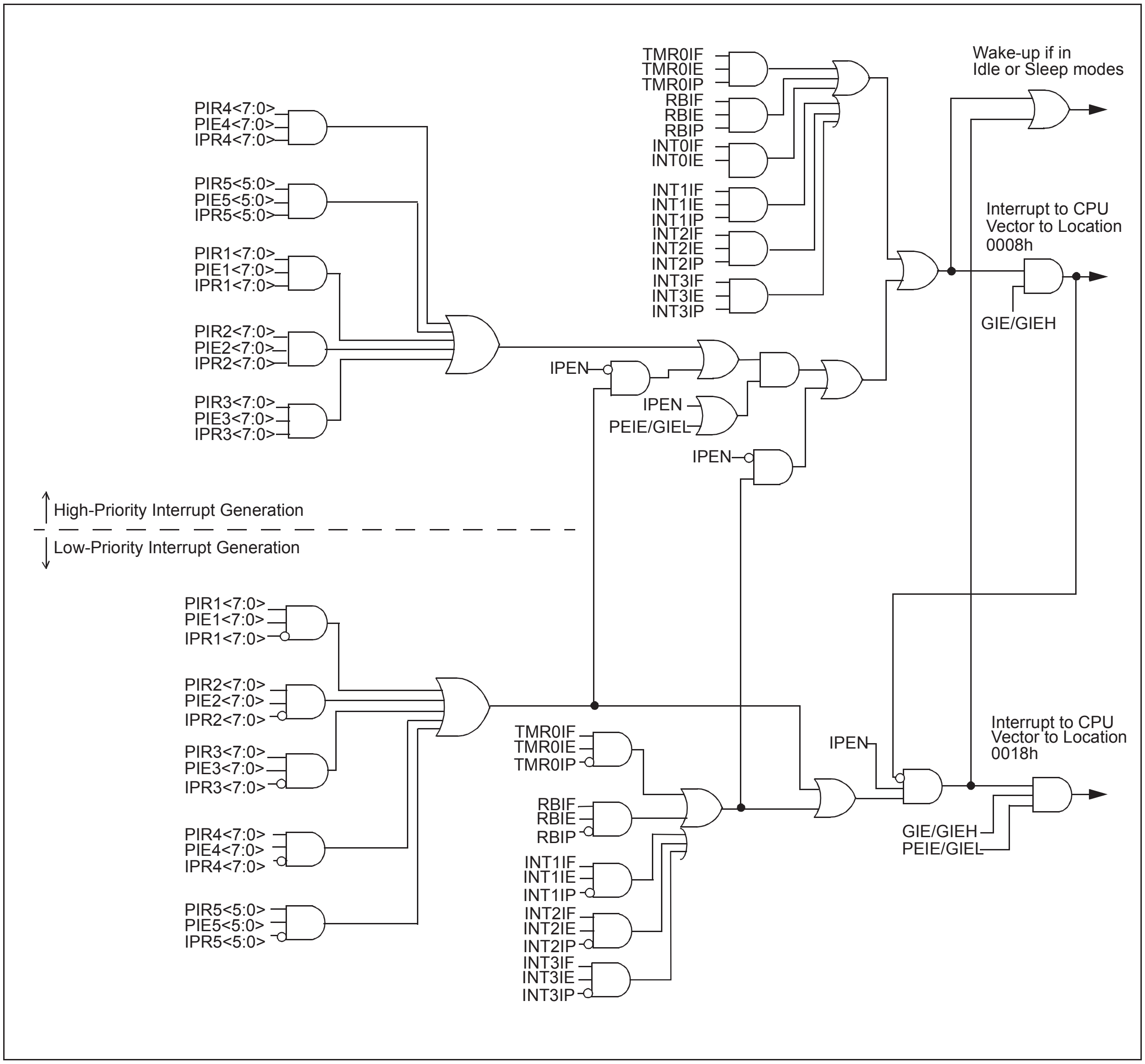
The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). Once in the Interrupt Service Routine (ISR), the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before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

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

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

Note: Do not use the `MOVFF` instruction to modify any of the Interrupt Control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior.

FIGURE 9-1: PIC18F47J13 FAMILY INTERRUPT LOGIC


TMR0IF
TMR0IE
TMR0IP
RBIF
RBIE
RBIP
INT0IF
INT0IE
INT1IF
INT1IE
INT1IP
INT2IF
INT2IE
INT2IP
INT3IF
INT3IE
INT3IP
Wake-up if in Idle or Sleep modes
PIR4<7:0>
PIE4<7:0>
IPR4<7:0>
PIR5<5:0>
PIE5<5:0>
IPR5<5:0>
PIR1<7:0>
PIE1<7:0>
IPR1<7:0>
PIR2<7:0>
PIE2<7:0>
IPR2<7:0>
PIR3<7:0>
PIE3<7:0>
IPR3<7:0>
Interrupt to CPU Vector to Location 0008h
GIE/GIEH
IPEN
IPEN
PEIE/GIEL
IPEN
High-Priority Interrupt Generation
Low-Priority Interrupt Generation
PIR1<7:0>
PIE1<7:0>
IPR1<7:0>
PIR2<7:0>
PIE2<7:0>
IPR2<7:0>
PIR3<7:0>
PIE3<7:0>
IPR3<7:0>
PIR4<7:0>
PIE4<7:0>
IPR4<7:0>
PIR5<5:0>
PIE5<5:0>
IPR5<5:0>
TMR0IF
TMR0IE
TMR0IP
RBIF
RBIE
RBIP
INT1IF
INT1IE
INT1IP
INT2IF
INT2IE
INT2IP
INT3IF
INT3IE
INT3IP
IPEN
Interrupt to CPU Vector to Location 0018h
GIE/GIEH
PEIE/GIEL

9.1 INTCON Registers

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

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

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER (ACCESS FF2h)

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` = The TMR0 register has overflowed (must be cleared in software) `0` = The 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 waiting 1 TCY will end the mismatch condition and allow the bit to be cleared.

REGISTER 9-2: INTCON2: INTERRUPT CONTROL REGISTER 2 (ACCESS FF1h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP
bit 7							bit 0

Legend:

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

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
`1` = All PORTB pull-ups are disabled
`0` = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0:** External Interrupt 0 Edge Select bit
`1` = Interrupt on rising edge
`0` = Interrupt on falling edge

bit 5 **INTEDG1:** External Interrupt 1 Edge Select bit
`1` = Interrupt on rising edge
`0` = Interrupt on falling edge

bit 4 **INTEDG2:** External Interrupt 2 Edge Select bit
`1` = Interrupt on rising edge
`0` = Interrupt on falling edge

bit 3 **INTEDG3:** External Interrupt 3 Edge Select bit
`1` = Interrupt on rising edge
`0` = Interrupt on falling edge

bit 2 **TMR0IP:** TMR0 Overflow Interrupt Priority bit
`1` = High priority
`0` = Low priority

bit 1 **INT3IP:** INT3 External Interrupt Priority bit
`1` = High priority
`0` = Low priority

bit 0 **RBIP:** RB Port Change Interrupt Priority bit
`1` = High priority
`0` = Low priority

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

REGISTER 9-3: INTCON3: INTERRUPT CONTROL REGISTER 3 (ACCESS FF0h)

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.

9.2 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Request (Flag) registers (PIR1, PIR2, PIR3).

Note 1: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE (INTCON<7>).

2: User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1 (ACCESS F9Eh)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PMPIF[1]	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

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

bit 7 **PMPIF:** Parallel Master Port Read/Write Interrupt Flag bit[1]
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF:** A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RC1IF:** EUSART1 Receive Interrupt Flag bit
1 = The EUSART1 receive buffer, RCREG1, is full (cleared when RCREG1 is read)
0 = The EUSART1 receive buffer is empty

bit 4 **TX1IF:** EUSART1 Transmit Interrupt Flag bit
1 = The EUSART1 transmit buffer, TXREG1, is empty (cleared when TXREG1 is written)
0 = The EUSART1 transmit buffer is full

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

bit 2 **CCP1IF:** ECCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1/TMR3 register capture occurred (must be cleared in software)
0 = No TMR1/TMR3 register capture occurred
Compare mode:
1 = A TMR1/TMR3 register compare match occurred (must be cleared in software)
0 = No TMR1/TMR3 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Note 1: These bits are unimplemented on 28-pin devices.

REGISTER 9-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2 (ACCESS FA1h)

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIF	CM2IF	CM1IF	—	BCL1IF	HLVDIF	TMR3IF	CCP2IF
bit 7							bit 0

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

bit 7 **OSCFIF:** Oscillator Fail Interrupt Flag bit
`1` = The device oscillator failed, clock input has changed to INTOSC (must be cleared in software)
`0` = The device clock operating

bit 6 **CM2IF:** Comparator 2 Interrupt Flag bit
`1` = The comparator input has changed (must be cleared in software)
`0` = The comparator input has not changed

bit 5 **CM1IF:** Comparator 1 Interrupt Flag bit
`1` = The comparator input has changed (must be cleared in software)
`0` = The comparator input has not changed

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

bit 3 **BCL1IF:** Bus Collision Interrupt Flag bit (MSSP1 module)
`1` = A bus collision occurred (must be cleared in software)
`0` = No bus collision occurred

bit 2 **HLVDIF:** High/Low-Voltage Detect (HLVD) Interrupt Flag bit
`1` = A High/Low-Voltage condition occurred (must be cleared in software)
`0` = An HLVD event has not occurred

bit 1 **TMR3IF:** TMR3 Overflow Interrupt Flag bit
`1` = The TMR3 register overflowed (must be cleared in software)
`0` = The TMR3 register did not overflow

bit 0 **CCP2IF:** ECCP2 Interrupt Flag bit
Capture mode:
`1` = A TMR1/TMR3 register capture occurred (must be cleared in software)
`0` = No TMR1/TMR3 register capture occurred
Compare mode:
`1` = A TMR1/TMR3 register compare match occurred (must be cleared in software)
`0` = No TMR1/TMR3 register compare match occurred
PWM mode:
Unused in this mode.

REGISTER 9-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3 (ACCESS FA4h)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	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 **SSP2IF:** Master Synchronous Serial Port 2 Interrupt Flag bit
`1` = The transmission/reception is complete (must be cleared in software)
`0` = Waiting to transmit/receive

bit 6 **BCL2IF:** Bus Collision Interrupt Flag bit (MSSP2 module)
`1` = A bus collision occurred (must be cleared in software)
`0` = No bus collision occurred

bit 5 **RC2IF:** EUSART2 Receive Interrupt Flag bit
`1` = The EUSART2 receive buffer, RCREG2, is full (cleared when RCREG2 is read)
`0` = The EUSART2 receive buffer is empty

bit 4 **TX2IF:** EUSART2 Transmit Interrupt Flag bit
`1` = The EUSART2 transmit buffer, TXREG2, is empty (cleared when TXREG2 is written)
`0` = The EUSART2 transmit buffer is full

bit 3 **TMR4IF:** TMR4 to PR4 Match Interrupt Flag bit
`1` = A TMR4 to PR4 match occurred (must be cleared in software)
`0` = No TMR4 to PR4 match occurred

bit 2 **CTMUIF:** Charge Time Measurement Unit Interrupt Flag bit
`1` = A CTMU event has occurred (must be cleared in software)
`0` = A CTMU event has not occurred

bit 1 **TMR3GIF:** Timer3 Gate Event Interrupt Flag bit
`1` = A Timer3 gate event completed (must be cleared in software)
`0` = No Timer3 gate event completed

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

REGISTER 9-7: PIR4: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 4 (ACCESS F8Fh)

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	CCP9IF	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 **CCP10IF:CCP4IF:** CCP<10:4> Interrupt Flag bits

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 **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
Unused in this mode.

REGISTER 9-8: PIR5: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 5 (ACCESS F98h)

U-0	U-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CM3IF	TMR8IF	TMR6IF	TMR5IF	TMR5GIF	TMR1GIF
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 **CM3IF:** Comparator Interrupt Flag bit

1 = Comparator 3 input has changed (must be cleared in software)
0 = Comparator 3 input has not changed

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 **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 2 **TMR5IF:** TMR3 Overflow Interrupt Flag bit

1 = TMR3 register overflowed (must be cleared in software)
0 = TMR3 register did not overflow

bit 1 **TMR5GIF:** TMR5 Gate Interrupt Flag bits

1 = TMR gate interrupt occurred (must be cleared in software)
0 = No TMR gate interrupt occurred

bit 0 **TMR1GIF:** TMR5 Gate Interrupt Flag bits

1 = TMR gate interrupt occurred (must be cleared in software)
0 = No TMR gate interrupt occurred

9.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Enable registers (PIE1, PIE2, PIE3). When IPEN = `0`, the PEIE bit must be set to enable any of these peripheral interrupts.

REGISTER 9-9: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1 (ACCESS F9Dh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PMPIE[(1)]	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

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

bit 7	**PMPIE:** Parallel Master Port Read/Write Interrupt Enable bit[(1)] `1` = Enables the PMP read/write interrupt `0` = Disables the PMP read/write interrupt
bit 6	**ADIE:** A/D Converter Interrupt Enable bit `1` = Enables the A/D interrupt `0` = Disables the A/D interrupt
bit 5	**RC1IE:** EUSART1 Receive Interrupt Enable bit `1` = Enables the EUSART1 receive interrupt `0` = Disables the EUSART1 receive interrupt
bit 4	**TX1IE:** EUSART1 Transmit Interrupt Enable bit `1` = Enables the EUSART1 transmit interrupt `0` = Disables the EUSART1 transmit interrupt
bit 3	**SSP1IE:** Master Synchronous Serial Port 1 Interrupt Enable bit `1` = Enables the MSSP1 interrupt `0` = Disables the MSSP1 interrupt
bit 2	**CCP1IE:** ECCP1 Interrupt Enable bit `1` = Enables the ECCP1 interrupt `0` = Disables the ECCP1 interrupt
bit 1	**TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit `1` = Enables the TMR2 to PR2 match interrupt `0` = Disables the TMR2 to PR2 match interrupt
bit 0	**TMR1IE:** TMR1 Overflow Interrupt Enable bit `1` = Enables the TMR1 overflow interrupt `0` = Disables the TMR1 overflow interrupt

Note 1: These bits are unimplemented on 28-pin devices.

REGISTER 9-10: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2 (ACCESS FA0h)

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
OSCFIE	CM2IE	CM1IE	—	BCL1IE	HLVDIE	TMR3IE	CCP2IE
bit 7							bit 0

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

bit 7	**OSCFIE:** Oscillator Fail Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 6	**CM2IE:** Comparator 2 Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 5	**CM1IE:** Comparator 1 Interrupt Enable bit 1 = Enabled 0 = Disabled
bit 4	**Unimplemented:** Read as ‘0’
bit 3	**BCL1IE:** Bus Collision Interrupt Enable bit (MSSP1 module) 1 = Enabled 0 = Disabled
bit 2	**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	**CCP2IE:** ECCP2 Interrupt Enable bit 1 = Enabled 0 = Disabled

REGISTER 9-11: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3 (ACCESS FA3h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	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 **SSP2IE:** Master Synchronous Serial Port 2 Interrupt Enable bit
`1` = Enabled
`0` = Disabled

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

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

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

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

bit 2 **CTMUIE:** Charge Time Measurement Unit (CTMU) Interrupt Enable bit
`1` = Enabled
`0` = Disabled

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

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

REGISTER 9-12: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4 (ACCESS F8Eh)

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	CCP9IE	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 7-1 **CCP10IE:CCP4IE:** CCP<10:4> Interrupt Enable bits

1 = Enabled
0 = Disabled

bit 0 **CCP3IE:** ECCP3 Interrupt Enable bit

1 = Enabled
0 = Disabled

REGISTER 9-13: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5 (ACCESS F91h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CM3IE	TMR8IE	TMR6IE	TMR5IE	TMR5GIE	TMR1GIE
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 **CM3IE:** Comparator 3 Receive Interrupt Enable bit

1 = Enabled
0 = Disabled

bit 4 **TMR8IE:** TMR8 to PR8 Match Interrupt Enable bit

1 = Enabled
0 = Disabled

bit 3 **TMR6IE:** TMR6 to PR6 Match Interrupt Enable bit

1 = Enabled
0 = Disabled

bit 2 **TMR5IE:** TMR5 Overflow Interrupt Enable bit

1 = Enabled
0 = Disabled

bit 1 **TMR5GIE:** TMR5 Gate Interrupt Enable bit

1 = Enabled
0 = Disabled

bit 0 **TMR1GIE:** TMR1 Gate Interrupt Enable bit

1 = Enabled
0 = Disabled

9.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Priority registers (IPR1, IPR2, IPR3). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

REGISTER 9-14: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1 (ACCESS F9Fh)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PMPIP[(1)]	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

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

bit 7 **PMPIP:** Parallel Master Port Read/Write Interrupt Priority bit[(1)]
1 = High priority
0 = Low priority

bit 6 **ADIP:** A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RC1IP:** EUSART1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TX1IP:** EUSART1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSP1IP:** Master Synchronous Serial Port Interrupt Priority bit (MSSP1 module)
1 = High priority
0 = Low priority

bit 2 **CCP1IP:** ECCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP:** TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

Note 1: These bits are unimplemented on 28-pin devices.

REGISTER 9-15: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2 (ACCESS FA2h)

R/W-1	R/W-1	R/W-1	U-0	R/W-1	R/W-1	R/W-1	R/W-1
OSCFIP	CM2IP	CM1IP	—	BCL1IP	HLVDIP	TMR3IP	CCP2IP
bit 7							bit 0

Legend:

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

bit 7 **OSCFIP:** Oscillator Fail Interrupt Priority bit
1 = High priority
0 = Low priority

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

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

bit 4 **Unimplemented:** Read as ‘0’

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

bit 2 **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 **CCP2IP:** ECCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 9-16: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3 (ACCESS FA5h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	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	**SSP2IP:** Master Synchronous Serial Port 2 Interrupt Priority bit `1` = High priority `0` = Low priority
bit 6	**BCL2IP:** Bus Collision Interrupt Priority bit (MSSP2 module) `1` = High priority `0` = Low priority
bit 5	**RC2IP:** EUSART2 Receive Interrupt Priority bit `1` = High priority `0` = Low priority
bit 4	**TX2IP:** EUSART2 Transmit Interrupt Priority bit `1` = High priority `0` = Low priority
bit 3	**TMR4IE:** TMR4 to PR4 Interrupt Priority bit `1` = High priority `0` = Low priority
bit 2	**CTMUIP:** Charge Time Measurement Unit (CTMU) Interrupt Priority bit `1` = High priority `0` = Low priority
bit 1	**TMR3GIP:** Timer3 Gate Interrupt Priority bit `1` = High priority `0` = Low priority
bit 0	**RTCCIP:** RTCC Interrupt Priority bit `1` = High priority `0` = Low priority

REGISTER 9-17: IPR4: PERIPHERAL INTERRUPT PRIORITY REGISTER 4 (ACCESS F90h)

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	CCP9IP	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 7-1	**CCP10IP:CCP4IP:** CCP<10:4> Interrupt Priority bits 1 = High priority 0 = Low priority
bit 0	**CCP3IP:** ECCP3 Interrupt Priority bit 1 = High priority 0 = Low priority

REGISTER 9-18: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5 (ACCESS F99h)

U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	CM3IP	TMR8IP	TMR6IP	TMR5IP	TMR5GIP	TMR1GIP
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	**CM3IP:** Comparator 3 Interrupt Priority bit 1 = High priority 0 = Low priority
bit 4	**TMR8IP:** TMR8 to PR8 Match Interrupt Priority bit 1 = High priority 0 = Low priority
bit 3	**TMR6IP:** TMR6 to PR6 Match Interrupt Priority bit 1 = High priority 0 = Low priority
bit 2	**TMR5IP:** TMR5 Overflow Interrupt Priority bit 1 = High priority 0 = Low priority
bit 1	**TMR5GIP:** TMR5 Gate Interrupt Priority bit 1 = High priority 0 = Low priority
bit 0	**TMR1GIP:** TMR1 Gate Interrupt Priority bit 1 = High priority 0 = Low priority

9.5 RCON Register

The RCON register contains bits used to determine the cause of the last Reset or wake-up from Idle or Sleep mode. RCON also contains the bit that enables interrupt priorities (IPEN).

REGISTER 9-19: RCON: RESET CONTROL REGISTER (ACCESS FD0h)

R/W-0	U-0	R/W-1	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	$\overline{\mathrm{CM}}$	$\overline{\mathrm{RI}}$	$\overline{\mathrm{TO}}$	$\overline{\mathrm{PD}}$	$\overline{\mathrm{POR}}$	$\overline{\mathrm{BOR}}$
bit 7							bit 0

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

bit 7	**IPEN:** Interrupt Priority Enable bit `1` = Enable priority levels on interrupts `0` = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)
bit 6	**Unimplemented:** Read as ‘`0`’
bit 5	**$\overline{\mathrm{CM}}$:** Configuration Mismatch Flag bit For details on bit operation, see Register 5-1.
bit 4	**$\overline{\mathrm{RI}}$:** `RESET` Instruction Flag bit For details on bit operation, see Register 5-1.
bit 3	**$\overline{\mathrm{TO}}$:** Watchdog Timer Time-out Flag bit For details on bit operation, see Register 5-1.
bit 2	**$\overline{\mathrm{PD}}$:** Power-Down Detection Flag bit For details on bit operation, see Register 5-1.
bit 1	**$\overline{\mathrm{POR}}$:** Power-on Reset Status bit For details on bit operation, see Register 5-1.
bit 0	**$\overline{\mathrm{BOR}}$:** Brown-out Reset Status bit For details on bit operation, see Register 5-1.

9.6 INTx Pin Interrupts

External interrupts on the INT0, INT1, INT2 and INT3 pins are edge-triggered. If the corresponding INTEDGx bit in the INTCON2 register is set (= `1`), the interrupt is triggered by a rising edge; if the bit is clear, the trigger is on the falling edge. When a valid edge appears on the INTx pin, the corresponding flag bit and INTxIF are set. This interrupt can be disabled by clearing the corresponding enable bit, INTxIE. Flag bit, INTxIF, must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts (INT0, INT1, INT2 and INT3) can wake up the processor from the Sleep and Idle modes if bit, INTxIE, was set prior to going into the power-managed modes. Deep Sleep mode can wake up from INT0, but the processor will start execution from the Power-on Reset vector rather than branch to the interrupt vector.

Interrupt priority for INT1, INT2 and INT3 is determined by the value contained in the Interrupt Priority bits, INT1IP (INTCON3<6>), INT2IP (INTCON3<7>) and INT3IP (INTCON2<1>). There is no priority bit associated with INT0; It is always a high-priority interrupt source.

9.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit, TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP (INTCON2<2>). See **Section 12.0 “Timer0 Module”** for further details on the Timer0 module.

9.8 PORTB Interrupt-on-Change

An input change on PORTB<7:4> sets flag bit, RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE (INTCON<3>). Interrupt priority for PORTB interrupt-on-change is determined by the value contained in the interrupt priority bit, RBIP (INTCON2<0>).

9.9 Context Saving During Interrupts

During interrupts, the return PC address is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the Fast Return Stack. If a fast return from interrupt is not used (see **Section 6.3 “Data Memory Organization”**), the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user’s application, other registers may also need to be saved. Example 9-1 saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine (ISR).

EXAMPLE 9-1: 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
```

10.0 I/O PORTS

Depending on the device selected and features enabled, there are up to five 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.

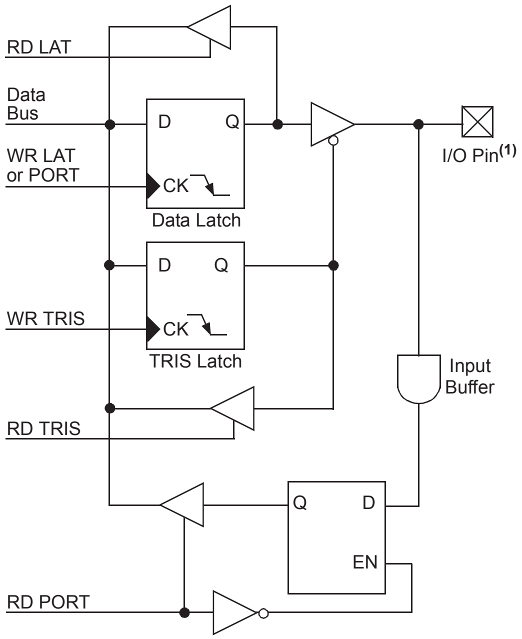
Each port has three registers for its operation. These registers are:

- TRIS register (Data Direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (Data Latch)

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

Figure 10-1 displays a simplified model of a generic I/O port, without the interfaces to other peripherals.

FIGURE 10-1: GENERIC I/O PORT OPERATION



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

10.1 I/O Port Pin Capabilities

When developing an application, the capabilities of the port pins must be considered. Outputs on some pins have higher output drive strength than others. Similarly, some pins can tolerate higher than VDD input levels.

10.1.1 PIN OUTPUT DRIVE

The output pin drive strengths vary for groups of pins intended to meet the needs for a variety of applications. PORTB and PORTC are designed to drive higher loads, such as LEDs. All other ports are designed for small loads, typically indication only. Table 10-1 summarizes the output capabilities. Refer to **Section 30.0 "Electrical Characteristics"** for more details.

TABLE 10-1: OUTPUT DRIVE LEVELS

Port	Drive	Description
PORTA	Minimum	Intended for indication.
PORTD		
PORTE		
PORTB	High	Suitable for direct LED drive levels.
PORTC		

10.1.2 INPUT PINS AND VOLTAGE CONSIDERATIONS

The voltage tolerance of pins used as device inputs is dependent on the pin's input function. Pins that are used as digital only inputs are able to handle DC voltages up to 5.5V; a level typical for digital logic circuits. In contrast, pins that also have analog input functions of any kind can only tolerate voltages up to VDD. Voltage excursions beyond VDD on these pins should be avoided. Table 10-2 summarizes the input capabilities. Refer to **Section 30.0 "Electrical Characteristics"** for more details.

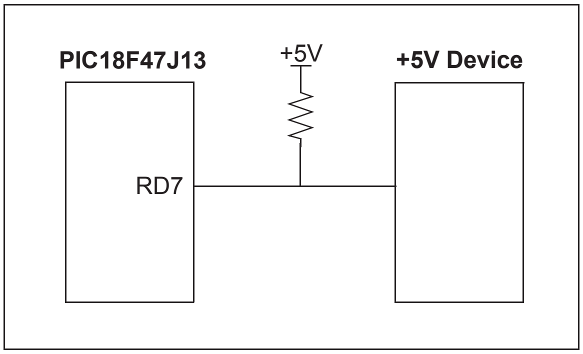
TABLE 10-2: INPUT VOLTAGE LEVELS

Port or Pin	Tolerated Input	Description
PORTA<7:0>	VDD	Only VDD input levels are tolerated.
PORTB<3:0>		
PORTC<2:0>		
PORTE<2:0>		
PORTB<7:4>	5.5V	Tolerates input levels above VDD, useful for most standard logic.
PORTC<7:3>		
PORTD<7:0>		

10.1.3 INTERFACING TO A 5V SYSTEM

Though the VDDMAX of the PIC18F47J13 Family is 3.6V, these devices are still capable of interfacing with 5V systems, even if the VIH of the target system is above 3.6V. This is accomplished by adding a pull-up resistor to the port pin (Figure 10-2), clearing the LAT bit for that pin and manipulating the corresponding TRISx bit (Figure 10-1) to either allow the line to be pulled high or to drive the pin low. Only port pins that are tolerant of voltages up to 5.5V can be used for this type of interface (refer to **Section 10.1.2 "Input Pins and Voltage Considerations"**).

FIGURE 10-2: +5V SYSTEM HARDWARE INTERFACE



EXAMPLE 10-1: COMMUNICATING WITH THE +5V SYSTEM

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

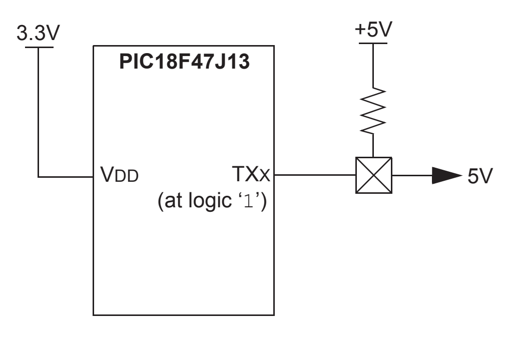
10.1.4 OPEN-DRAIN OUTPUTS

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

The open-drain option is implemented on port pins specifically associated with the data and clock outputs of the EUSARTs, the MSSP modules (in SPI mode) and the ECCP modules. It is selectively enabled by setting the open-drain control bit for the corresponding module in the ODCON registers (Register 10-1, Register 10-2 and Register 10-3). Their configuration is discussed in more detail with the individual port where these peripherals are multiplexed. Output functions that are routed through the PPS module may also use the open-drain option. The open-drain functionality will follow the I/O pin assignment in the PPS module.

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 5.5V (Figure 10-3). When a digital logic high signal is output, it is pulled up to the higher voltage level.

FIGURE 10-3: USING THE OPEN-DRAIN OUTPUT (USART SHOWN AS EXAMPLE)



10.1.5 TTL INPUT BUFFER OPTION

Many of the digital I/O ports use Schmitt Trigger (ST) input buffers. While this form of buffering works well with many types of input, some applications may require TTL level signals to interface with external logic devices. This is particularly true for the Parallel Master Port (PMP), which is likely to be interfaced to TTL level logic or memory devices.

The inputs for the PMP can be optionally configured for TTL buffers with the PMPTTL bit in the PADCFG1 register (Register 10-4). Setting this bit configures all data and control input pins for the PMP to use TTL buffers. By default, these PMP inputs use the port's ST buffers.

REGISTER 10-1: ODCON1: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 1 (BANKED F42h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CCP8OD	CCP7OD	CCP6OD	CCP5OD	CCP4OD	ECCP3OD	ECCP2OD	ECCP1OD
bit 7							bit 0

Legend:

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

bit 7 **CCP8OD:** CCP8 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 6 **CCP7OD:** CCP7 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 5 **CCP6OD:** CCP6 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 4 **CCP5OD:** CCP5 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 3 **CCP4OD:** CCP4 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 2 **ECCP3OD:** ECCP3 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 1 **ECCP2OD:** ECCP2 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 0 **ECCP1OD:** ECCP1 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

REGISTER 10-2: ODCON2: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 2 (BANKED F41h)

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	CCP10OD	CCP9OD	U2OD	U1OD
bit 7							bit 0

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

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3 **CCP10OD:** CCP10 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 2 **CCP9OD:** CCP9 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 1 **U2OD:** USART2 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 0 **U1OD:** USART1 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

REGISTER 10-3: ODCON3: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 3 (BANKED F40h)

R/W-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
CTMUDS	—	—	—	—	—	SPI2OD	SPI1OD
bit 7							bit 0

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

bit 7 **CTMUDS:** CTMU Pulse Delay Enable bit
1 = Pulse delay input for CTMU enabled on pin, RA1

bit 6-2 **Unimplemented:** Read as ‘0’

bit 1 **SPI2OD:** SPI2 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

bit 0 **SPI1OD:** SPI1 Open-Drain Output Enable bit
1 = Open-drain capability is enabled
0 = Open-drain capability is disabled

REGISTER 10-4: PADCFG1: PAD CONFIGURATION CONTROL REGISTER 1 (BANKED F3Ch)

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	RTSECSEL1[1]	RTSECSEL0[1]	PMPTTL
bit 7							bit 0

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

bit 7-3 **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 (can be INTRC, T1OSC or T1CKI depending upon the RTCOSC (CONFIG3L<1>) and T1OSCEN (T1CON<3>) bit settings)
01 = RTCC seconds clock is selected for the RTCC pin
00 = RTCC alarm pulse is selected for the RTCC pin

bit 0 **PMPTTL:** PMP Module TTL Input Buffer Select bit

1 = PMP module uses TTL input buffers
0 = PMP module uses Schmitt Trigger input buffers

Note 1: To enable the actual RTCC output, the RTCOE (RTCCFG<2>) bit needs to be set.

10.2 PORTA, TRISA and LATA Registers

PORTA is a 7-bit wide, bidirectional port. It may function as a 5-bit port, depending on the oscillator mode selected. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it, will write to the port latch.

The Data Latch (LATA) register is also memory mapped. Read-modify-write operations on the LATA register read and write the latched output value for PORTA.

The other PORTA pins are multiplexed with analog inputs, the analog VREF+ and VREF- inputs and the comparator voltage reference output. The operation of pins, RA<3:0> and RA5, as A/D Converter inputs is selected by clearing or setting the control bits in the ANCON0 register (A/D Port Configuration Register 0).

The RAx pins may also be used as comparator inputs by setting the appropriate bits in the CMxCON registers. To use RAx as digital inputs, it is necessary to turn off the comparators.

Note: On a Power-on Reset (POR), RA5 and RA<3:0> are configured as analog inputs and read as ‘0’.

All PORTA pins have TTL input levels and full CMOS output drivers.

The TRISA register controls the direction of the PORTA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 10-2: INITIALIZING PORTA

```
CLRF    PORTA   ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF    LATA    ; Alternate method
                ; to clear output
                ; data latches
MOVLB   0x0F    ; ANCONx register not in
                ; Access Bank
MOVLW   0x0F    ; Configure A/D
MOVWF   ANCON0  ; for digital inputs
MOVLW   0xCF    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISA   ; Set RA<3:0> as inputs
                ; RA<5:4> as outputs
```

TABLE 10-3: PORTA I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RA0/AN0/C1INA/ULPWU/PMA6/RP0	RA0	1	I	TTL	PORTA<0> data input; disabled when analog input is enabled.
		0	O	DIG	LATA<0> data output; not affected by analog input.
	AN0	1	I	ANA	A/D Input Channel 0 and Comparator C1- input. Default input configuration on POR; does not affect digital output.
	C1INA	1	I	ANA	Comparator 1 Input A.
	ULPWU	1	I	ANA	Ultra low-power wake-up input.
	PMA6[1]	x	I/O	ST/TTL/DIG	Parallel Master Port digital I/O.
	RP0	1	I	ST	Remappable Peripheral Pin 0 input.
		0	O	DIG	Remappable Peripheral Pin 0 output.
RA1/AN1/C2INA/VBG/CTDIN/PMA7/RP1	RA1	1	I	TTL	PORTA<1> data input; disabled when analog input is enabled.
		0	O	DIG	LATA<1> data output; not affected by analog input.
	AN1	1	I	ANA	A/D Input Channel 1 and Comparator C2- input. Default input configuration on POR; does not affect digital output.
	C2INA	1	I	ANA	Comparator 1 Input A.
	VBG	x	O	ANA	Band Gap Voltage Reference output. (Enabled by setting the VBGOE bit (WDTCON<4>.)
	CTDIN	1	I	ST	CTMU pulse delay input.
	PMA7[1]	1	I	ST/TTL	Parallel Master Port (`io_addr_in[7]`).
		0	O	DIG	Parallel Master Port address.
	RP1	1	I	ST	Remappable Peripheral Pin 1 input.
		0	O	DIG	Remappable Peripheral Pin 1 output
RA2/AN2/C2INB/C1IND/C3INB/VREF-/CVREF	RA2	0	O	DIG	LATA<2> data output; not affected by analog input. Disabled when CVREF output is enabled.
		1	I	TTL	PORTA<2> data input. Disabled when analog functions are enabled; disabled when CVREF output is enabled.
	AN2	1	I	ANA	A/D Input Channel 2 and Comparator C2+ input. Default input configuration on POR; not affected by analog output.
	C2INB	1	I	ANA	Comparator 2 Input B.
		0	O	ANA	CTMU pulse generator charger for the C2INB comparator input.
	C1IND	1	I	ANA	Comparator 1 Input D.
	C3INB	1	I	ANA	Comparator 3 Input B.
	VREF-	1	I	ANA	A/D and comparator voltage reference low input.
	CVREF	x	O	ANA	Comparator voltage reference output. Enabling this feature disables digital I/O.
RA3/AN3/C1INB/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 and Comparator C1+ input. Default input configuration on POR.
	C1INB	1	I	ANA	Comparator 1 Input B
	VREF+	1	I	ANA	A/D and comparator voltage reference high input.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRISx bit does not affect port direction or is overridden for this option)

Note 1: This bit is only available on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

TABLE 10-3: PORTA I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RA5/AN4/C1INC/$\overline{\text{SS1}}$/HLVDIN/RP2	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.
	C1INC	0	O	DIG	Comparator 1 Input C.
	$\overline{\text{SS1}}$	1	I	TTL	Slave select input for MSSP1.
	HLVDIN	1	I	ANA	High/Low-Voltage Detect external trip point reference input.
	RP2	1	I	ST	Remappable Peripheral Pin 2 input.
		0	O	DIG	Remappable Peripheral Pin 2 output.
OSC2/CLKO/RA6	OSC2	x	O	ANA	Main oscillator feedback output connection (HS mode).
	CLKO	x	O	DIG	System cycle clock output (FOSC/4) in RC and EC Oscillator modes.
	RA6	1	I	TTL	PORTA<6> data input.
		0	O	DIG	LATA<6> data output.
OSC1/CLKI/RA7	OSC1	1	I	ANA	Main oscillator input connection.
	CLKI	1	I	ANA	Main clock input connection.
	RA7	1	I	TTL	PORTA<6> data input.
		0	O	DIG	LATA<6> data output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRISx bit does not affect port direction or is overridden for this option)

Note 1: This bit is only available on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

TABLE 10-4: 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	RA6	RA5	—	RA3	RA2	RA1	RA0
LATA	LAT7	LAT6	LAT5	—	LAT3	LAT2	LAT1	LAT0
TRISA	TRIS7	TRIS6	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0
ANCON0	PCFG7[(1)]	PCFG6[(1)]	PCFG5[(1)]	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
CMxCON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0
WDTCON	REGSLP	LVDSTAT	ULPLVL	VBGOE	DS	ULPEN	ULPSINK	SWDTEN
HLVDCON	VDIRMAG	BGVST	IRVST	HLVDEN	HLVDL3	HLVDL2	HLVDL1	HLVDL0

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

Note 1: These bits are only available in 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

10.3 PORTB, TRISB and LATB Registers

PORTB is an 8-bit wide, bidirectional port. The corresponding Data Direction register is TRISB. Setting a TRISB bit (= `1`) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISB bit (= `0`) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATB) is also memory mapped. Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

EXAMPLE 10-3: INITIALIZING PORTB

```
CLRF   PORTB        ; Initialize PORTB by
                    ; clearing output
                    ; data latches
CLRF   LATB         ; Alternate method
                    ; to clear output
                    ; data latches
MOVLB  0x0F         ; ANCON1 not in Access
                    ; Bank
MOVLW  0x17         ; Configure as digital I/O
MOVWF  ANCON1       ; pins in this example

MOVLW  0xCF         ; 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 POR. The integrated weak pull-ups consist of a semiconductor structure similar to, but somewhat different from, a discrete resistor. On an unloaded I/O pin, the weak pull-ups are intended to provide logic high indication, but will not necessarily pull the pin all the way to VDD levels.

Note: On a POR, the RB<3:0> bits are configured as analog inputs by default and read as ‘`0`’; RB<7:4> bits are configured as digital inputs.

Four of the PORTB pins (RB<7:4>) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB<7:4> pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB<7:4>) are compared with the old value latched on the last read of PORTB. The “mismatch” outputs of RB<7:4> are ORed together to generate the RB Port Change Interrupt with Flag bit, RBIF (INTCON<0>).

This interrupt can wake the device from Sleep mode or any of the Idle modes. Application software can clear the interrupt flag by following these steps:

1. Any read or write of PORTB (except with the `MOVFF (ANY), PORTB` instruction).
2. Wait one instruction cycle (such as executing a `NOP` instruction).
3. Clear flag bit, RBIF.

A mismatch condition continues to set flag bit, RBIF. Reading PORTB will end the mismatch condition and allow flag bit, RBIF, to be cleared after one instruction cycle of 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 RB5 pin is multiplexed with the Timer0 module clock input and one of the comparator outputs to become the RB5/CCP5/PMA0/KBI1/RP8 pin.

TABLE 10-5: PORTB I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RB0/AN12/ C3IND/INT0/ RP3	RB0	1	I	TTL	PORTB<0> data input; weak pull-up when the $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input is enabled.(1)
		0	O	DIG	LATB<0> data output; not affected by an analog input.
	AN12	1	I	ANA	A/D Input Channel 12.(1)
	C3IND	1	I	ANA	Comparator 3 Input D.
	INT0	1	I	ST	External Interrupt 0 input.
	RP3	1	I	ST	Remappable Peripheral Pin 3 input.
		0	O	DIG	Remappable Peripheral Pin 3 output.
RB1/AN10/ C3INC/PMBE/ RTCC/RP4	RB1	1	I	TTL	PORTB<1> data input; weak pull-up when the $\overline{\text{RBPU}}$ bit is cleared. Disabled when an analog input is enabled.(1)
		0	O	DIG	LATB<1> data output; not affected by an analog input.
	AN10	1	I	ANA	A/D Input Channel 10.(1)
	C3INC	1	I	ANA	Comparator 3 Input C.
	PMBE(3)	x	O	DIG	Parallel Master Port byte enable.
	RTCC	0	O	DIG	Asynchronous serial transmit data output (USART module).
	RP4	1	I	ST	Remappable Peripheral Pin 4 input.
		0	O	DIG	Remappable Peripheral Pin 4 output.
RB2/AN8/ C2INC/CTED1/ PMA3/REFO/ RP5	RB2	1	I	TTL	PORTB<2> data input; weak pull-up when the $\overline{\text{RBPU}}$ bit is cleared. Disabled when an analog input is enabled.(1)
		0	O	DIG	LATB<2> data output; not affected by an analog input.
	AN8	1	I	ANA	A/D Input Channel 8.(1)
	C2INC	1	I	ANA	Comparator 2 Input C.
	CTED1	1	I	ST	CTMU Edge 1 input.
	PMA3(3)	x	O	DIG	Parallel Master Port address.
	REFO	0	O	DIG	Reference output clock.
	RP5	1	I	ST	Remappable Peripheral Pin 5 input.
		0	O	DIG	Remappable Peripheral Pin 5 output.
RB3/AN9/ C3INA/CTED2/ PMA2/RP6	RB3	0	O	DIG	LATB<3> data output; not affected by analog input.
		1	I	TTL	PORTB<3> data input; weak pull-up when the $\overline{\text{RBPU}}$ bit is cleared. Disabled when analog input is enabled.(1)
	AN9	1	I	ANA	A/D Input Channel 9.(1)
	C3INA	1	I	ANA	Comparator 3 Input A.
	CTED2	1	I	ST	CTMU Edge 2 input.
	PMA2(3)	x	O	DIG	Parallel Master Port address.
	RP6	1	I	ST	Remappable Peripheral Pin 6 input.
		0	O	DIG	Remappable Peripheral Pin 6 output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRISx bit does not affect port direction or is overridden for this option)

Note 1: Pins are configured as analog inputs by default on POR. Using these pins for digital inputs requires setting the appropriate bits in the ANCON1 register.

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

3: Only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

4: Only on 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).

TABLE 10-5: PORTB I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RB4/CCP4/ PMA1/KBI0/ SCL2[(4)]/RP7	RB4	`0`	O	DIG	LATB<4> data output; not affected by an analog input.
		`1`	I	TTL	PORTB<4> data input; weak pull-up when the $\overline{\text{RBPU}}$ bit is cleared. Disabled when an analog input is enabled.(1)
	CCP4(3)	`1`	I	ST	Capture input.
		`0`	O	DIG	Compare/PWM output.
	PMA1	`x`	I/O	ST/TTL/ DIG	Parallel Master Port address.
	KBI0	`1`	I	TTL	Interrupt-on-change pin.
	SCL2(4)	`1`	I	I^2C/ SMBus	I^2C clock input (MSSP2 module).
	RP7	`1`	I	ST	Remappable Peripheral Pin 7 input.
		`0`	O	DIG	Remappable Peripheral Pin 7 output.
RB5/CCP5/ PMA0/KBI1/ SDA2(4)/RP8	RB5	`0`	O	DIG	LATB<5> data output.
		`1`	I	TTL	PORTB<5> data input; weak pull-up when the $\overline{\text{RBPU}}$ bit is cleared.
	CCP5(3)	`1`	I	ST	Capture input.
		`0`	O	DIG	Compare/PWM output.
	PMA0(3)	`x`	I/O	ST/TTL/ DIG	Parallel Master Port address.
	KBI1	`1`	I	TTL	Interrupt-on-change pin.
	SDA2(4)	`1`	I	I^2C/ SMBus	I^2C data input (MSSP2 module).
	RP8	`1`	I	ST	Remappable Peripheral Pin 8 input.
		`0`	O	DIG	Remappable Peripheral Pin 8 output.
RB6/CCP6/ KBI2/PGC/RP9	RB6	`0`	O	DIG	LATB<6> data output.
		`1`	I	TTL	PORTB<6> data input; weak pull-up when the $\overline{\text{RBPU}}$ bit is cleared.
	CCP6(3)	`1`	I	ST	Capture input.
		`0`	O	DIG	Compare/PWM output.
	KBI2	`1`	I	TTL	Interrupt-on-change pin.
	PGC	`x`	I	ST	Serial execution (ICSP™) clock input for ICSP and ICD operation.(2)
	RP9	`1`	I	ST	Remappable Peripheral Pin 9 input.
		`0`	O	DIG	Remappable Peripheral Pin 9 output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; `x` = Don't care (TRISx bit does not affect port direction or is overridden for this option)

Note 1: Pins are configured as analog inputs by default on POR. Using these pins for digital inputs requires setting the appropriate bits in the ANCON1 register.

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

3: Only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

4: Only on 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).

TABLE 10-5: PORTB I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RB7/CCP7/ KBI3/PGD/ RP10	RB7	`0`	O	DIG	LATB<7> data output.
		`1`	I	TTL	PORTB<7> data input; weak pull-up when the $\overline{\text{RBPU}}$ bit is cleared.
	CCP7	`1`	I	ST	Capture input.
		`0`	O	DIG	Compare/PWM output.
	KBI3	`1`	O	TTL	Interrupt-on-change pin.
	PGD	`x`	O	DIG	Serial execution data output for ICSP and ICD operation.(2)
		`x`	I	ST	Serial execution data input for ICSP and ICD operation.(2)
	RP10	`1`	I	ST	Remappable Peripheral Pin 10 input.
		`0`	O	DIG	Remappable Peripheral Pin 10 output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; `x` = Don't care (TRISx bit does not affect port direction or is overridden for this option)

Note 1: Pins are configured as analog inputs by default on POR. Using these pins for digital inputs requires setting the appropriate bits in the ANCON1 register.

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

3: Only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

4: Only on 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF27J13).

TABLE 10-6: 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
ANCON1	VBGEN	—	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
REFOCON	ROON	—	ROSSLP	ROSEL	RODIV3	RODIV2	RODIV1	RODIV0
CM3CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0
PADCFG1	—	—	—	—	—	RTSECSEL1	RTSECSEL0	PMPTTL
RTCCFG	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0

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

10.4 PORTC, TRISC and LATC Registers

PORTC is an 8-bit wide, bidirectional port. The corresponding Data Direction register is TRISC. Setting a TRISC bit (= `1`) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISC bit (= `0`) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATC) is also memory mapped. Read-modify-write operations on the LATC register read and write the latched output value for PORTC.

PORTC is multiplexed with several peripheral functions (see Table 10-7). The pins have Schmitt Trigger input buffers.

When enabling peripheral functions, care should be taken in defining TRISx bits for each PORTC pin. Some peripherals override the TRISx bit to make a pin an output, while other peripherals override the TRISx bit to make a pin an input. The user should refer to the corresponding peripheral section for additional information.

Note: On a Power-on Reset, PORTC pins (except RC2) are configured as digital inputs. RC2 will default as an analog input (controlled by the ANCON1 register).

The contents of the TRISC register are affected by peripheral overrides. Reading TRISC always returns the current contents, even though a peripheral device may be overriding one or more of the pins.

EXAMPLE 10-4: INITIALIZING PORTC

```
CLRF   PORTC    ;  Initialize PORTC by
                ; clearing output
                ; data latches
CLRF   LATC     ; Alternate method
                ; to clear output
                ; data latches
MOVLW  0x3F     ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISC    ; Set RC<5:0> as inputs
                ;  RC<7:6> as outputs
MOVLB  0x0F     ; ANCON register is not in
                  Access Bank
BSF    ANCON1,PCFG11
                ;Configure RC2/AN11 as
                 digital input
```

TABLE 10-7: PORTC I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RC0/T1OSO/ T1CKI/RP11	RC0	1	I	ST	PORTC<0> data input.
		0	O	DIG	LATC<0> data output.
	T1OSO	x	O	ANA	Timer1 oscillator output; enabled when Timer1 oscillator is enabled. Disables digital I/O.
	T1CKI	1	I	ST	Timer1 digital clock input.
	RP11	1	I	ST	Remappable Peripheral Pin 11 input.
		0	O	DIG	Remappable Peripheral Pin 11 output.
RC1/CCP8/ T1OSI/RP12	RC1	1	I	ST	PORTC<1> data input.
		0	O	DIG	LATC<1> data output.
	CCP8	1	I	ST	Capture input.
		0	O	DIG	Compare/PWM output.
	T1OSI	x	I	ANA	Timer1 oscillator input; enabled when Timer1 oscillator is enabled. Disables digital I/O.
	RP12	1	I	ST	Remappable Peripheral Pin 12 input.
		0	O	DIG	Remappable Peripheral Pin 12 output.
RC2/AN11/ C2IND/CTPLS/ RP13	RC2	1	I	ST	PORTC<2> data input.
		0	O	DIG	PORTC<2> data output.
	AN11	1	I	ANA	A/D Input Channel 11.
	C2IND	1	I	ANA	Comparator 2 Input D.
	CTPLS	0	O	DIG	CTMU pulse generator output.
	RP13	1	I	ST	Remappable Peripheral Pin 13 input.
		0	O	DIG	Remappable Peripheral Pin 13 output.
RC3/SCK1/ SCL1/RP14	RC3	1	I	ST	PORTC<3> data input.
		0	O	DIG	PORTC<3> data output.
	SCK1	1	I	ST	SPI clock input (MSSP1 module).
		0	O	DIG	SPI clock output (MSSP1 module).
	SCL1	1	I	I^{2}C/ SMBus	I^{2}C clock input (MSSP1 module).
		0	O	DIG	I^{2}C clock output (MSSP1 module).
	RP14	1	I	ST	Remappable Peripheral Pin 14 input.
		0	O	DIG	Remappable Peripheral Pin 14 output.
RC4/SDI1/ SDA1/RP15	RC4	0	O	DIG	PORTC<4> data output.
	SDI1	1	I	ST	SPI data input (MSSP1 module).
	SDA1	1	I	I^{2}C/ SMBus	I^{2}C data input (MSSP1 module).
		0	O	DIG	I^{2}C data output (MSSP1 module).
	RP15	1	I	ST	Remappable Peripheral Pin 15 input.
		0	O	DIG	Remappable Peripheral Pin 15 output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^{2}C/SMB = I^{2}C/SMBus input buffer; `x` = Don’t care (TRISx bit does not affect port direction or is overridden for this option)

Note 1: This bit is only available on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

TABLE 10-7: PORTC I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RC5/SDO1/ RP16	RC5	0	O	DIG	PORTC<5> data output.
	SDO1	x	O	DIG	SPI data output (MSSP1 module).
	RP16	1	I	ST	Remappable Peripheral Pin 16 input.
		0	O	DIG	Remappable Peripheral Pin 16 output.
RC6/CCP9/ PMA5/TX1/ CK1/RP17	RC6	1	I	ST	PORTC<6> data input.
		0	O	DIG	LATC<6> data output.
	CCP9	1	I	ST	Capture input.
		0	O	DIG	Compare/PWM output.
	PMA5(1)	1	I	ST/TTL	Parallel Master Port `io_addr_in<5>`.
		0	O	DIG	Parallel Master Port address.
	TX1	0	O	DIG	Asynchronous serial transmit data output (EUSART module); takes priority over port data. User must configure as an output.
	CK1	1	I	ST	Synchronous serial clock input (EUSART module).
		0	O	DIG	Synchronous serial clock output (EUSART module); takes priority over port data.
	RP17	1	I	ST	Remappable Peripheral Pin 17 input.
		0	O	DIG	Remappable Peripheral Pin 17 output.
RC7/CCP10/ PMA4/RX1/ DT1/RP18	RC7	1	I	ST	PORTC<7> data input.
		0	O	DIG	LATC<7> data output.
	CCP10	1	I	ST	Capture input.
		0	O	DIG	Compare/PWM output.
	PMA4(1)	x	I/O	ST/TTL/ DIG	Parallel Master Port address.
	RX1	1	I	ST	Asynchronous serial receive data input (EUSART module).
	DT1	1	1	ST	Synchronous serial data input (EUSART module). User must configure as an input.
		0	O	DIG	Synchronous serial data output (EUSART module); takes priority over port data.
	RP18	1	I	ST	Remappable Peripheral Pin 18 input.
		0	O	DIG	Remappable Peripheral Pin 18 output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; I^2C/SMB = I^2C/SMBus input buffer; x = Don’t care (TRISx bit does not affect port direction or is overridden for this option)

Note 1: This bit is only available on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

TABLE 10-8: 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	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
ANCON1	VBGEN	—	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
CM2CON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0
RTCCFG	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0

10.5 PORTD, TRISD and LATD Registers

Note: PORTD is available only in 44-pin devices.

PORTD is an 8-bit wide, bidirectional port. The corresponding Data Direction register is TRISD. Setting a TRISD bit (= `1`) will make the corresponding PORTD pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISD bit (= `0`) will make the corresponding PORTD pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATD) is also memory mapped. Read-modify-write operations on the LATD register read and write the latched output value for PORTD.

All pins on PORTD are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Note: On a POR, these pins are configured as digital inputs.

EXAMPLE 10-5: INITIALIZING PORTD

```
CLRF   PORTD    ;  Initialize PORTD by
                ; clearing output
                ; data latches
CLRF   LATD     ; Alternate method
                ; to clear output
                ; data latches
MOVLW  0xCF     ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISD    ; Set RD<3:0> as inputs
                ;  RD<5:4> as outputs
                ;  RD<7:6> as inputs
```

Each of the PORTD pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit, $\overline{\text{RDPU}}$ (TRISE<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 POR. The integrated weak pull-ups consist of a semiconductor structure similar to, but somewhat different from, a discrete resistor. On an unloaded I/O pin, the weak pull-ups are intended to provide logic high indication, but will not necessarily pull the pin all the way to VDD levels.

Note that the pull-ups can be used for any set of features, similar to the pull-ups found on PORTB.

TABLE 10-9: PORTD I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RD0/PMD0/SCL2	RD0	1	I	ST	PORTD<0> data input.
		0	O	DIG	LATD<0> data output.
	PMD0(1)	1	I	ST/TTL	Parallel Master Port data in.
		0	O	DIG	Parallel Master Port data out.
	SCL2(1)	1	I	I^2C/SMB	I^2C clock input (MSSP2 module); input type depends on the module setting.
		0	O	DIG	I^2C clock output (MSSP2 module); takes priority over port data.
RD1/PMD1/SDA2	RD1	1	I	ST	PORTD<1> data input.
		0	O	DIG	LATD<1> data output.
	PMD1(1)	1	I	ST/TTL	Parallel Master Port data in.
		0	O	DIG	Parallel Master Port data out.
	SDA2(1)	1	I	I^2C/SMB	I^2C data input (MSSP2 module); input type depends on the module setting.
		0	O	DIG	I^2C data output (MSSP2 module); takes priority over port data.
RD2/PMD2/RP19	RD2	1	I	ST	PORTD<2> data input.
		0	O	DIG	LATD<2> data output.
	PMD2(1)	1	I	ST/TTL	Parallel Master Port data in.
		0	O	DIG	Parallel Master Port data out.
	RP19	1	I	ST	Remappable Peripheral Pin 19 input.
		0	O	DIG	Remappable Peripheral Pin 19 output.
RD3/PMD3/RP20	RD3	1	I	ST	PORTD<3> data input.
		0	O	DIG	LATD<3> data output.
	PMD3(1)	1	I	ST/TTL	Parallel Master Port data in.
		0	O	DIG	Parallel Master Port data out.
	RP20	1	I	ST	Remappable Peripheral Pin 20 input.
		0	O	DIG	Remappable Peripheral Pin 20 output.
RD4/PMD4/RP21	RD4	1	I	ST	PORTD<4> data input.
		0	O	DIG	LATD<4> data output.
	PMD4(1)	1	I	ST/TTL	Parallel Master Port data in.
		0	O	DIG	Parallel Master Port data out.
	RP21	1	I	ST	Remappable Peripheral Pin 21 input.
		0	O	DIG	Remappable Peripheral Pin 21 output.
RD5/PMD5/RP22	RD5	1	I	ST	PORTD<5> data input.
		0	O	DIG	LATD<5> data output.
	PMD5(1)	1	I	ST/TTL	Parallel Master Port data in.
		0	O	DIG	Parallel Master Port data out.
	RP22	1	I	ST	Remappable Peripheral Pin 22 input.
		0	O	DIG	Remappable Peripheral Pin 22 output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; I^2C/SMB = I^2C/SMBus input buffer; `x` = Don't care (TRISx bit does not affect port direction or is overridden for this option).

Note 1: Only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

TABLE 10-9: PORTD I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RD6/PMD6/ RP23	RD6	`1`	I	ST	PORTD<6> data input.
		`0`	O	DIG	LATD<6> data output.
	PMD6[1]	`1`	I	ST/TTL	Parallel Master Port data in.
		`0`	O	DIG	Parallel Master Port data out.
	RP23	`1`	I	ST	Remappable Peripheral Pin 23 input.
		`0`	O	DIG	Remappable Peripheral Pin 23 output.
RD7/PMD7/ RP24	RD7	`1`	I	ST	PORTD<7> data input.
		`0`	O	DIG	LATD<7> data output.
	PMD7[1]	`1`	I	ST/TTL	Parallel Master Port data in.
		`0`	O	DIG	Parallel Master Port data out.
	RP24	`1`	I	ST	Remappable Peripheral Pin 24 input.
		`0`	O	DIG	Remappable Peripheral Pin 24 output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; I^{2}C/SMB = I^{2}C/SMBus input buffer; `x` = Don't care (TRISx bit does not affect port direction or is overridden for this option).

Note 1: Only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13).

TABLE 10-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTD[1]	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
LATD[1]	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0
TRISD[1]	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0

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

Note 1: These registers are not available in 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF26J13).

10.6 PORTE, TRISE and LATE Registers

Note: PORTE is available only in 44-pin devices.

Depending on the particular PIC18F47J13 Family device selected, PORTE is implemented in two different ways.

For 44-pin devices, PORTE is a 3-bit wide port. Three pins (RE0/AN5/PMRD, RE1/AN6/PMWR and RE2/AN7/PMCS) are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers. When selected as analog inputs, these pins will read as '`0`'s.

The corresponding Data Direction register is TRISE. Setting a TRISE bit (= `1`) will make the corresponding PORTE pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISE bit (= `0`) will make the corresponding PORTE pin an output (i.e., put the contents of the output latch on the selected pin).

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

Note: On a POR, RE<2:0> are configured as analog inputs.

The Data Latch register (LATE) is also memory mapped. Read-modify-write operations on the LATE register read and write the latched output value for PORTE.

EXAMPLE 10-6: INITIALIZING PORTE

```
CLRF   PORTE    ; Initialize PORTE by
                ; clearing output
                ; data latches
CLRF   LATE     ; Alternate method
                ; to clear output
                ; data latches
MOVLW  0xE0     ; Configure REx
MOVWF  ANCON0   ; for digital inputs
MOVLW  0x03     ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISE    ; Set RE<0> as inputs
                ; RE<1> as outputs
                ; RE<2> as inputs
```

Each of the PORTE pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit, $\overline{\text{REPU}}$ (TRISE<6>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a POR. The integrated weak pull-ups consist of a semiconductor structure similar to, but somewhat different, from a discrete resistor. On an unloaded I/O pin, the weak pull-ups are intended to provide logic high indication, but will not necessarily pull the pin all the way to VDD levels.

Note that the pull-ups can be used for any set of features, similar to the pull-ups found on PORTB.

TABLE 10-11: PORTE I/O SUMMARY

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RE0/AN5/ PMRD	RE0	`1`	I	ST	PORTE<0> data input; disabled when analog input is enabled.
		`0`	O	DIG	LATE<0> data output; not affected by analog input.
	AN5	`1`	I	ANA	A/D Input Channel 5; default input configuration on POR.
	PMRD	`1`	I	ST/TTL	Parallel Master Port (`io_rd_in`).
		`0`	O	DIG	Parallel Master Port read strobe.
RE1/AN6/ PMWR	RE1	`1`	I	ST	PORTE<1> data input; disabled when analog input is enabled.
		`0`	O	DIG	LATE<1> data output; not affected by analog input.
	AN6	`1`	I	ANA	A/D Input Channel 6; default input configuration on POR.
	PMWR	`1`	I	ST/TTL	Parallel Master Port (`io_wr_in`).
		`0`	O	DIG	Parallel Master Port write strobe.
RE2/AN7/ PMCS	RE2	`1`	I	ST	PORTE<2> data input; disabled when analog input is enabled.
		`0`	O	DIG	LATE<2> data output; not affected by an analog input.
	AN7	`1`	I	ANA	A/D Input Channel 7; default input configuration on POR.
	PMCS	`0`	O	DIG	Parallel Master Port byte enable.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level I = Input; O = Output; P = Power

TABLE 10-12: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTE(1)	—	—	—	—	—	RE2	RE1	RE0
LATE(1)	—	—	—	—	—	LATE2	LATE1	LATE0
TRISE(1)	$\overline{\text{RDPU}}$	$\overline{\text{REPU}}$	—	—	—	TRISE2	TRISE1	TRISE0
ANCON0	PCFG7(1)	PCFG6(1)	PCFG5(1)	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0

Legend: — = unimplemented, read as ‘`0`’. Shaded cells are not used by PORTE.

Note 1: These registers and/or bits are not available in 28-pin devices (PIC18F26J13, PIC18F27J13, PIC18LF26J13 and PIC18LF26J13).

Note: bit 7 **$\overline{\text{RDPU}}$:** PORTD Pull-up Enable bit
`0` = All PORTD pull-ups are disabled
`1` = $\overline{\text{PORTD}}$ pull-ups are enabled for any input pad
bit 6 **$\overline{\text{REPU}}$:** PORTE Pull-up Enable bit
`0` = All PORTE pull-ups are disabled
`1` = PORTE pull-ups are enabled for any input pad

10.7 Peripheral Pin Select (PPS)

A major challenge in general purpose devices is providing the largest possible set of peripheral features while minimizing the conflict of features on I/O pins. The challenge is even greater on low pin count devices similar to the PIC18F47J13 Family. In an application that needs to use more than one peripheral, multiplexed on a single pin, inconvenient work arounds in application code or a complete redesign may be the only option.

The Peripheral Pin Select (PPS) feature provides an alternative to these choices by enabling the user's peripheral set selection and its placement on a wide range of I/O pins. By increasing the pinout options available on a particular device, users can better tailor the microcontroller to their entire application, rather than trimming the application to fit the device.

The PPS feature operates over a fixed subset of digital I/O pins. Users may independently map the input and/or output of any one of the many digital peripherals to any one of these I/O pins. PPS is performed in software, and generally, does not require the device to be reprogrammed. Hardware safeguards are included that prevent accidental or spurious changes to the peripheral mapping once it has been established.

10.7.1 AVAILABLE PINS

The PPS feature is used with a range of up to 22 pins. The number of available pins is dependent on the particular device and its pin count. Pins that support the PPS feature include the designation “RPn” in their full pin designation, where “RP” designates a remappable peripheral and “n” is the remappable pin number. See Table 1-3 and Table 1-4 for pinout options in each package offering.

10.7.2 AVAILABLE PERIPHERALS

The peripherals managed by the PPS are all digital only peripherals. These include general serial communications (UART and SPI), general purpose timer clock inputs, timer-related peripherals (input capture and output compare) and external interrupt inputs. Also included are the outputs of the comparator module, since these are discrete digital signals.

The PPS module is not applied to I^2C, change notification inputs, RTCC alarm outputs or peripherals with analog inputs. Additionally, the MSSP1 and EUSART1 modules are not routed through the PPS module.

A key difference between pin select and non-pin select peripherals is that pin select peripherals are not associated with a default I/O pin. The peripheral must always be assigned to a specific I/O pin before it can be used. In contrast, non-pin select peripherals are always available on a default pin, assuming that the peripheral is active and not conflicting with another peripheral.

10.7.2.1 Peripheral Pin Select Function Priority

When a pin selectable peripheral is active on a given I/O pin, it takes priority over all other digital I/O and digital communication peripherals associated with the pin. Priority is given regardless of the type of peripheral that is mapped. Pin select peripherals never take priority over any analog functions associated with the pin.

10.7.3 CONTROLLING PERIPHERAL PIN SELECT

PPS features are controlled through two sets of Special Function Registers (SFRs): one to map peripheral inputs and the other to map outputs. Because they are separately controlled, a particular peripheral's input and output (if the peripheral has both) can be placed on any selectable function pin without constraint.

The association of a peripheral to a peripheral selectable pin is handled in two different ways, depending on whether an input or an output is being mapped.

10.7.3.1 Input Mapping

The inputs of the PPS options are mapped on the basis of the peripheral; that is, a control register associated with a peripheral dictates the pin it will be mapped to. The RPINRx registers are used to configure peripheral input mapping (see Register 10-6 through Register 10-23). Each register contains a 5-bit field which is associated with one of the pin selectable peripherals. Programming a given peripheral's bit field with an appropriate 5-bit value maps the RPn pin with that value to that peripheral. For any given device, the valid range of values for any of the bit fields corresponds to the maximum number of Peripheral Pin Selections supported by the device.

TABLE 10-13: SELECTABLE INPUT SOURCES (MAPS INPUT TO FUNCTION)(1)

Input Name	Function Name	Register	Configuration Bits
External Interrupt 1	INT1	RPINR1	INTR1R<4:0>
External Interrupt 2	INT2	RPINR2	INTR2R<4:0>
External Interrupt 3	INT3	RPINR3	INTR3R<4:0>
Timer0 External Clock Input	T0CKI	RPINR4	T0CKR<4:0>
Timer3 External Clock Input	T3CKI	RPINR6	T3CKR<4:0>
Timer5 External Clock Input	T5CKI	RPINR15	T5CKR<4:0>
Input Capture 1	CCP1	RPINR7	IC1R<4:0>
Input Capture 2	CCP2	RPINR8	IC2R<4:0>
Input Capture 3	CCP3	RPINR9	IC3R<4:0>
Timer1 Gate Input	T1G	RPINR12	T1GR<4:0>
Timer3 Gate Input	T3G	RPINR13	T3GR<4:0>
Timer5 Gate Input	T5G	RPINR14	T5GR<4:0>
EUSART2 Asynchronous Receive/Synchronous Receive	RX2/DT2	RPINR16	RX2DT2R<4:0>
EUSART2 Asynchronous Clock Input	CK2	RPINR17	CK2R<4:0>
SPI2 Data Input	SDI2	RPINR21	SDI2R<4:0>
SPI2 Clock Input	SCK2IN	RPINR22	SCK2R<4:0>
SPI2 Slave Select Input	SS2IN	RPINR23	SS2R<4:0>
PWM Fault Input	FLT0	RPINR24	OCFAR<4:0>

Note 1: Unless otherwise noted, all inputs use the Schmitt Trigger input buffers.

10.7.3.2 Output Mapping

In contrast to inputs, the outputs of the PPS options are mapped on the basis of the pin. In this case, a control register associated with a particular pin dictates the peripheral output to be mapped. The RPORx registers are used to control output mapping. The value of the bit field corresponds to one of the peripherals and that peripheral's output is mapped to the pin (see Table 10-14).

Because of the mapping technique, the list of peripherals for output mapping also includes a null value of '`00000`'. This permits any given pin to remain disconnected from the output of any of the pin selectable peripherals.

TABLE 10-14: SELECTABLE OUTPUT SOURCES (MAPS FUNCTION TO OUTPUT)

Function	Output Function Number[1]	Output Name
`NULL`	0	`NULL`[2]
C1OUT	1	Comparator 1 Output
C2OUT	2	Comparator 2 Output
C3OUT	3	Comparator 3 Output
TX2/CK2	6	EUSART2 Asynchronous Transmit/Asynchronous Clock Output
DT2	7	EUSART2 Synchronous Transmit
SDO2	10	SPI2 Data Output
SCK2	11	SPI2 Clock Output
SSDMA	12	SPI DMA Slave Select
ULPOUT	13	Ultra Low-Power Wake-up Event
CCP1/P1A	14	ECCP1 Compare or PWM Output Channel A
P1B	15	ECCP1 Enhanced PWM Output, Channel B
P1C	16	ECCP1 Enhanced PWM Output, Channel C
P1D	17	ECCP1 Enhanced PWM Output, Channel D
CCP2/P2A	18	ECCP2 Compare or PWM Output
P2B	19	ECCP2 Enhanced PWM Output, Channel B
P2C	20	ECCP2 Enhanced PWM Output, Channel C
P2D	21	ECCP2 Enhanced PWM Output, Channel D
CCP3/P3A	22	ECCP3 Compare or PWM Output
P3B	23	ECCP3 Enhanced PWM Output, Channel B
P3C	24	ECCP3 Enhanced PWM Output, Channel C
P3D	25	ECCP3 Enhanced PWM Output, Channel D

Note 1: Value assigned to the RP<4:0> pins corresponds to the peripheral output function number.

2: The `NULL` function is assigned to all RPn outputs at device Reset and disables the RPn output function.

10.7.3.3 Mapping Limitations

The control schema of the PPS is extremely flexible. Other than systematic blocks that prevent signal contention caused by two physical pins being configured as the same functional input or two functional outputs configured as the same pin, there are no hardware enforced lockouts. The flexibility extends to the point of allowing a single input to drive multiple peripherals or a single functional output to drive multiple output pins.

10.7.4 CONTROLLING CONFIGURATION CHANGES

Because peripheral remapping can be changed during run time, some restrictions on peripheral remapping are needed to prevent accidental configuration changes. PIC18F devices include three features to prevent alterations to the peripheral map:

- Control register lock sequence
- Continuous state monitoring
- Configuration bit remapping lock

10.7.4.1 Control Register Lock

Under normal operation, writes to the RPINRx and RPORx registers are not allowed. Attempted writes will appear to execute normally, but the contents of the registers will remain unchanged. To change these registers, they must be unlocked in hardware. The register lock is controlled by the IOLOCK bit (PPSCON<0>). Setting IOLOCK prevents writes to the control registers; clearing IOLOCK allows writes.

To set or clear IOLOCK, a specific command sequence must be executed:

1. Write 55h to EECON2<7:0>.
2. Write AAh to EECON2<7:0>.
3. Clear (or set) IOLOCK as a single operation.

IOLOCK remains in one state until changed. This allows all of the PPS registers to be configured with a single unlock sequence followed by an update to all control registers, then locked with a second lock sequence.

10.7.4.2 Continuous State Monitoring

In addition to being protected from direct writes, the contents of the RPINRx and RPORx registers are constantly monitored in hardware by shadow registers. If an unexpected change in any of the registers occurs (such as cell disturbances caused by ESD or other external events), a Configuration Mismatch Reset will be triggered.

10.7.4.3 Configuration Bit Pin Select Lock

As an additional level of safety, the device can be configured to prevent more than one write session to the RPINRx and RPORx registers. The IOL1WAY (CONFIG3H<0>) Configuration bit blocks the IOLOCK bit from being cleared after it has been set once. If IOLOCK remains set, the register unlock procedure will not execute and the PPS Control registers cannot be written to. The only way to clear the bit and re-enable peripheral remapping is to perform a device Reset.

In the default (unprogrammed) state, IOL1WAY is set, restricting users to one write session. Programming IOL1WAY allows users unlimited access (with the proper use of the unlock sequence) to the PPS registers.

10.7.5 CONSIDERATIONS FOR PERIPHERAL PIN SELECTION

The ability to control Peripheral Pin Selection introduces several considerations into application design that could be overlooked. This is particularly true for several common peripherals that are available only as remappable peripherals.

The main consideration is that the PPS is not available on default pins in the device's default (Reset) state. Since all RPINRx registers reset to '`11111`' and all RPORx registers reset to '`00000`', all PPS inputs are tied to RP31 and all PPS outputs are disconnected.

Note: In tying PPS inputs to RP31, RP31 does not have to exist on a device for the registers to be reset to it.

This situation requires the user to initialize the device with the proper peripheral configuration before any other application code is executed. Since the IOLOCK bit resets in the unlocked state, it is not necessary to execute the unlock sequence after the device has come out of Reset.

For application safety, however, it is best to set IOLOCK and lock the configuration after writing to the control registers.

The unlock sequence is timing-critical. Therefore, it is recommended that the unlock sequence be executed as an assembly language routine with interrupts temporarily disabled. If the bulk of the application is written in C or another high-level language, the unlock sequence should be performed by writing in-line assembly.

Choosing the configuration requires the review of all PPSs and their pin assignments, especially those that will not be used in the application. In all cases, unused pin selectable peripherals should be disabled completely. Unused peripherals should have their inputs assigned to an unused RPn pin function. I/O pins with unused RPn functions should be configured with the null peripheral output.

The assignment of a peripheral to a particular pin does not automatically perform any other configuration of the pin's I/O circuitry. In theory, this means adding a pin selectable output to a pin may mean inadvertently driving an existing peripheral input when the output is driven. Users must be familiar with the behavior of other fixed peripherals that share a remappable pin and know when to enable or disable them. To be safe, fixed digital peripherals that share the same pin should be disabled when not in use.

Along these lines, configuring a remappable pin for a specific peripheral does not automatically turn that feature on. The peripheral must be specifically configured for operation and enabled, as if it were tied to a fixed pin. Where this happens in the application code (immediately following device Reset and peripheral configuration or inside the main application routine) depends on the peripheral and its use in the application.

A final consideration is that the PPS functions neither override analog inputs nor reconfigure pins with analog functions for digital I/O. If a pin is configured as an analog input on device Reset, it must be explicitly reconfigured as digital I/O when used with a PPS.

Example 10-7 provides a configuration for bidirectional communication with flow control using EUSART2. The following input and output functions are used:

- Input Function RX2
- Output Function TX2

EXAMPLE 10-7: CONFIGURING EUSART2 INPUT AND OUTPUT FUNCTIONS

```
;*************************************
; Unlock Registers
;*************************************
MOVLB   0x0E    ; PPS registers in BANK 14
BCF     INTCON, GIE ; Disable interrupts
MOVLW   0x55
MOVWF   EECON2, 0
MOVLW   0xAA
MOVWF   EECON2, 0
; Turn off PPS Write Protect
BCF     PPSCON, IOLOCK, BANKED

;***************************
; Configure Input Functions
; (See Table 10-13)
;***************************
  ;***************************
  ; Assign RX2 To Pin RP0
  ;***************************
MOVLW   0x00
MOVWF   RPINR16, BANKED

;***************************
; Configure Output Functions
; (See Table 10-14)
;***************************
  ;***************************
  ; Assign TX2 To Pin RP1
  ;***************************
MOVLW   0x06
MOVWF   RPOR1, BANKED

;*************************************
; Lock Registers
;*************************************
BCF     INTCON, GIE
MOVLW   0x55
MOVWF   EECON2, 0
MOVLW   0xAA
MOVWF   EECON2, 0

; Write Protect PPS
BSF     PPSCON, IOLOCK, BANKED
```

Note: If the Configuration bit, IOL1WAY = `1`, once the IOLOCK bit is set, it cannot be cleared, preventing any future RP register changes. The IOLOCK bit is cleared back to '`0`' on any device Reset.

10.7.6 PERIPHERAL PIN SELECT REGISTERS

The PIC18F47J13 Family of devices implements a total of 37 registers for remappable peripheral configuration of 44-pin devices. The 28-pin devices have 31 registers for remappable peripheral configuration.

Note: Input and output register values can only be changed if IOLOCK (PPSCON<0>) = `0`. See Example 10-7 for a specific command sequence.

REGISTER 10-5: PPSCON: PERIPHERAL PIN SELECT INPUT REGISTER 0 (BANKED EBFh)[1]

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	IOLOCK
bit 7							bit 0

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

bit 7-1 **Unimplemented:** Read as ‘`0`’

bit 0 **IOLOCK:** I/O Lock Enable bit

`1` = I/O lock is active, RPORx and RPINRx registers are write-protected
`0` = I/O lock is not active, pin configurations can be changed

Note 1: Register values can only be changed if IOLOCK (PPSCON<0>) = `0`.

REGISTER 10-6: RPINR1: PERIPHERAL PIN SELECT INPUT REGISTER 1 (BANKED EE1h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	INTR1R4	INTR1R3	INTR1R2	INTR1R1	INTR1R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0 **INTR1R<4:0>:** Assign External Interrupt 1 (INT1) to the Corresponding RPn Pin bits

REGISTER 10-7: RPINR2: PERIPHERAL PIN SELECT INPUT REGISTER 2 (BANKED EE2h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	INTR2R4	INTR2R3	INTR2R2	INTR2R1	INTR2R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0 **INTR2R<4:0>:** Assign External Interrupt 2 (INT2) to the Corresponding RPn Pin bits

REGISTER 10-8: RPINR3: PERIPHERAL PIN SELECT INPUT REGISTER 3 (BANKED EE3h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	INTR3R4	INTR3R3	INTR3R2	INTR3R1	INTR3R0
bit 7							bit 0

Legend: R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0
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 4-0	**INTR3R<4:0>:** Assign External Interrupt 3 (INT3) to the Corresponding RPn Pin bits

REGISTER 10-9: RPINR4: PERIPHERAL PIN SELECT INPUT REGISTER 4 (BANKED EE4h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	T0CKR4	T0CKR3	T0CKR2	T0CKR1	T0CKR0
bit 7							bit 0

Legend: R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0
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 4-0	**T0CKR<4:0>:** Timer0 External Clock Input (T0CKI) to the Corresponding RPn Pin bits

REGISTER 10-10: RPINR6: PERIPHERAL PIN SELECT INPUT REGISTER 6 (BANKED EE6h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	T3CKR4	T3CKR3	T3CKR2	T3CKR1	T3CKR0
bit 7							bit 0

Legend: R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0
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 4-0	**T3CKR<4:0>:** Timer3 External Clock Input (T3CKI) to the Corresponding RPn Pin bits

REGISTER 10-11: RPINR7: PERIPHERAL PIN SELECT INPUT REGISTER 7 (BANKED EE8h)

U-0	U-0	U-0	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1
—	—	—	IC1R4	IC1R3	IC1R2	IC1R1	IC1R0
bit 7							bit 0

Legend: R/$\overline{\text{W}}$ = Readable bit, Writable bit if IOLOCK = 0

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 4-0	**IC1R<4:0>:** Assign Input Capture 1 (ECCP1) to the Corresponding RPn Pin bits

REGISTER 10-12: RPINR8: PERIPHERAL PIN SELECT INPUT REGISTER 8 (BANKED EE9h)

U-0	U-0	U-0	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1
—	—	—	IC2R4	IC2R3	IC2R2	IC2R1	IC2R0
bit 7							bit 0

Legend: R/$\overline{\text{W}}$ = Readable bit, Writable bit if IOLOCK = 0

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 4-0	**IC2R<4:0>:** Assign Input Capture 2 (ECCP2) to the Corresponding RPn Pin bits

REGISTER 10-13: RPINR9: PERIPHERAL PIN SELECT INPUT REGISTER 9 (BANKED EEAh)

U-0	U-0	U-0	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1
—	—	—	IC3R4	IC3R3	IC3R2	IC3R1	IC3R0
bit 7							bit 0

Legend: R/$\overline{\text{W}}$ = Readable bit, Writable bit if IOLOCK = 0

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 4-0	**IC3R<4:0>:** Assign Input Capture 3 (ECCP3) to the Corresponding RPn Pin bits

REGISTER 10-14: RPINR12: PERIPHERAL PIN SELECT INPUT REGISTER 12 (BANKED EF2h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	T1GR4	T1GR3	T1GR2	T1GR1	T1GR0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0	**T1GR<4:0>:** Timer1 Gate Input (T1G) to the Corresponding RPn Pin bits

REGISTER 10-15: RPINR13: PERIPHERAL PIN SELECT INPUT REGISTER 13 (BANKED EF3h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	T3GR4	T3GR3	T3GR2	T3GR1	T3GR0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0	**T3GR<4:0>:** Timer3 Gate Input (T3G) to the Corresponding RPn Pin bits

REGISTER 10-16: RPINR14: PERIPHERAL PIN SELECT INPUT REGISTER 14 (BANKED EF4h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	T5GR4	T5GR3	T5GR2	T5GR1	T5GR0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0	**T5GR<4:0>:** Timer5 Gate Input (T5G) to the Corresponding RPn Pin bits

REGISTER 10-17: RPINR16: PERIPHERAL PIN SELECT INPUT REGISTER 16 (BANKED EF7h)

U-0	U-0	U-0	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1
—	—	—	RX2DT2R4	RX2DT2R3	RX2DT2R2	RX2DT2R1	RX2DT2R0
bit 7							bit 0

Legend:	R/$\overline{\text{W}}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RX2DT2R<4:0>:** EUSART2 Synchronous/Asynchronous Receive (RX2/DT2) to the Corresponding RPn Pin bits

REGISTER 10-18: RPINR15: PERIPHERAL PIN SELECT INPUT REGISTER 15 (BANKED EE7h)

U-0	U-0	U-0	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1
—	—	—	T5CKR4	T5CKR3	T5CKR2	T5CKR1	T5CKR0
bit 7							bit 0

Legend:	R/$\overline{\text{W}}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**T5CKR<4:0>:** Timer5 External Clock Input (T5CKI) to the Corresponding RPn Pin bits

REGISTER 10-19: RPINR17: PERIPHERAL PIN SELECT INPUT REGISTER 17 (BANKED EF8h)

U-0	U-0	U-0	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1	R/$\overline{\text{W}}$-1
—	—	—	CK2R4	CK2R3	CK2R2	CK2R1	CK2R0
bit 7							bit 0

Legend:	R/$\overline{\text{W}}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**CK2R<4:0>:** EUSART2 Clock Input (CK2) to the Corresponding RPn Pin bits

REGISTER 10-20: RPINR21: PERIPHERAL PIN SELECT INPUT REGISTER 21 (BANKED EFCh)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	SDI2R4	SDI2R3	SDI2R2	SDI2R1	SDI2R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0 **SDI2R<4:0>:** Assign SPI2 Data Input (SDI2) to the Corresponding RPn Pin bits

REGISTER 10-21: RPINR22: PERIPHERAL PIN SELECT INPUT REGISTER 22 (BANKED EFDh)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	SCK2R4	SCK2R3	SCK2R2	SCK2R1	SCK2R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0 **SCK2R<4:0>:** Assign SPI2 Clock Input (SCK2) to the Corresponding RPn Pin bits

REGISTER 10-22: RPINR23: PERIPHERAL PIN SELECT INPUT REGISTER 23 (BANKED EFEh)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	SS2R4	SS2R3	SS2R2	SS2R1	SS2R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0 **SS2R<4:0>:** Assign SPI2 Slave Select Input (SS2) to the Corresponding RPn Pin bits

REGISTER 10-23: RPINR24: PERIPHERAL PIN SELECT INPUT REGISTER 24 (BANKED EFFh)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	OCFAR4	OCFAR3	OCFAR2	OCFAR1	OCFAR0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**OCFAR<4:0>:** Assign PWM Fault Input (FLT0) to the Corresponding RPn Pin bits

REGISTER 10-24: RPOR0: PERIPHERAL PIN SELECT OUTPUT REGISTER 0 (BANKED EC1h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP0R4	RP0R3	RP0R2	RP0R1	RP0R0
bit 7							bit 0

Legend: R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`
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 4-0	**RP0R<4:0>:** Peripheral Output Function is Assigned to RP0 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-25: RPOR1: PERIPHERAL PIN SELECT OUTPUT REGISTER 1 (BANKED EC7h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP1R4	RP1R3	RP1R2	RP1R1	RP1R0
bit 7							bit 0

Legend: R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`
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 4-0	**RP1R<4:0>:** Peripheral Output Function is Assigned to RP1 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-26: RPOR2: PERIPHERAL PIN SELECT OUTPUT REGISTER 2 (BANKED EC3h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP2R4	RP2R3	RP2R2	RP2R1	RP2R0
bit 7							bit 0

Legend: R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`
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 4-0	**RP2R<4:0>:** Peripheral Output Function is Assigned to RP2 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-27: RPOR3: PERIPHERAL PIN SELECT OUTPUT REGISTER 3 (BANKED EC3h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP3R4	RP3R3	RP3R2	RP3R1	RP3R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP3R<4:0>:** Peripheral Output Function is Assigned to RP3 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-28: RPOR4: PERIPHERAL PIN SELECT OUTPUT REGISTER 4 (BANKED EC4h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP4R4	RP4R3	RP4R2	RP4R1	RP4R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP4R<4:0>:** Peripheral Output Function is Assigned to RP4 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-29: RPOR5: PERIPHERAL PIN SELECT OUTPUT REGISTER 5 (BANKED EC5h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP5R4	RP5R3	RP5R2	RP5R1	RP5R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP5R<4:0>:** Peripheral Output Function is Assigned to RP5 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-30: RPOR6: PERIPHERAL PIN SELECT OUTPUT REGISTER 6 (BANKED EC6h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP6R4	RP6R3	RP6R2	RP6R1	RP6R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0	**RP6R<4:0>:** Peripheral Output Function is Assigned to RP6 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-31: RPOR7: PERIPHERAL PIN SELECT OUTPUT REGISTER 7 (BANKED EC7h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP7R4	RP7R3	RP7R2	RP7R1	RP7R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0	**RP7R<4:0>:** Peripheral Output Function is Assigned to RP7 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-32: RPOR8: PERIPHERAL PIN SELECT OUTPUT REGISTER 8 (BANKED EC8h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP8R4	RP8R3	RP8R2	RP8R1	RP8R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0	**RP8R<4:0>:** Peripheral Output Function is Assigned to RP8 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-33: RPOR9: PERIPHERAL PIN SELECT OUTPUT REGISTER 9 (BANKED EC9h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP9R4	RP9R3	RP9R2	RP9R1	RP9R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0 **RP9R<4:0>:** Peripheral Output Function is Assigned to RP9 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-34: RPOR10: PERIPHERAL PIN SELECT OUTPUT REGISTER 10 (BANKED ECAh)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP10R4	RP10R3	RP10R2	RP10R1	RP10R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0 **RP10R<4:0>:** Peripheral Output Function is Assigned to RP10 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-35: RPOR11: PERIPHERAL PIN SELECT OUTPUT REGISTER 11 (BANKED ECBh)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP11R4	RP11R3	RP11R2	RP11R1	RP11R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0 **RP11R<4:0>:** Peripheral Output Function is Assigned to RP11 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-36: RPOR12: PERIPHERAL PIN SELECT OUTPUT REGISTER 12 (BANKED ECCh)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP12R4	RP12R3	RP12R2	RP12R1	RP12R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP12R<4:0>:** Peripheral Output Function is Assigned to RP12 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-37: RPOR13: PERIPHERAL PIN SELECT OUTPUT REGISTER 13 (BANKED ECDh)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP13R4	RP13R3	RP13R2	RP13R1	RP13R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP13R<4:0>:** Peripheral Output Function is Assigned to RP13 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-38: RPOR14: PERIPHERAL PIN SELECT OUTPUT REGISTER 14 (BANKED ECEh)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP14R4	RP14R3	RP14R2	RP14R1	RP14R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP14R<4:0>:** Peripheral Output Function is Assigned to RP14 Output Pin bits (see Table 10-14 for peripheral function numbers)

REGISTER 10-39: RPOR15: PERIPHERAL PIN SELECT OUTPUT REGISTER 15 (BANKED ECFh)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP15R4	RP15R3	RP15R2	RP15R1	RP15R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0 **RP15R<4:0>:** Peripheral Output Function is Assigned to RP15 Output Pin bits
(see Table 10-14 for peripheral function numbers)

REGISTER 10-40: RPOR16: PERIPHERAL PIN SELECT OUTPUT REGISTER 16 (BANKED ED0h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP16R4	RP16R3	RP16R2	RP16R1	RP16R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0 **RP16R<4:0>:** Peripheral Output Function is Assigned to RP16 Output Pin bits
(see Table 10-14 for peripheral function numbers)

REGISTER 10-41: RPOR17: PERIPHERAL PIN SELECT OUTPUT REGISTER 17 (BANKED ED1h)

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP17R4	RP17R3	RP17R2	RP17R1	RP17R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = 0		
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 4-0 **RP17R<4:0>:** Peripheral Output Function is Assigned to RP17 Output Pin bits
(see Table 10-14 for peripheral function numbers)

REGISTER 10-42: RPOR18: PERIPHERAL PIN SELECT OUTPUT REGISTER 18 (BANKED ED2h)

U-0	U-0	U-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0
—	—	—	RP18R4	RP18R3	RP18R2	RP18R1	RP18R0
bit 7							bit 0

Legend: R/$\overline{\text{W}}$ = Readable bit, Writable bit if IOLOCK = 0
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 4-0 **RP18R<4:0>:** Peripheral Output Function is Assigned to RP18 Output Pin bits
(see Table 10-14 for peripheral function numbers)

REGISTER 10-43: RPOR19: PERIPHERAL PIN SELECT OUTPUT REGISTER 19 (BANKED ED3h)(1)

U-0	U-0	U-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0
—	—	—	RP19R4	RP19R3	RP19R2	RP19R1	RP19R0
bit 7							bit 0

Legend: R/$\overline{\text{W}}$ = Readable bit, Writable bit if IOLOCK = 0
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 4-0 **RP19R<4:0>:** Peripheral Output Function is Assigned to RP19 Output Pin bits
(see Table 10-14 for peripheral function numbers)

Note 1: RP19 pins are not available on 28-pin devices.

REGISTER 10-44: RPOR20: PERIPHERAL PIN SELECT OUTPUT REGISTER 20 (BANKED ED4h)(1)

U-0	U-0	U-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0	R/$\overline{\text{W}}$-0
—	—	—	RP20R4	RP20R3	RP20R2	RP20R1	RP20R0
bit 7							bit 0

Legend: R/$\overline{\text{W}}$ = Readable bit, Writable bit if IOLOCK = 0
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 4-0 **RP20R<4:0>:** Peripheral Output Function is Assigned to RP20 Output Pin bits
(see Table 10-14 for peripheral function numbers)

Note 1: RP20 pins are not available on 28-pin devices.

REGISTER 10-45: RPOR21: PERIPHERAL PIN SELECT OUTPUT REGISTER 21 (BANKED ED5h)[1]

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP21R4	RP21R3	RP21R2	RP21R1	RP21R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP21R<4:0>:** Peripheral Output Function is Assigned to RP21 Output Pin bits (see Table 10-14 for peripheral function numbers)

Note 1: RP21 pins are not available on 28-pin devices.

REGISTER 10-46: RPOR22: PERIPHERAL PIN SELECT OUTPUT REGISTER 22 (BANKED ED6h)[1]

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP22R4	RP22R3	RP22R2	RP22R1	RP22R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP22R<4:0>:** Peripheral Output Function is Assigned to RP22 Output Pin bits (see Table 10-14 for peripheral function numbers)

Note 1: RP22 pins are not available on 28-pin devices.

REGISTER 10-47: RPOR23: PERIPHERAL PIN SELECT OUTPUT REGISTER 23 (BANKED ED7h)[1]

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP23R4	RP23R3	RP23R2	RP23R1	RP23R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP23R<4:0>:** Peripheral Output Function is Assigned to RP23 Output Pin bits (see Table 10-14 for peripheral function numbers)

Note 1: RP23 pins are not available on 28-pin devices.

REGISTER 10-48: RPOR24: PERIPHERAL PIN SELECT OUTPUT REGISTER 24 (BANKED ED8h)[1]

U-0	U-0	U-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0	R/$\overline{W}$-0
—	—	—	RP24R4	RP24R3	RP24R2	RP24R1	RP24R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable bit, Writable bit if IOLOCK = `0`		
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 4-0	**RP24R<4:0>:** Peripheral Output Function is Assigned to RP24 Output Pin bits (see Table 10-14 for peripheral function numbers)

Note 1: RP24 pins are not available on 28-pin devices.

11.0 PARALLEL MASTER PORT (PMP)

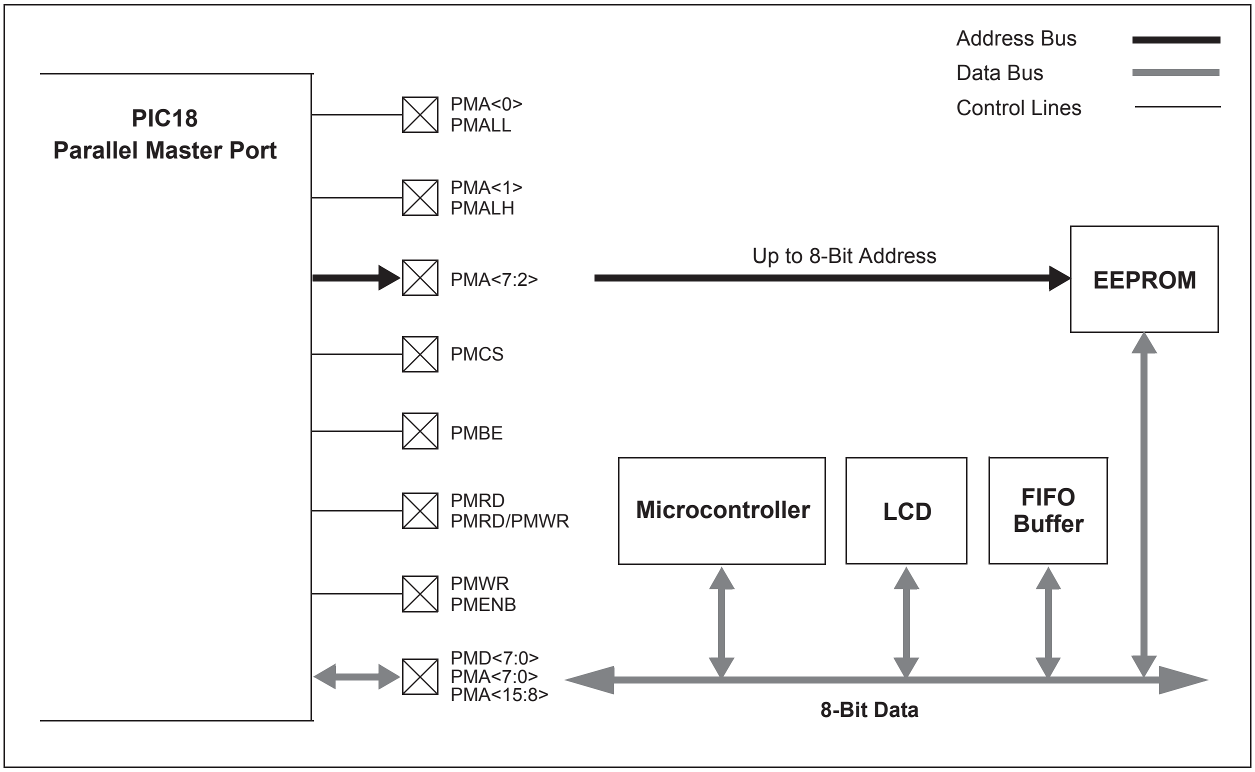
The Parallel Master Port module (PMP) is an 8-bit parallel I/O module, specifically designed to communicate with a wide variety of parallel devices, such as communication peripherals, LCDs, external memory devices and microcontrollers. Because the interface to parallel peripherals varies significantly, the PMP is highly configurable. The PMP module can be configured to serve as either a PMP or as a Parallel Slave Port (PSP).

> **Note:** The PMP module is not implemented on 28-pin devices. It is available only on the PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13.

Key features of the PMP module are:

- Up to 16 bits of Addressing when using Data/Address Multiplexing
- Up to 8 Programmable Address Lines
- One Chip Select Line
- Programmable Strobe Options:
  - Individual Read and Write Strobes or;
  - Read/$\overline{\text{Write}}$ Strobe with Enable Strobe
- Address Auto-Increment/Auto-Decrement
- Programmable Address/Data Multiplexing
- Programmable Polarity on Control Signals
- Legacy Parallel Slave Port Support
- Enhanced Parallel Slave Support:
  - Address Support
  - 4-Byte Deep, Auto-Incrementing Buffer
- Programmable Wait States
- Selectable Input Voltage Levels

FIGURE 11-1: PMP MODULE OVERVIEW


Address Bus
Data Bus
Control Lines
PIC18
Parallel Master Port
PMA<0>
PMALL
PMA<1>
PMALH
PMA<7:2>
PMCS
PMBE
PMRD
PMRD/PMWR
PMWR
PMENB
PMD<7:0>
PMA<7:0>
PMA<15:8>
Up to 8-Bit Address
EEPROM
Microcontroller
LCD
FIFO
Buffer
8-Bit Data

11.1 Module Registers

The PMP module has a total of 14 Special Function Registers (SFRs) for its operation, plus one additional register to set configuration options. Of these, eight registers are used for control and six are used for PMP data transfer.

11.1.1 CONTROL REGISTERS

The eight PMP Control registers are:

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

The PMCON registers (Register 11-1 and Register 11-2) control basic module operations, including turning the module on or off. They also configure address multiplexing and control strobe configuration.

The PMMODE registers (Register 11-3 and Register 11-4) configure the various Master and Slave modes, the data width and interrupt generation.

The PMEH and PMEL registers (Register 11-5 and Register 11-6) configure the module's operation at the hardware (I/O pin) level.

The PMSTAT registers (Register 11-5 and Register 11-6) provide status flags for the module's input and output buffers, depending on the operating mode.

REGISTER 11-1: PMCONH: PARALLEL PORT CONTROL REGISTER HIGH BYTE (BANKED F5Fh)[1]

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PMPEN	—	—	ADRMUX1	ADRMUX0	PTBEEN	PTWREN	PTRDEN
bit 7							bit 0

Legend:

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

bit 7	**PMPEN:** Parallel Master Port Enable bit 1 = PMP enabled 0 = PMP disabled, no off-chip access performed
bit 6-5	**Unimplemented:** Read as '0'
bit 4-3	**ADRMUX<1:0>:** Address/Data Multiplexing Selection bits 11 = Reserved 10 = All 16 bits of address are multiplexed on PMD<7:0> pins 01 = Lower 8 bits of address are multiplexed on PMD<7:0> pins (only eight bits of address are available in this mode) 00 = Address and data appear on separate pins (only eight bits of address are available in this mode)
bit 2	**PTBEEN:** Byte Enable Port Enable bit (16-Bit Master mode) 1 = PMBE port enabled 0 = PMBE port disabled
bit 1	**PTWREN:** Write Enable Strobe Port Enable bit 1 = PMWR/PMENB port enabled 0 = PMWR/PMENB port disabled
bit 0	**PTRDEN:** Read/Write Strobe Port Enable bit 1 = PMRD/PMWR port enabled 0 = PMRD/PMWR port disabled

Note 1: This register is only available on 44-pin devices.

REGISTER 11-2: PMCONL: PARALLEL PORT CONTROL REGISTER LOW BYTE (BANKED F5Eh)(1)

R/W-0	R/W-0	R/W-0(2)	U-0	R/W-0(2)	R/W-0	R/W-0	R/W-0
CSF1	CSF0	ALP	—	CS1P	BEP	WRSP	RDSP
bit 7							bit 0

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

bit 7-6 **CSF<1:0>:** Chip Select Function bits

`11` = Reserved
`10` = Chip select function is enabled and PMCS acts as chip select (in Master mode). Up to 13 address bits only can be generated.
`01` = Reserved
`00` = Chip select function is disabled (in Master mode). All 16 address bits can be generated.

bit 5 **ALP:** Address Latch Polarity bit(2)

`1` = Active-high (PMALL and PMALH)
`0` = Active-low ($\overline{\text{PMALL}}$ and $\overline{\text{PMALH}}$)

bit 4 **Unimplemented:** Maintain as ‘`0`’

bit 3 **CS1P:** Chip Select Polarity bit(2)

`1` = Active-high (PMCS)
`0` = Active-low ($\overline{\text{PMCS}}$)

bit 2 **BEP:** Byte Enable Polarity bit

`1` = Byte enable active-high (PMBE)
`0` = Byte enable active-low ($\overline{\text{PMBE}}$)

bit 1 **WRSP:** Write Strobe Polarity bit

For Slave modes and Master Mode 2 (PMMODEH<1:0> = `00,01,10`):
`1` = Write strobe active-high (PMWR)
`0` = Write strobe active-low ($\overline{\text{PMWR}}$)

For Master Mode 1 (PMMODEH<1:0> = `11`):
`1` = Enable strobe active-high (PMENB)
`0` = Enable strobe active-low ($\overline{\text{PMENB}}$)

bit 0 **RDSP:** Read Strobe Polarity bit

For Slave modes and Master Mode 2 (PMMODEH<1:0> = `00,01,10`):
`1` = Read strobe active-high (PMRD)
`0` = Read strobe active-low ($\overline{\text{PMRD}}$)

For Master Mode 1 (PMMODEH<1:0> = `11`):
`1` = Read/write strobe active-high (PMRD/PMWR)
`0` = Read/write strobe active-low ($\overline{\text{PMRD/PMWR}}$)

Note 1: This register is only available on 44-pin devices.
2: These bits have no effect when their corresponding pins are used as address lines.

REGISTER 11-3: PMMODEH: PARALLEL PORT MODE REGISTER HIGH BYTE (BANKED F5Dh)(1)

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUSY	IRQM1	IRQM0	INCM1	INCM0	MODE16	MODE1	MODE0
bit 7							bit 0

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

bit 7 **BUSY:** Busy bit (Master mode only)
`1` = Port is busy
`0` = Port is not busy

bit 6-5 **IRQM<1:0>:** Interrupt Request Mode bits
`11` = Interrupt generated when Read Buffer 3 is read or Write Buffer 3 is written (Buffered PSP mode) or on a read or write operation when PMA<1:0> = `11` (Addressable PSP mode only)
`10` = No interrupt generated, processor stall activated
`01` = Interrupt generated at the end of the read/write cycle
`00` = No interrupt generated

bit 4-3 **INCM<1:0>:** Increment Mode bits
`11` = PSP read and write buffers auto-increment (Legacy PSP mode only)
`10` = Decrement ADDR<15,13:0> by 1 every read/write cycle
`01` = Increment ADDR<15,13:0> by 1 every read/write cycle
`00` = No increment or decrement of address

bit 2 **MODE16:** 8/16-Bit Mode bit
`1` = 16-bit mode: Data register is 16 bits; a read or write to the Data register invokes two 8-bit transfers
`0` = 8-bit mode: Data register is 8 bits; a read or write to the Data register invokes one 8-bit transfer

bit 1-0 **MODE<1:0>:** Parallel Port Mode Select bits
`11` = Master Mode 1 (PMCS, PMRD/PMWR, PMENB, PMBE, PMA<x:0> and PMD<7:0>)
`10` = Master Mode 2 (PMCS, PMRD, PMWR, PMBE, PMA<x:0> and PMD<7:0>)
`01` = Enhanced PSP, control signals (PMRD, PMWR, PMCS, PMD<7:0> and PMA<1:0>)
`00` = Legacy Parallel Slave Port, control signals (PMRD, PMWR, PMCS and PMD<7:0>)

Note 1: This register is only available on 44-pin devices.

REGISTER 11-4: PMMODEL: PARALLEL PORT MODE REGISTER LOW BYTE (BANKED F5Ch)(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
WAITB1(2)	WAITB0(2)	WAITM3	WAITM2	WAITM1	WAITM0	WAITE1(2)	WAITE0(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 **WAITB<1:0>:** Data Setup to Read/Write Wait State Configuration bits(2)

`11` = Data Wait of 4 T_{CY}; multiplexed address phase of 4 T_{CY}
`10` = Data Wait of 3 T_{CY}; multiplexed address phase of 3 T_{CY}
`01` = Data Wait of 2 T_{CY}; multiplexed address phase of 2 T_{CY}
`00` = Data Wait of 1 T_{CY}; multiplexed address phase of 1 T_{CY}

bit 5-2 **WAITM<3:0>:** Read to Byte Enable Strobe Wait State Configuration bits

`1111` = Wait of additional 15 T_{CY}
.
.
.
`0001` = Wait of additional 1 T_{CY}
`0000` = No additional Wait cycles (operation forced into one T_{CY})

bit 1-0 **WAITE<1:0>:** Data Hold After Strobe Wait State Configuration bits(2)

`11` = Wait of 4 T_{CY}
`10` = Wait of 3 T_{CY}
`01` = Wait of 2 T_{CY}
`00` = Wait of 1 T_{CY}

Note 1: This register is only available on 44-pin devices.
2: WAITBx and WAITEx bits are ignored whenever WAITM<3:0> = `0000`.

REGISTER 11-5: PMEH: PARALLEL PORT ENABLE REGISTER HIGH BYTE (BANKED F57h)[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
PTEN15	PTEN14	PTEN13	PTEN12	PTEN11	PTEN10	PTEN9	PTEN8
bit 7							bit 0

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

bit 7-6 **PTEN<15:14>:** PMCS Port Enable bits

1 = PMA<15:14> function as either PMA<15:14> or PMCS
0 = PMA<15:14> function as port I/O

bit 5-0 **PTEN<13:8>:** PMP Address Port Enable bits

1 = PMA<13:8> function as PMP address lines
0 = PMA<13:8> function as port I/O

Note 1: This register is only available on 44-pin devices.

REGISTER 11-6: PMEL: PARALLEL PORT ENABLE REGISTER LOW BYTE (BANKED F56h)[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
PTEN7	PTEN6	PTEN5	PTEN4	PTEN3	PTEN2	PTEN1	PTEN0
bit 7							bit 0

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

bit 7-2 **PTEN<7:2>:** PMP Address Port Enable bits

1 = PMA<7:2> function as PMP address lines
0 = PMA<7:2> function as port I/O

bit 1-0 **PTEN<1:0>:** PMALH/PMALL Strobe Enable bits

1 = PMA<1:0> function as either PMA<1:0> or PMALH and PMALL
0 = PMA<1:0> pads functions as port I/O

Note 1: This register is only available on 44-pin devices.

REGISTER 11-7: PMSTATH: PARALLEL PORT STATUS REGISTER HIGH BYTE (BANKED F55h)[1]

R-0	R/W-0	U-0	U-0	R-0	R-0	R-0	R-0
IBF	IBOV	—	—	IB3F	IB2F	IB1F	IB0F
bit 7							bit 0

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

bit 7 **IBF:** Input Buffer Full Status bit
`1` = All writable Input Buffer registers are full
`0` = Some or all of the writable Input Buffer registers are empty

bit 6 **IBOV:** Input Buffer Overflow Status bit
`1` = A write attempt to a full Input Byte register occurred (must be cleared in software)
`0` = No overflow occurred

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

bit 3-0 **IB<3:0>F:** Input Buffer x Status Full bits
`1` = Input buffer contains data that has not been read (reading the buffer will clear this bit)
`0` = Input buffer does not contain any unread data

Note 1: This register is only available on 44-pin devices.

REGISTER 11-8: PMSTATL: PARALLEL PORT STATUS REGISTER LOW BYTE (BANKED F54h)[1]

R-1	R/W-0	U-0	U-0	R-1	R-1	R-1	R-1
OBE	OBUF	—	—	OB3E	OB2E	OB1E	OB0E
bit 7							bit 0

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

bit 7 **OBE:** Output Buffer Empty Status bit
`1` = All readable Output Buffer registers are empty
`0` = Some or all of the readable Output Buffer registers are full

bit 6 **OBUF:** Output Buffer Underflow Status bit
`1` = A read occurred from an empty Output Byte register (must be cleared in software)
`0` = No underflow occurred

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

bit 3-0 **OB<3:0>E:** Output Buffer x Status Empty bits
`1` = Output buffer is empty (writing data to the buffer will clear this bit)
`0` = Output buffer contains data that has not been transmitted

Note 1: This register is only available on 44-pin devices.

11.1.2 DATA REGISTERS

The PMP module uses eight registers for transferring data into and out of the microcontroller. They are arranged as four pairs to allow the option of 16-bit data operations:

- PMDIN1H and PMDIN1L
- PMDIN2H and PMDIN2L
- PMADDRH/PMDOUT1H and PMADDRL/PMDOUT1L
- PMDOUT2H and PMDOUT2L

The PMDIN1 registers are used for incoming data in Slave modes, and both input and output data in Master modes. The PMDIN2 registers are used for buffering input data in select Slave modes.

The PMADDR/PMDOUT1 registers are actually a single register pair; the name and function are dictated by the module's operating mode. In Master modes, the registers function as the PMADDRH and PMADDRL registers, and contain the address of any incoming or outgoing data. In Slave modes, the registers function as PMDOUT1H and PMDOUT1L, and are used for outgoing data.

PMADDRH differs from PMADDRL in that it can also have limited PMP control functions. When the module is operating in select Master mode configurations, the upper two bits of the register can be used to determine the operation of chip select signals. If these are not used, PMADDR simply functions to hold the upper 8 bits of the address. Register 11-9 provides the function of the individual bits in PMADDRH.

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

11.1.3 PAD CONFIGURATION CONTROL REGISTER

In addition to the module level configuration options, the PMP module can also be configured at the I/O pin for electrical operation. This option allows users to select either the normal Schmitt Trigger input buffer on digital I/O pins shared with the PMP, or use TTL level compatible buffers instead. Buffer configuration is controlled by the PMPTTL bit in the PADCFG1 register.

REGISTER 11-9: PMADDRH: PARALLEL PORT ADDRESS REGISTER HIGH BYTE (MASTER MODES ONLY) (ACCESS F6Fhh)[1]

U0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CS1	Parallel Master Port Address High Byte<13:8>					
bit 7							bit 0

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

bit 7	**Unimplemented:** Read as ‘`0`’
bit 6	**CS1:** Chip Select bit If PMCON<7:6> = `10`: `1` = Chip select is active `0` = Chip select is inactive If PMCON<7:6> = `11` or `00`: Bits function as ADDR<14>.
bit 5-0	**Parallel Master Port Address:** High Byte<13:8> bits

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

REGISTER 11-10: PMADDRL: PARALLEL PORT ADDRESS REGISTER LOW BYTE (MASTER MODES ONLY) (ACCESS F6Eh)[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
Parallel Master Port Address Low Byte<7:0>							
bit 7							bit 0

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

bit 7-0	**Parallel Master Port Address:** Low Byte<7:0> bits

Note 1: In Enhanced Slave mode, PMADDRL functions as PMDOUT1L, one of the Output Data Buffer registers.

11.2 Slave Port Modes

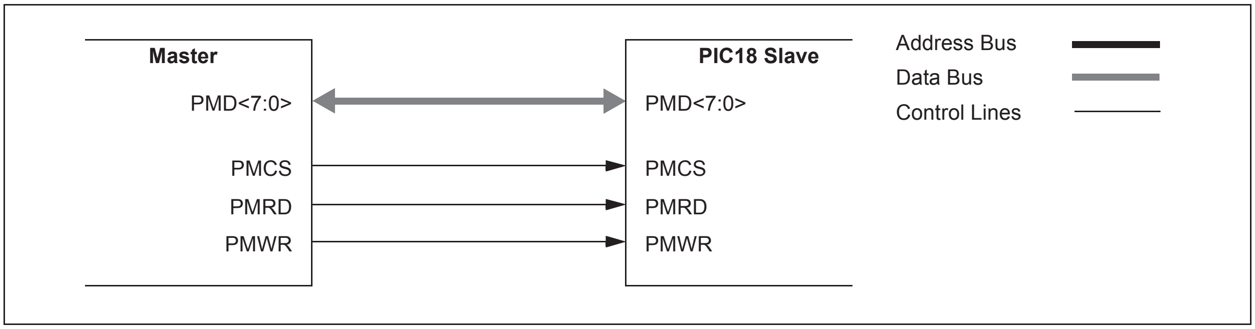
The primary mode of operation for the module is configured using the MODE<1:0> bits in the PMMODEH register. The setting affects whether the module acts as a slave or a master and it determines the usage of the control pins.

11.2.1 LEGACY MODE (PSP)

In Legacy mode (PMMODEH<1:0> = `00` and PMPEN = `1`), the module is configured as a Parallel Slave Port (PSP) with the associated enabled module pins dedicated to the module. In this mode, an external device, such as another microcontroller or microprocessor, can asynchronously read and write data using the 8-bit data bus (PMD<7:0>), the read (PMRD), write (PMWR) and chip select (PMCS) inputs. It acts as a slave on the bus and responds to the read/write control signals.

Figure 11-2 displays the connection of the PSP. When chip select is active and a write strobe occurs (PMCS = `1` and PMWR = `1`), the data from PMD<7:0> is captured into the PMDIN1L register.

FIGURE 11-2: LEGACY PARALLEL SLAVE PORT EXAMPLE



11.2.2 WRITE TO SLAVE PORT

When chip select is active and a write strobe occurs (PMCS = `1` and PMWR = `1`), the data from PMD<7:0> is captured into the lower PMDIN1L register. The PMPIF and IBF flag bits are set when the write ends.The timing for the control signals in Write mode is displayed in Figure 11-3. The polarity of the control signals are configurable.

11.2.3 READ FROM SLAVE PORT

When chip select is active and a read strobe occurs (PMCS = `1` and PMRD = `1`), the data from the PMDOUT1L register (PMDOUT1L<7:0>) is presented on to PMD<7:0>. Figure 11-4 provides the timing for the control signals in Read mode.

FIGURE 11-3: PARALLEL SLAVE PORT WRITE WAVEFORMS

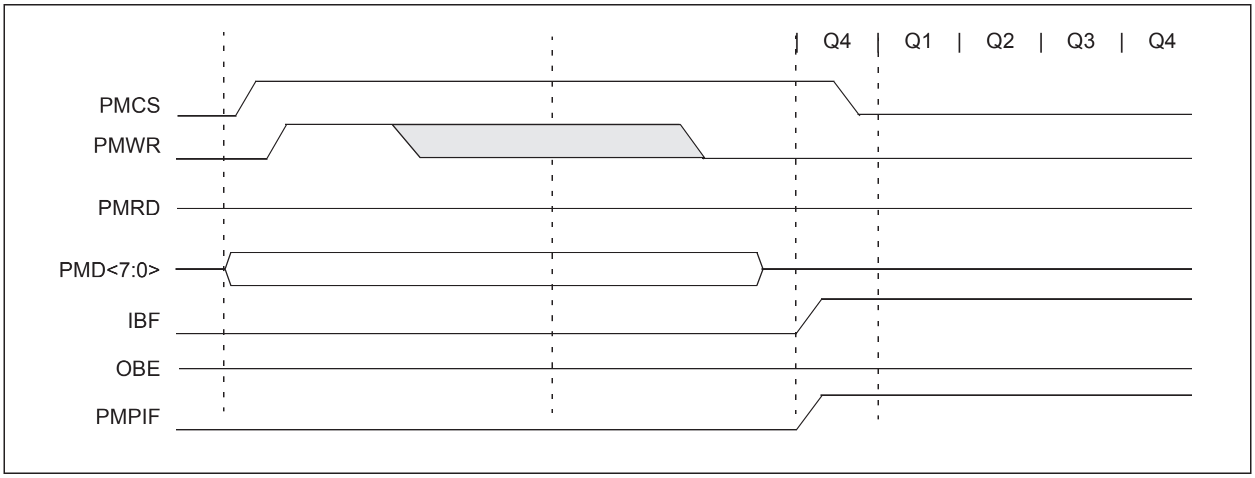
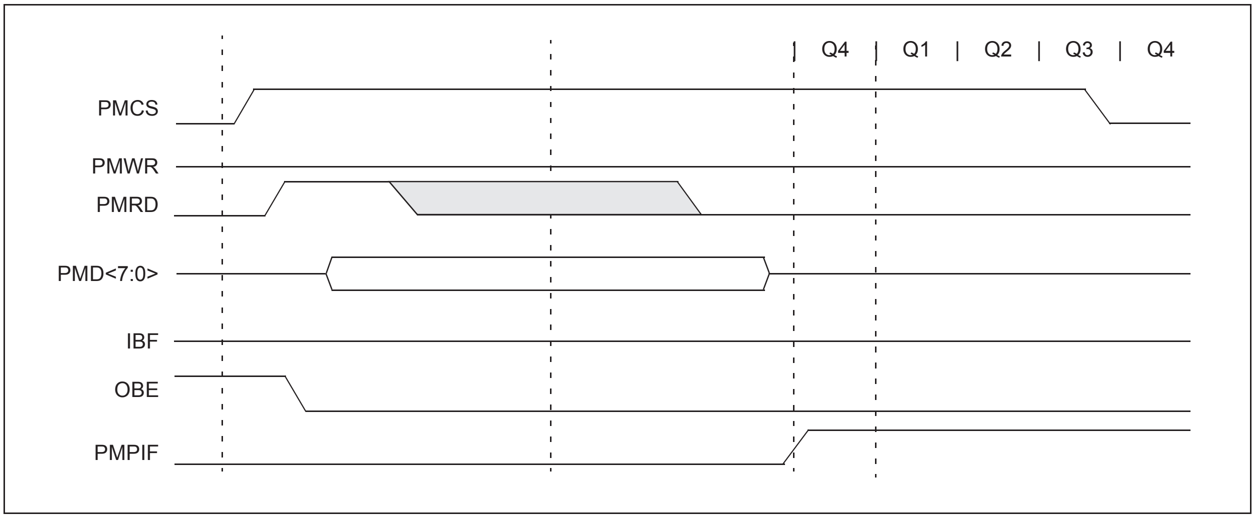

Q4
Q1
Q2
Q3
Q4
PMCS
PMWR
PMRD
PMD<7:0>
IBF
OBE
PMPIF

FIGURE 11-4: PARALLEL SLAVE PORT READ WAVEFORMS


Q4
Q1
Q2
Q3
Q4
PMCS
PMWR
PMRD
PMD<7:0>
IBF
OBE
PMPIF

11.2.4 BUFFERED PARALLEL SLAVE PORT MODE

Buffered Parallel Slave Port mode is functionally identical to the Legacy PSP mode with one exception, the implementation of 4-level read and write buffers. Buffered PSP mode is enabled by setting the INCM bits in the PMMODEH register. If the INCM<1:0> bits are set to '`11`', the PMP module will act as the buffered PSP.

When the Buffered PSP mode is active, the PMDIN1L, PMDIN1H, PMDIN2L and PMDIN2H registers become the write buffers and the PMDOUT1L, PMDOUT1H, PMDOUT2L and PMDOUT2H registers become the read buffers. Buffers are numbered, 0 through 3, starting with the lower byte of PMDIN1L to PMDIN2H as the read buffers and PMDOUT1L to PMDOUT2H as the write buffers.

11.2.4.1 READ FROM SLAVE PORT

For read operations, the bytes will be sent out sequentially, starting with Buffer 0 (PMDOUT1L<7:0>) and ending with Buffer 3 (PMDOUT2H<7:0>) for every read strobe. The module maintains an internal pointer to keep track of which buffer is to be read. Each buffer has a corresponding read status bit, OBxE, in the PMSTATL register. This bit is cleared when a buffer contains data that has not been written to the bus and is set when data is written to the bus. If the current buffer location being read from is empty, a buffer underflow is generated and the Buffer Overflow Flag bit, OBUF, is set. If all four OBxE status bits are set, then the Output Buffer Empty flag (OBE) will also be set.

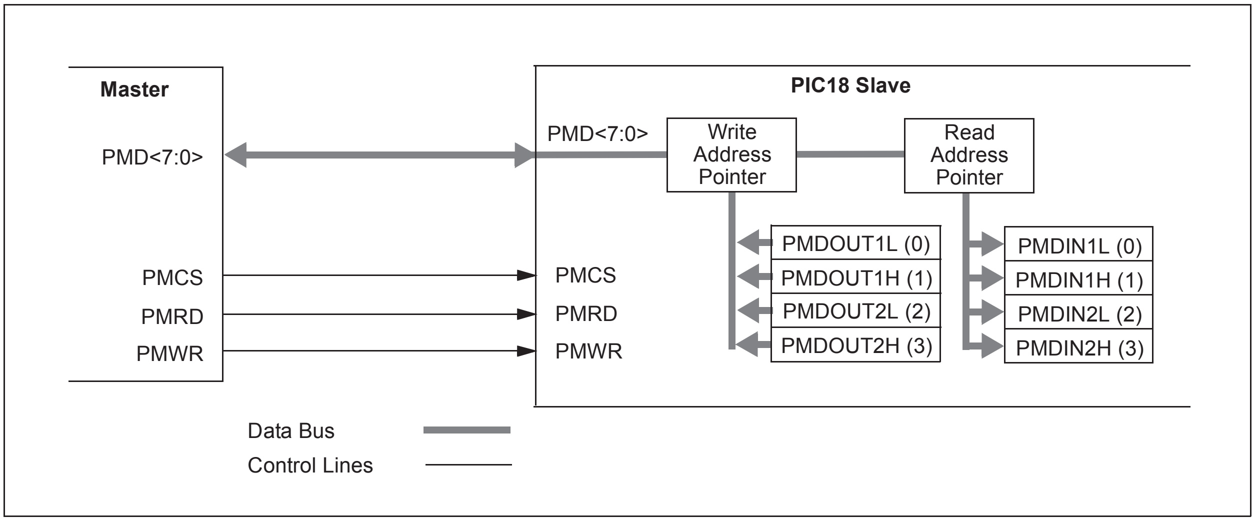
11.2.4.2 WRITE TO SLAVE PORT

For write operations, the data has to be stored sequentially, starting with Buffer 0 (PMDIN1L<7:0>) and ending with Buffer 3 (PMDIN2H<7:0>). As with read operations, the module maintains an internal pointer to the buffer that is to be written next.

The input buffers have their own write status bits: IBxF in the PMSTATH register. The bit is set when the buffer contains unread incoming data and cleared when the data has been read. The flag bit is set on the write strobe. If a write occurs on a buffer when its associated IBxF bit is set, the Buffer Overflow flag, IBOV, is set; any incoming data in the buffer will be lost. If all four IBxF flags are set, the Input Buffer Full Flag (IBF) is set.

In Buffered Slave mode, the module can be configured to generate an interrupt on every read or write strobe (IRQM<1:0> = `01`). It can be configured to generate an interrupt on a read from Read Buffer 3 or a write to Write Buffer 3, which is essentially an interrupt every fourth read or write strobe (RQM<1:0> = `11`). When interrupting every fourth byte for input data, all Input Buffer registers should be read to clear the IBxF flags. If these flags are not cleared, then there is a risk of hitting an overflow condition.

FIGURE 11-5: PARALLEL MASTER/SLAVE CONNECTION BUFFERED EXAMPLE



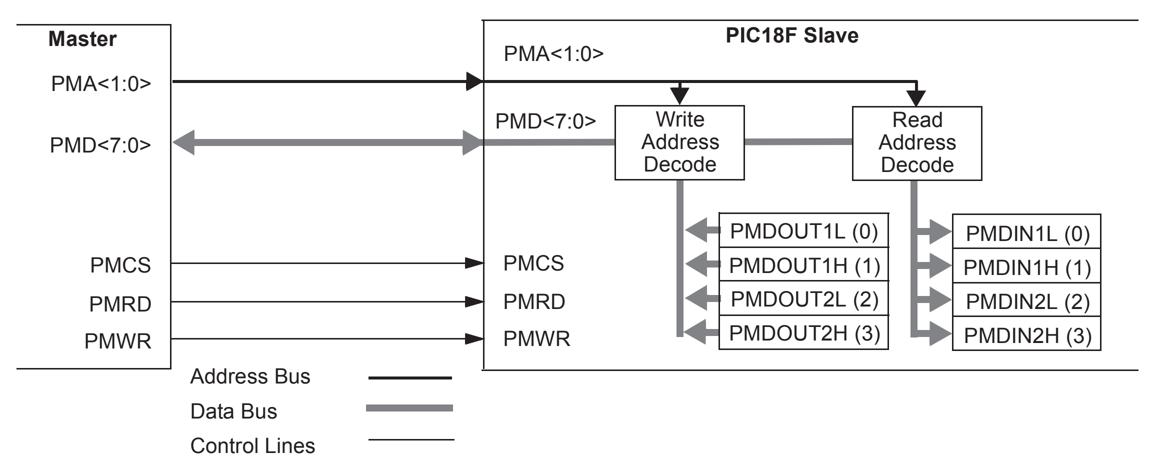
11.2.5 ADDRESSABLE PARALLEL SLAVE PORT MODE

In the Addressable Parallel Slave Port mode (PMMODEH<1:0> = `01`), the module is configured with two extra inputs, PMA<1:0>, which are the Address Lines 1 and 0. This makes the 4-byte buffer space directly addressable as fixed pairs of read and write buffers. As with Legacy Buffered mode, data is output from PMDOUT1L, PMDOUT1H, PMDOUT2L and PMDOUT2H, and is read in PMDIN1L, PMDIN1H, PMDIN2L and PMDIN2H. Table 11-1 provides the buffer addressing for the incoming address to the input and output registers.

TABLE 11-1: SLAVE MODE BUFFER ADDRESSING

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

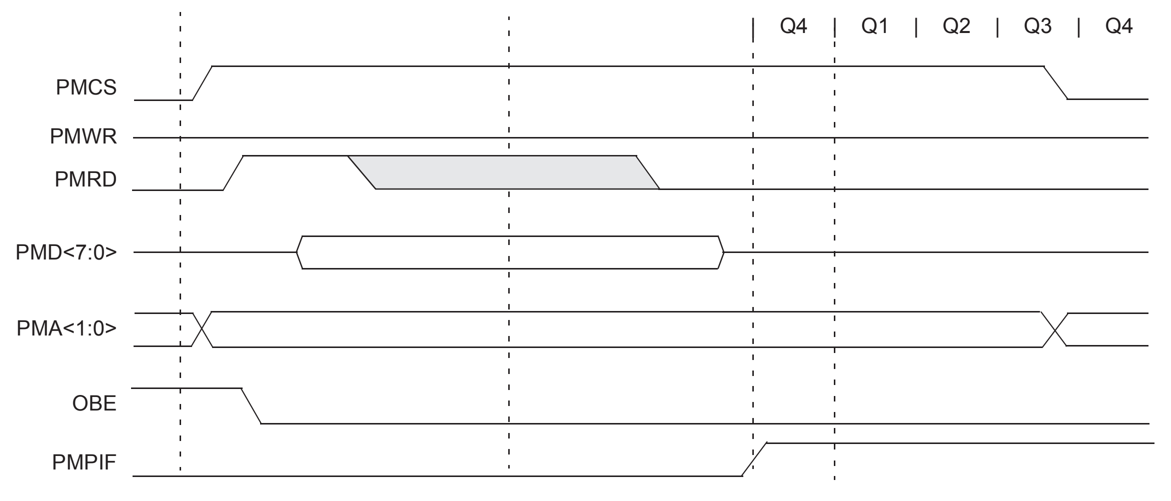
FIGURE 11-6: PARALLEL MASTER/SLAVE CONNECTION ADDRESSED BUFFER EXAMPLE


Master
PMA<1:0>
PMD<7:0>
PMCS
PMRD
PMWR
PIC18F Slave
PMA<1:0>
PMD<7:0>
Write Address Decode
Read Address Decode
PMDOUT1L (0)
PMDOUT1H (1)
PMDOUT2L (2)
PMDOUT2H (3)
PMDIN1L (0)
PMDIN1H (1)
PMDIN2L (2)
PMDIN2H (3)
PMCS
PMRD
PMWR
Address Bus
Data Bus
Control Lines

11.2.5.1 READ FROM SLAVE PORT

When chip select is active and a read strobe occurs (PMCS = `1` and PMRD = `1`), the data from one of the four output bytes is presented onto PMD<7:0>. Which byte is read depends on the 2-bit address placed on ADDR<1:0>. Table 11-1 provides the corresponding output registers and their associated address. When an output buffer is read, the corresponding OBxE bit is set. The OBxE flag bit is set when all the buffers are empty. If any buffer is already empty (OBxE = `1`), the next read to that buffer will generate an OBUF event.

FIGURE 11-7: PARALLEL SLAVE PORT READ WAVEFORMS


Q4
Q1
Q2
Q3
Q4
PMCS
PMWR
PMRD
PMD<7:0>
PMA<1:0>
OBE
PMPIF

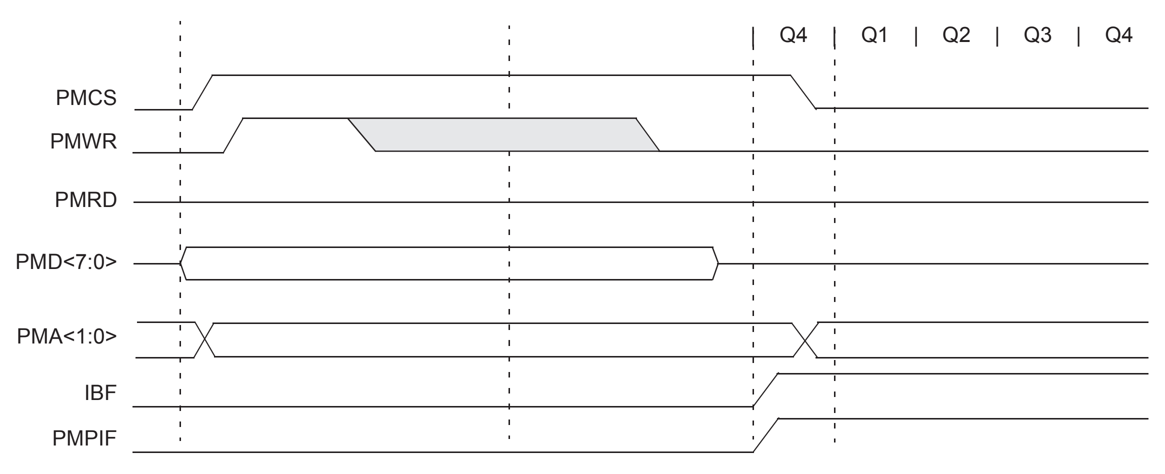
11.2.5.2 WRITE TO SLAVE PORT

When chip select is active and a write strobe occurs (PMCS = `1` and PMWR = `1`), the data from PMD<7:0> is captured into one of the four input buffer bytes. Which byte is written depends on the 2-bit address placed on ADDRL<1:0>.

Table 11-1 provides the corresponding input registers and their associated address.

When an input buffer is written, the corresponding IBxF bit is set. The IBF flag bit is set when all the buffers are written. If any buffer is already written (IBxF = `1`), the next write strobe to that buffer will generate an OBUF event and the byte will be discarded.

FIGURE 11-8: PARALLEL SLAVE PORT WRITE WAVEFORMS



11.3 MASTER PORT MODES

In its Master modes, the PMP module provides an 8-bit data bus, up to 16 bits of address, and all the necessary control signals to operate a variety of external parallel devices, such as memory devices, peripherals and slave microcontrollers. To use the PMP as a master, the module must be enabled (PMPEN = `1`) and the mode must be set to one of the two possible Master modes (PMMODEH<1:0> = `10` or `11`).

Because there are a number of parallel devices with a variety of control methods, the PMP module is designed to be extremely flexible to accommodate a range of configurations. Some of these features include:

- 8-Bit and 16-Bit Data modes on an 8-bit data bus
- Configurable address/data multiplexing
- Up to two chip select lines
- Up to 16 selectable address lines
- Address auto-increment and auto-decrement
- Selectable polarity on all control lines
- Configurable Wait states at different stages of the read/write cycle

11.3.1 PMP AND I/O PIN CONTROL

Multiple control bits are used to configure the presence or absence of control and address signals in the module. These bits are PTBEEN, PTWREN, PTRDEN and PTEN<15:0>. They give the user the ability to conserve pins for other functions and allow flexibility to control the external address. When any one of these bits is set, the associated function is present on its associated pin; when clear, the associated pin reverts to its defined I/O port function.

Setting a PTENx bit will enable the associated pin as an address pin and drive the corresponding data contained in the PMADDR register. Clearing a PTENx bit will force the pin to revert to its original I/O function.

For the pins configured as chip select (PMCS) with the corresponding PTENx bit set, the PTEN0 and PTEN1 bits will also control the PMALL and PMALH signals. When multiplexing is used, the associated address latch signals should be enabled.

11.3.2 READ/WRITE-CONTROL

The PMP module supports two distinct read/write signaling methods. In Master Mode 1, read and write strobes are combined into a single control line, PMRD/PMWR. A second control line, PMENB, determines when a read or write action is to be taken. In Master Mode 2, separate read and write strobes (PMRD and PMWR) are supplied on separate pins.

All control signals (PMRD, PMWR, PMBE, PMENB, PMAL and PMCS) can be individually configured as either positive or negative polarity. Configuration is controlled by separate bits in the PMCONL register. Note that the polarity of control signals that share the same output pin (for example, PMWR and PMENB) are controlled by the same bit; the configuration depends on which Master Port mode is being used.

11.3.3 DATA WIDTH

The PMP supports data widths of both 8 bits and 16 bits. The data width is selected by the MODE16 bit (PMMODEH<2>). Because the data path into and out of the module is only 8 bits wide, 16-bit operations are always handled in a multiplexed fashion, with the Least Significant Byte (LSB) of data being presented first. To differentiate data bytes, the byte enable control strobe, PMBE, is used to signal when the Most Significant Byte (MSB) of data is being presented on the data lines.

11.3.4 ADDRESS MULTIPLEXING

In either of the Master modes (PMMODEH<1:0> = `1x`), the user can configure the address bus to be multiplexed together with the data bus. This is accomplished by using the ADRMUX<1:0> bits (PMCONH<4:3>). There are three address multiplexing modes available. Typical pinout configurations for these modes are displayed in Figure 11-9, Figure 11-10 and Figure 11-11.

In Demultiplexed mode (PMCONH<4:3> = `00`), data and address information are completely separated. Data bits are presented on PMD<7:0>, and address bits are presented on PMADDRH<6:0> and PMADDRL<7:0>.

In Partially Multiplexed mode (PMCONH<4:3> = `01`), the lower eight bits of the address are multiplexed with the data pins on PMD<7:0>. The upper eight bits of the address are unaffected and are presented on PMADDRH<6:0>. The PMA0 pin is used as an address latch and presents the Address Latch Low (PMALL) enable strobe. The read and write sequences are extended by a complete CPU cycle during which the address is presented on the PMD<7:0> pins.

In Fully Multiplexed mode (PMCONH<4:3> = `10`), the entire 16 bits of the address are multiplexed with the data pins on PMD<7:0>. The PMA0 and PMA1 pins are used to present Address Latch Low (PMALL) enable and Address Latch High (PMALH) enable strobes, respectively. The read and write sequences are extended by two complete CPU cycles. During the first cycle, the lower eight bits of the address are presented on the PMD<7:0> pins with the PMALL strobe active. During the second cycle, the upper eight bits of the address are presented on the PMD<7:0> pins with the PMALH strobe active. In the event the upper address bits are configured as chip select pins, the corresponding address bits are automatically forced to ‘`0`’.

FIGURE 11-9: DEMULTIPLEXED ADDRESSING MODE (SEPARATE READ AND WRITE STROBES WITH CHIP SELECT)

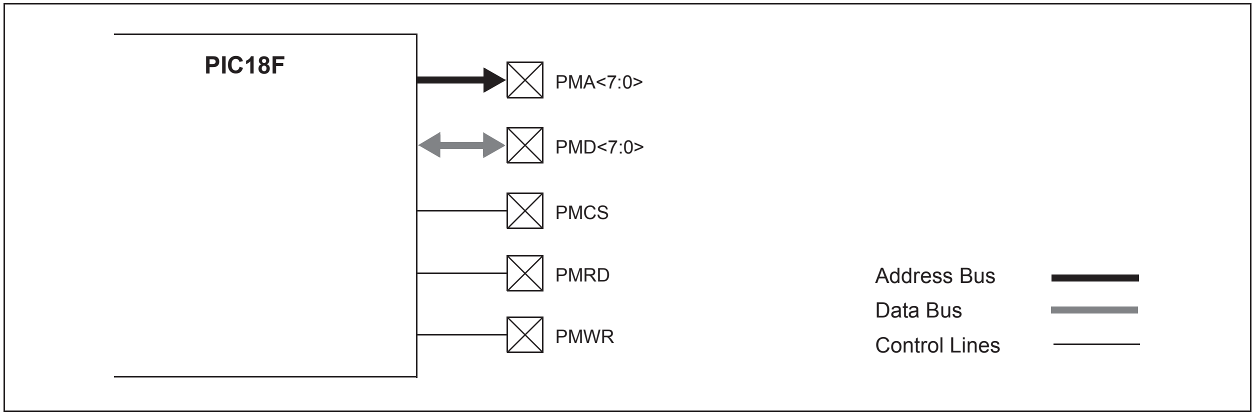

PIC18F
PMA<7:0>
PMD<7:0>
PMCS
PMRD
PMWR
Address Bus
Data Bus
Control Lines

FIGURE 11-10: PARTIALLY MULTIPLEXED ADDRESSING MODE (SEPARATE READ AND WRITE STROBES WITH CHIP SELECT)

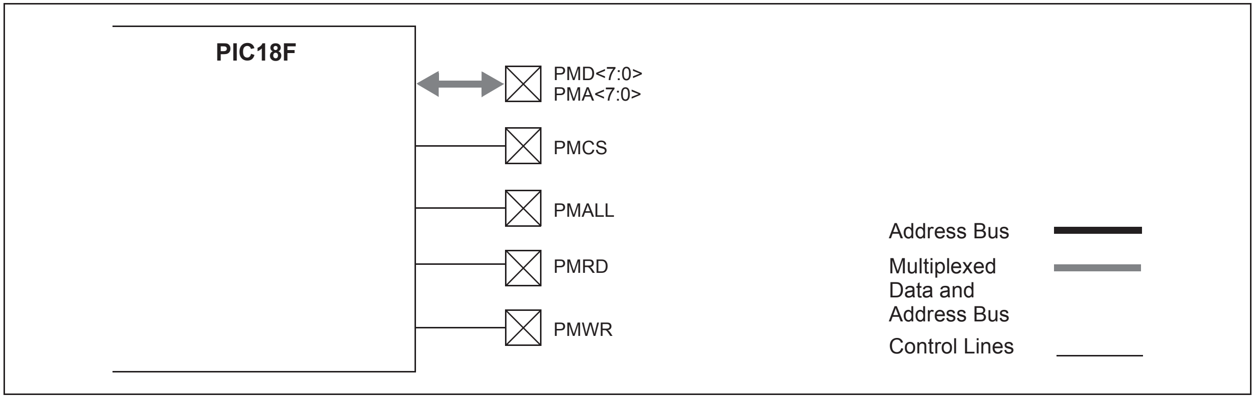
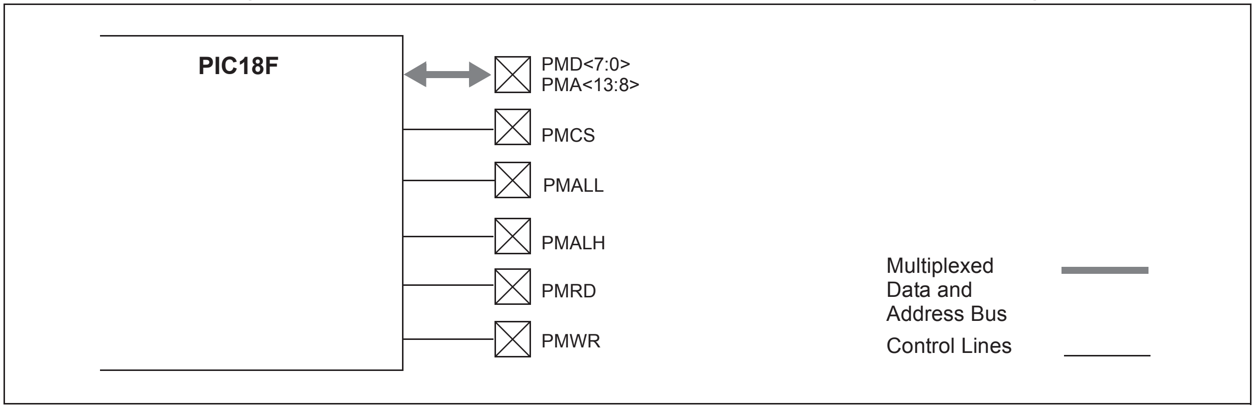

PIC18F
PMD<7:0>
PMA<7:0>
PMCS
PMALL
PMRD
PMWR
Address Bus
Multiplexed Data and Address Bus
Control Lines

FIGURE 11-11: FULLY MULTIPLEXED ADDRESSING MODE (SEPARATE READ AND WRITE STROBES WITH CHIP SELECT)


PIC18F
PMD<7:0>
PMA<13:8>
PMCS
PMALL
PMALH
PMRD
PMWR
Multiplexed Data and Address Bus
Control Lines

11.3.5 CHIP SELECT FEATURES

A chip select line, PMCS, is available for the Master modes of the PMP. The chip select line is multiplexed with the second Most Significant bit (MSb) of the address bus (PMADDRH<6>). When configured for chip select, the PMADDRH<7:6> bits are not included in any address auto-increment/decrement. The function of the chip select signal is configured using the chip select function bits (PMCONL<7:6>).

11.3.6 AUTO-INCREMENT/DECREMENT

While the module is operating in one of the Master modes, the INCMx bits (PMMODEH<4:3>) control the behavior of the address value. The address can be made to automatically increment or decrement after each read and write operation. The address increments once each operation is completed and the BUSY bit goes to '`0`'. If the chip select signals are disabled and configured as address bits, the bits will participate in the increment and decrement operations; otherwise, the CS1 bit values will be unaffected.

11.3.7 WAIT STATES

In Master mode, the user has control over the duration of the read, write and address cycles by configuring the module Wait states. Three portions of the cycle, the beginning, middle and end, are configured using the corresponding WAITBx, WAITMx and WAITEx bits in the PMMODEL register.

The WAITBx bits (PMMODEL<7:6>) set the number of Wait cycles for the data setup prior to the PMRD/PMWT strobe in Mode 10, or prior to the PMENB strobe in Mode 11. The WAITMx bits (PMMODEL<5:2>) set the number of Wait cycles for the PMRD/PMWT strobe in Mode 10, or for the PMENB strobe in Mode 11. When this Wait state setting is '`0`', then WAITB and WAITE have no effect. The WAITE bits (PMMODEL<1:0>) define the number of Wait cycles for the data hold time after the PMRD/PMWT strobe in Mode 10 or after the PMENB strobe in Mode 11.

11.3.8 READ OPERATION

To perform a read on the PMP, the user reads the PMDIN1L register. This causes the PMP to output the desired values on the chip select lines and the address bus. Then, the read line (PMRD) is strobed. The read data is placed into the PMDIN1L register.

If the 16-bit mode is enabled (MODE16 = `1`), the read of the low byte of the PMDIN1L register will initiate two bus reads. The first read data byte is placed into the PMDIN1L register and the second read data byte is placed into the PMDIN1H.

Note that the read data obtained from the PMDIN1L register is actually the read value from the previous read operation. Hence, the first user read will be a dummy read to initiate the first bus read and fill the read register. Also, the requested read value will not be ready until after the BUSY bit is observed low. Thus, in a back-to-back read operation, the data read from the register will be the same for both reads. The next read of the register will yield the new value.

11.3.9 WRITE OPERATION

To perform a write onto the parallel bus, the user writes to the PMDIN1L register. This causes the module to first output the desired values on the chip select lines and the address bus. The write data from the PMDIN1L register is placed onto the PMD<7:0> data bus. Then, the write line (PMWR) is strobed. If the 16-bit mode is enabled (MODE16 = `1`), the write to the PMDIN1L register will initiate two bus writes. The first write will consist of the data contained in PMDIN1L and the second write will contain the PMDIN1H.

11.3.10 PARALLEL MASTER PORT STATUS

11.3.10.1 The BUSY Bit

In addition to the PMP interrupt, a BUSY bit is provided to indicate the status of the module. This bit is used only in Master mode. While any read or write operation is in progress, the BUSY bit is set for all but the very last CPU cycle of the operation. In effect, if a single-cycle read or write operation is requested, the BUSY bit will never be active. This allows back-to-back transfers. While the bit is set, any request by the user to initiate a new operation will be ignored (i.e., writing or reading the lower byte of the PMDIN1L register will neither initiate a read nor a write).

11.3.10.2 Interrupts

When the PMP module interrupt is enabled for Master mode, the module will interrupt on every completed read or write cycle; otherwise, the BUSY bit is available to query the status of the module.

11.3.11 MASTER MODE TIMING

This section contains a number of timing examples that represent the common Master mode configuration options. These options vary from 8-bit to 16-bit data, fully demultiplexed to fully multiplexed address and Wait states.

FIGURE 11-12: READ AND WRITE TIMING, 8-BIT DATA, DEMULTIPLEXED ADDRESS

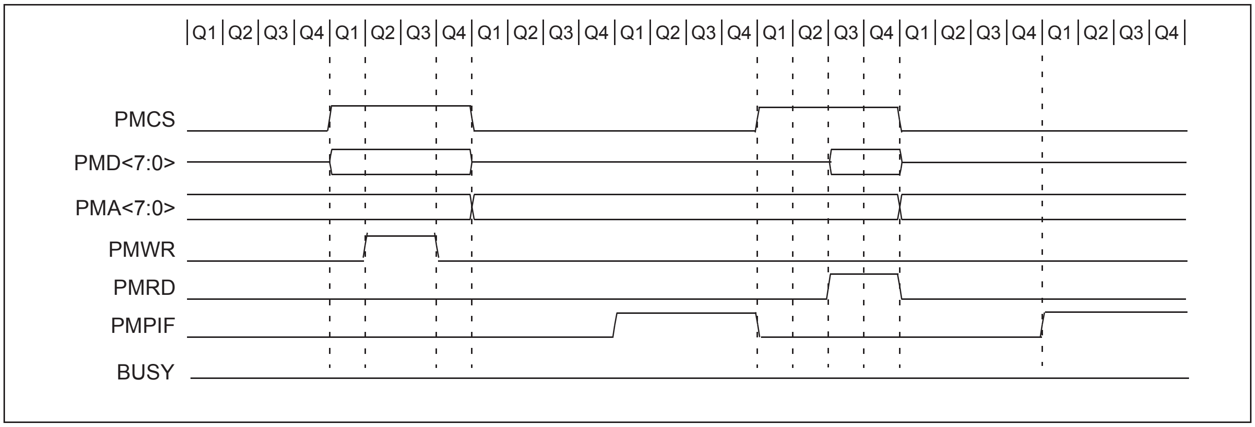

|Q1|Q2|Q3|Q4|Q1|Q2|Q3|Q4|Q1|Q2|Q3|Q4|Q1|Q2|Q3|Q4|Q1|Q2|Q3|Q4|Q1|Q2|Q3|Q4|Q1|Q2|Q3|Q4|
PMCS
PMD<7:0>
PMA<7:0>
PMWR
PMRD
PMPIF
BUSY

FIGURE 11-13: READ TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS

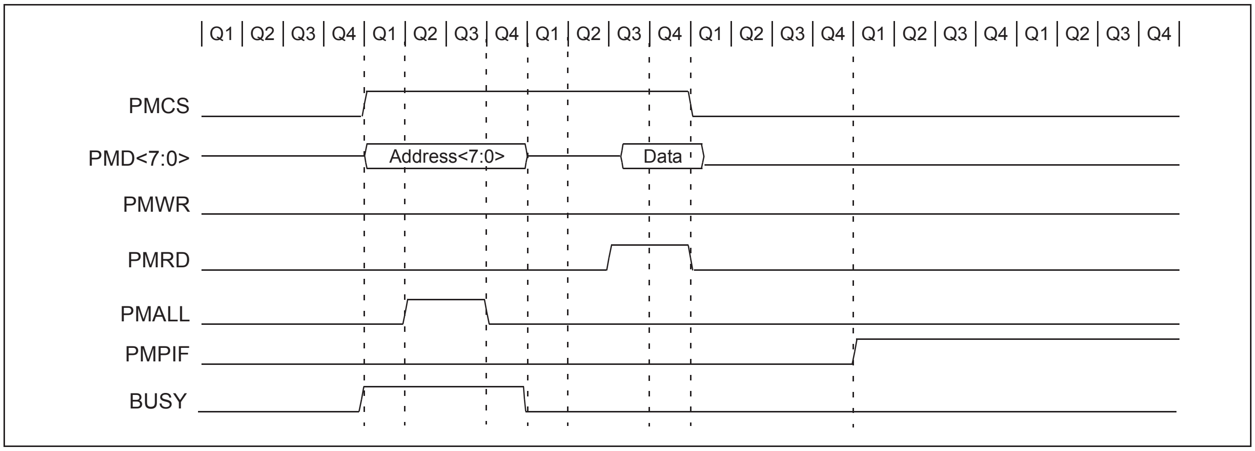

| Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 |
PMCS
PMD<7:0>
Address<7:0>
Data
PMWR
PMRD
PMALL
PMPIF
BUSY

FIGURE 11-14: READ TIMING, 8-BIT DATA, WAIT STATES ENABLED, PARTIALLY MULTIPLEXED ADDRESS

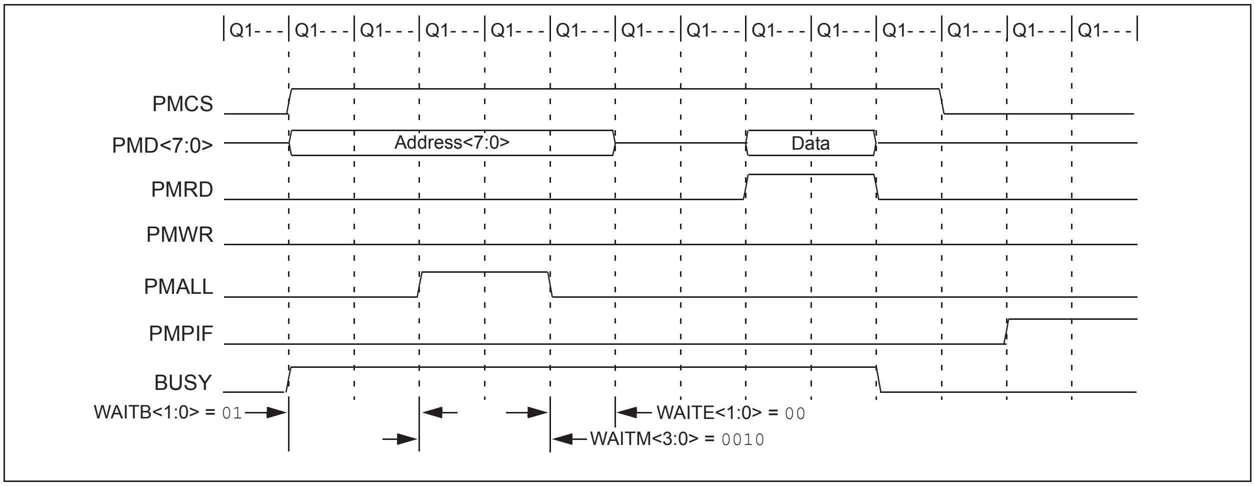

|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|Q1- - -|
PMCS
PMD<7:0>
Address<7:0>
Data
PMRD
PMWR
PMALL
PMPIF
BUSY
WAITB<1:0> = 01
WAITE<1:0> = 00
WAITM<3:0> = 0010

FIGURE 11-15: WRITE TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS

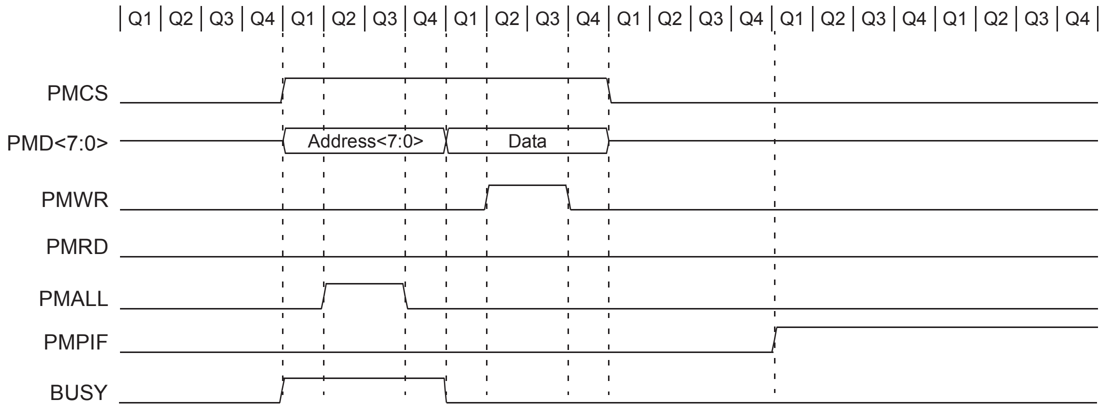

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
Address<7:0>
Data
PMWR
PMRD
PMALL
PMPIF
BUSY

FIGURE 11-16: WRITE TIMING, 8-BIT DATA, WAIT STATES ENABLED, PARTIALLY MULTIPLEXED ADDRESS

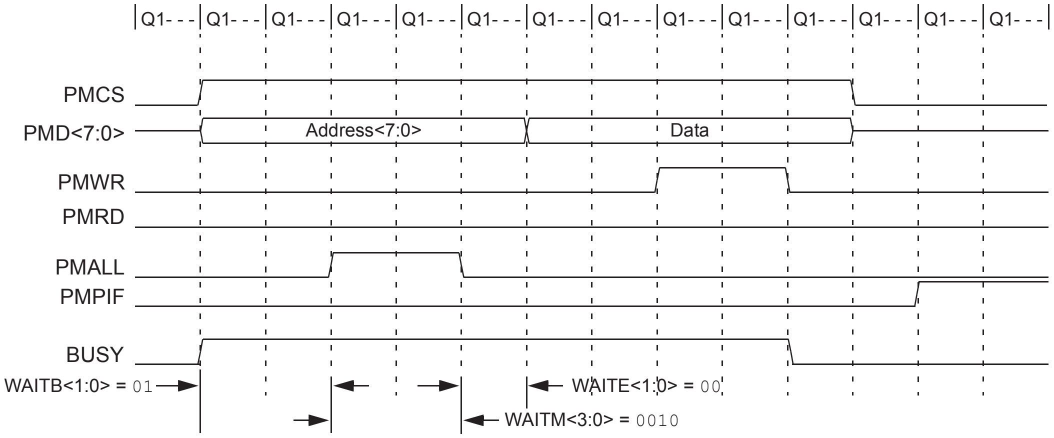

Q1- - - Q1- - - Q1- - - Q1- - - Q1- - - Q1- - - Q1- - - Q1- - - Q1- - - Q1- - - Q1- - - Q1- - - Q1- - - Q1- - -
PMCS
PMD<7:0>
Address<7:0>
Data
PMWR
PMRD
PMALL
PMPIF
BUSY
WAITB<1:0> = 01
WAITE<1:0> = 00
WAITM<3:0> = 0010

FIGURE 11-17: READ TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS, ENABLE STROBE

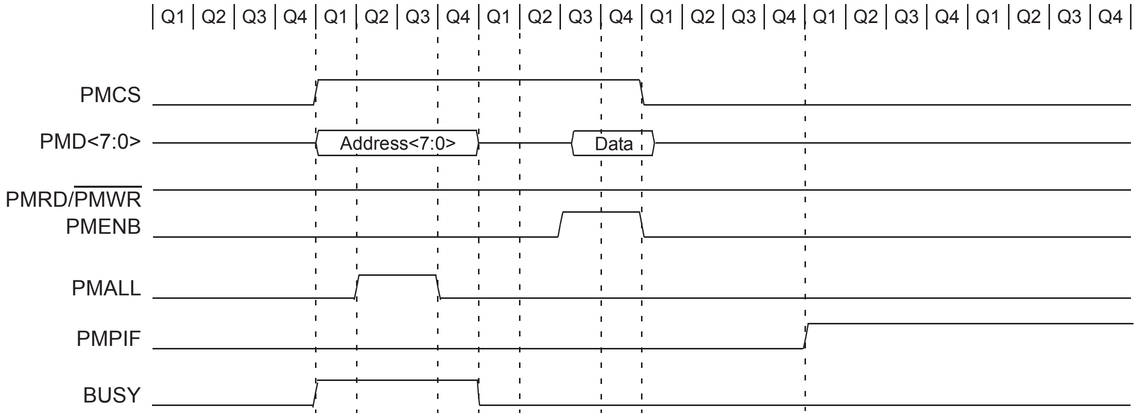

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
Address<7:0>
Data
PMRD/$\overline{\text{PMWR}}$
PMENB
PMALL
PMPIF
BUSY

FIGURE 11-18: WRITE TIMING, 8-BIT DATA, PARTIALLY MULTIPLEXED ADDRESS, ENABLE STROBE

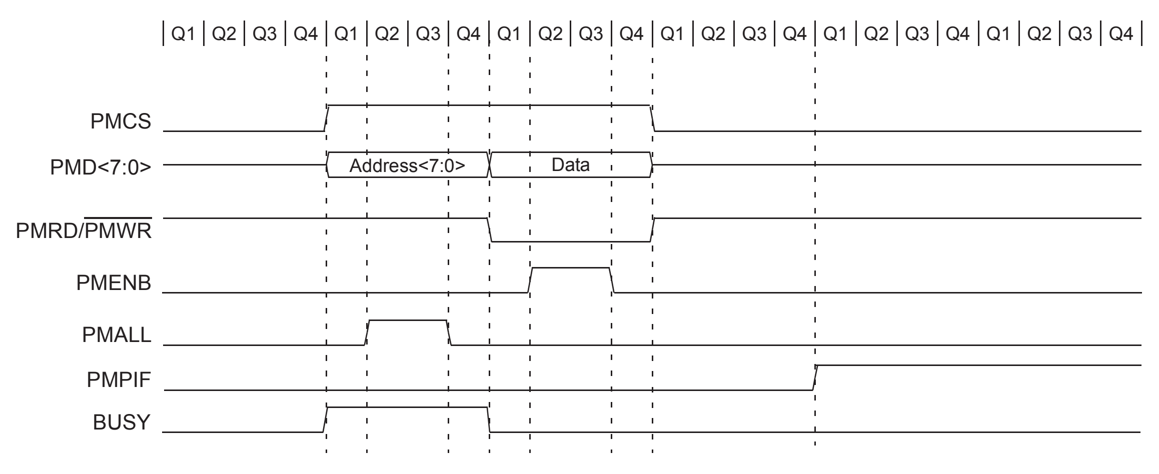

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
Address<7:0>
Data
PMRD/$\overline{\text{PMWR}}$
PMENB
PMALL
PMPIF
BUSY

FIGURE 11-19: READ TIMING, 8-BIT DATA, FULLY MULTIPLEXED 16-BIT ADDRESS

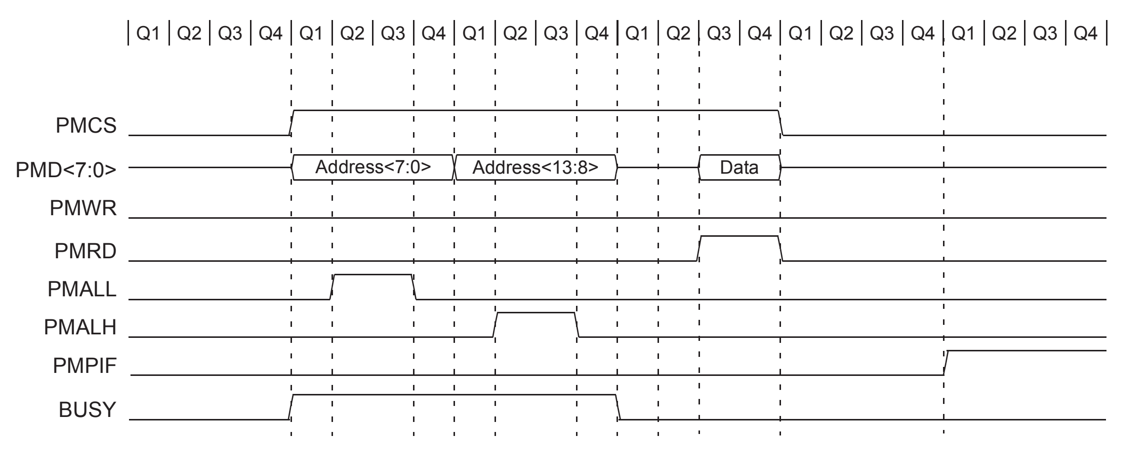

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
Address<7:0>
Address<13:8>
Data
PMWR
PMRD
PMALL
PMALH
PMPIF
BUSY

FIGURE 11-20: WRITE TIMING, 8-BIT DATA, FULLY MULTIPLEXED 16-BIT ADDRESS

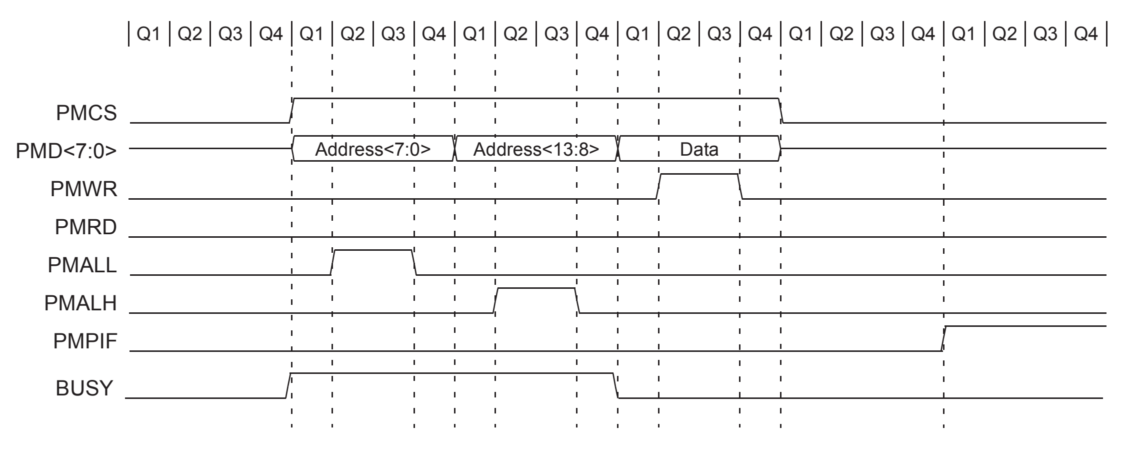

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
Address<7:0>
Address<13:8>
Data
PMWR
PMRD
PMALL
PMALH
PMPIF
BUSY

FIGURE 11-21: READ TIMING, 16-BIT DATA, DEMULTIPLEXED ADDRESS

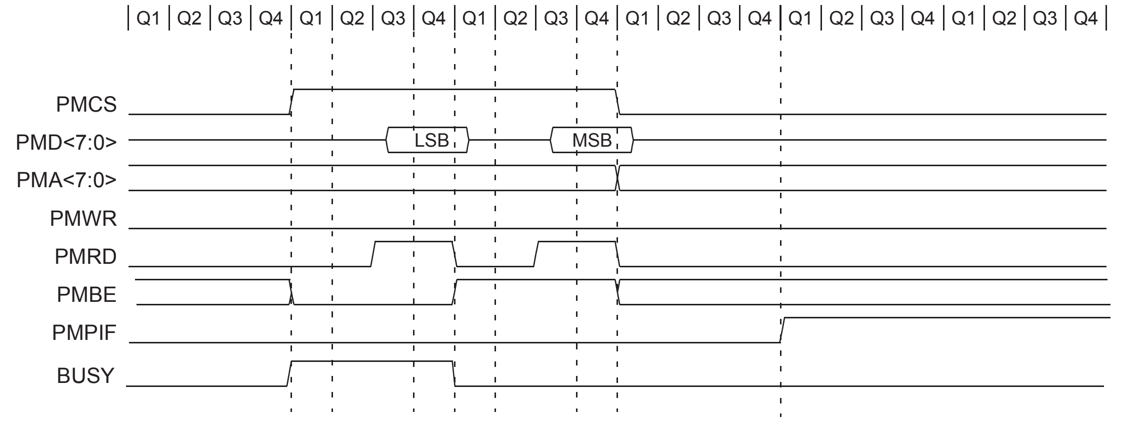

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
LSB
MSB
PMA<7:0>
PMWR
PMRD
PMBE
PMPIF
BUSY

FIGURE 11-22: WRITE TIMING, 16-BIT DATA, DEMULTIPLEXED ADDRESS

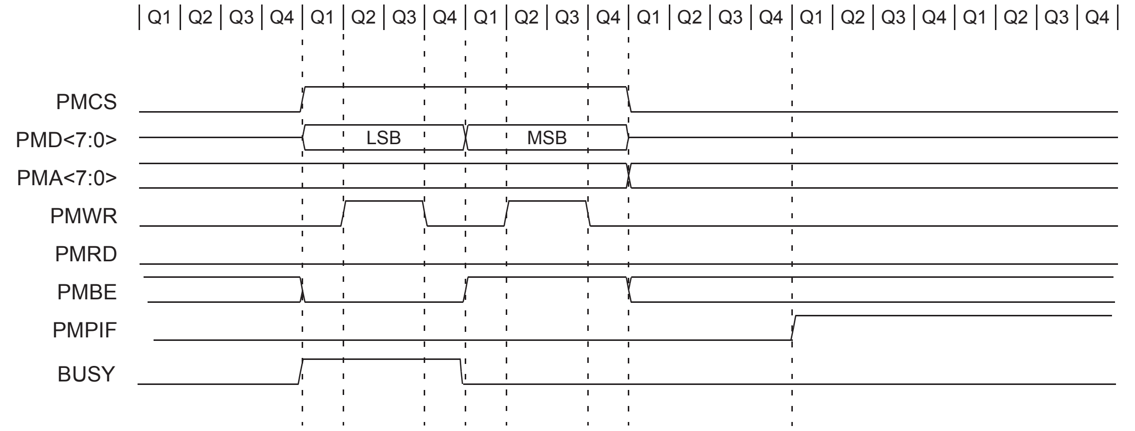

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
LSB
MSB
PMA<7:0>
PMWR
PMRD
PMBE
PMPIF
BUSY

FIGURE 11-23: READ TIMING, 16-BIT MULTIPLEXED DATA, PARTIALLY MULTIPLEXED ADDRESS

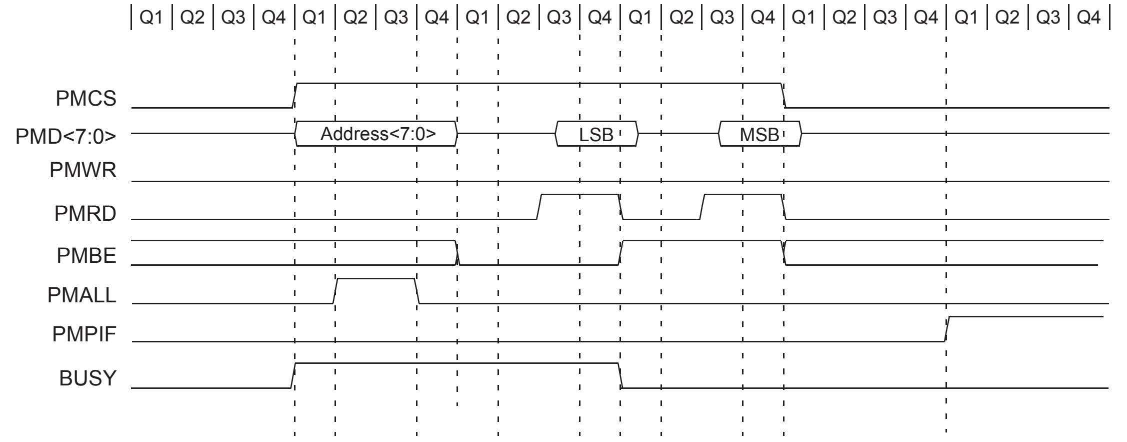

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
Address<7:0>
LSB
MSB
PMWR
PMRD
PMBE
PMALL
PMPIF
BUSY

FIGURE 11-24: WRITE TIMING, 16-BIT MULTIPLEXED DATA, PARTIALLY MULTIPLEXED ADDRESS

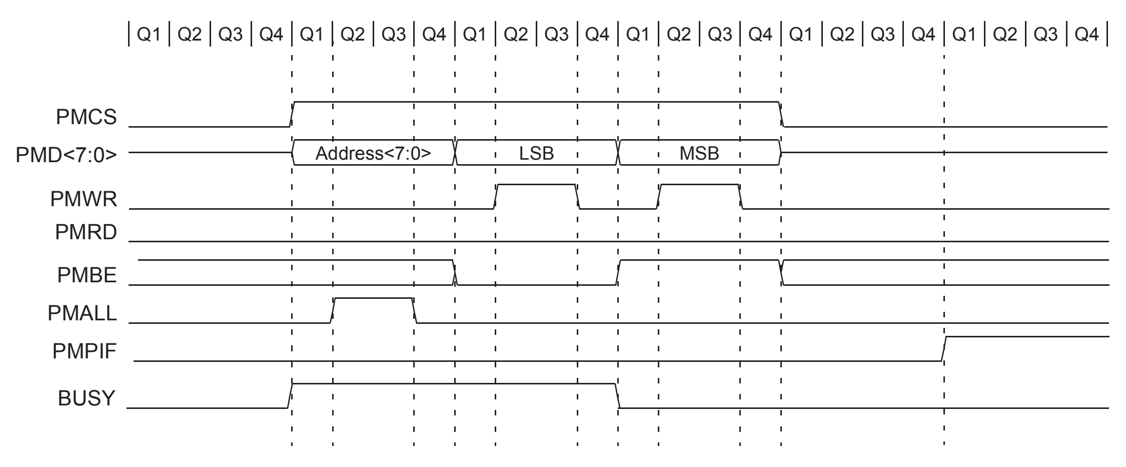

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
Address<7:0>
LSB
MSB
PMWR
PMRD
PMBE
PMALL
PMPIF
BUSY

FIGURE 11-25: READ TIMING, 16-BIT MULTIPLEXED DATA, FULLY MULTIPLEXED 16-BIT ADDRESS

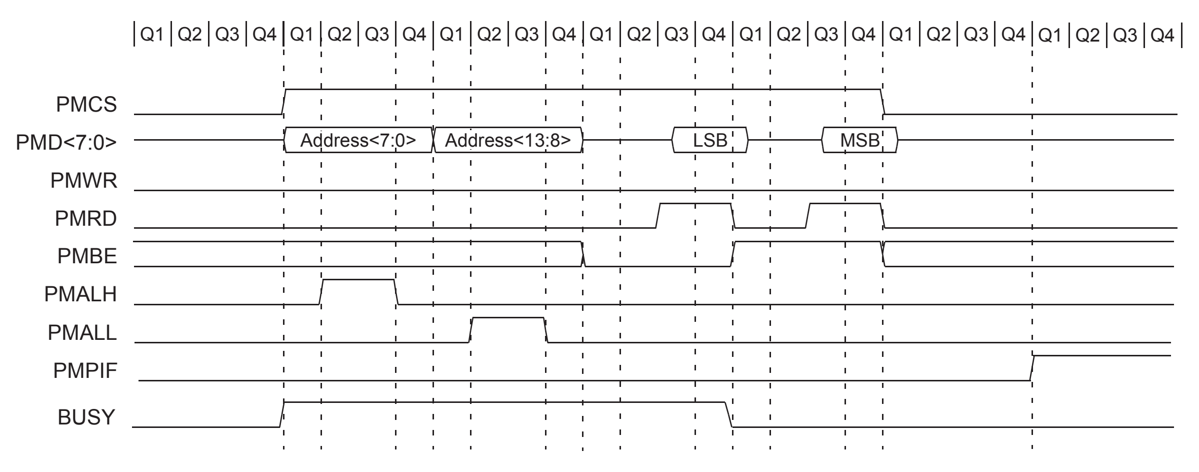
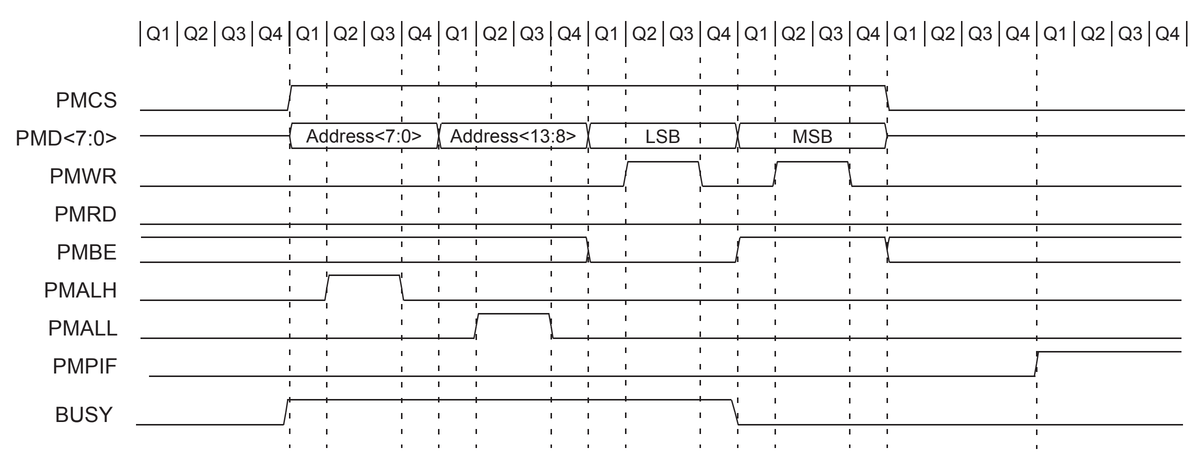

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
Address<7:0>
Address<13:8>
LSB
MSB
PMWR
PMRD
PMBE
PMALH
PMALL
PMPIF
BUSY

FIGURE 11-26: WRITE TIMING, 16-BIT MULTIPLEXED DATA, FULLY MULTIPLEXED 16-BIT ADDRESS


Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PMCS
PMD<7:0>
Address<7:0>
Address<13:8>
LSB
MSB
PMWR
PMRD
PMBE
PMALH
PMALL
PMPIF
BUSY

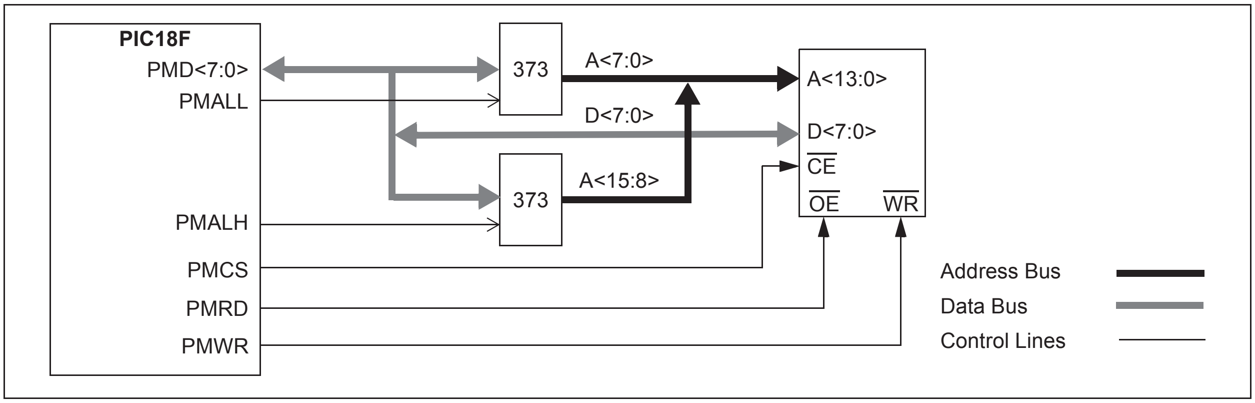
11.4 Application Examples

This section introduces some potential applications for the PMP module.

11.4.1 MULTIPLEXED MEMORY OR PERIPHERAL

Figure 11-27 demonstrates the hookup of a memory or another addressable peripheral in Full Multiplex mode. Consequently, this mode achieves the best pin saving from the microcontroller perspective. However, for this configuration, there needs to be some external latches to maintain the address.

FIGURE 11-27: MULTIPLEXED ADDRESSING APPLICATION EXAMPLE


PIC18F
PMD<7:0>
PMALL
PMALH
PMCS
PMRD
PMWR
373
373
A<7:0>
D<7:0>
A<15:8>
A<13:0>
D<7:0>
CE
OE
WR
Address Bus
Data Bus
Control Lines

11.4.2 PARTIALLY MULTIPLEXED MEMORY OR PERIPHERAL

Partial multiplexing implies using more pins; however, for a few extra pins, some extra performance can be achieved. Figure 11-28 provides an example of a memory or peripheral that is partially multiplexed with an external latch. If the peripheral has internal latches, as displayed in Figure 11-29, then no extra circuitry is required except for the peripheral itself.

FIGURE 11-28: EXAMPLE OF A PARTIALLY MULTIPLEXED ADDRESSING APPLICATION

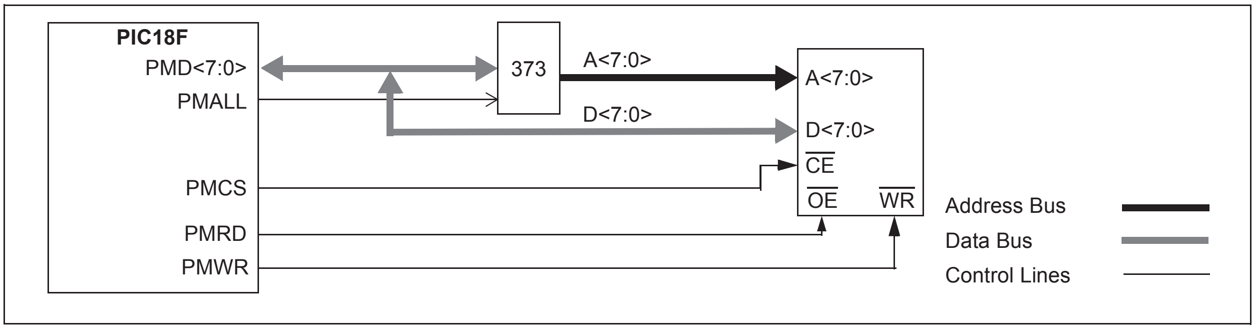
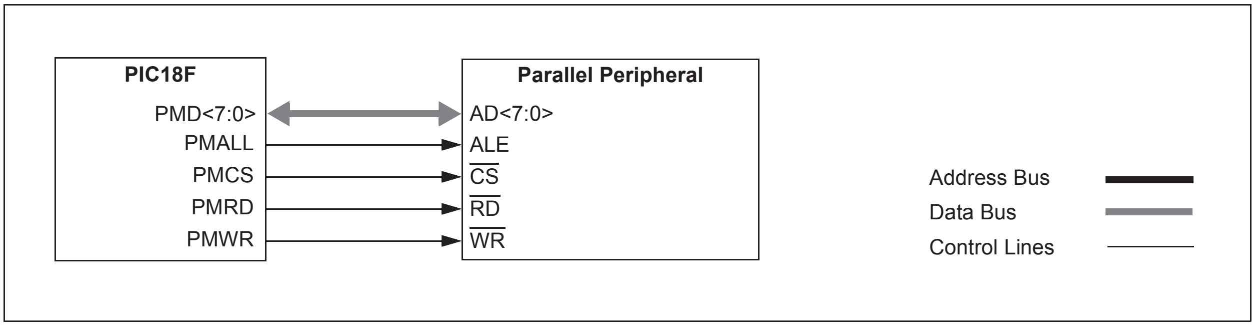

PIC18F
PMD<7:0>
PMALL
PMCS
PMRD
PMWR
373
A<7:0>
D<7:0>
A<7:0>
D<7:0>
CE
OE
WR
Address Bus
Data Bus
Control Lines

FIGURE 11-29: EXAMPLE OF AN 8-BIT MULTIPLEXED ADDRESS AND DATA APPLICATION


PIC18F
PMD<7:0>
PMALL
PMCS
PMRD
PMWR
Parallel Peripheral
AD<7:0>
ALE
CS
RD
WR
Address Bus
Data Bus
Control Lines

11.4.3 PARALLEL EEPROM EXAMPLE

Figure 11-30 provides an example connecting parallel EEPROM to the PMP. Figure 11-31 demonstrates a slight variation to this, configuring the connection for 16-bit data from a single EEPROM.

FIGURE 11-30: PARALLEL EEPROM EXAMPLE (UP TO 15-BIT ADDRESS, 8-BIT DATA)

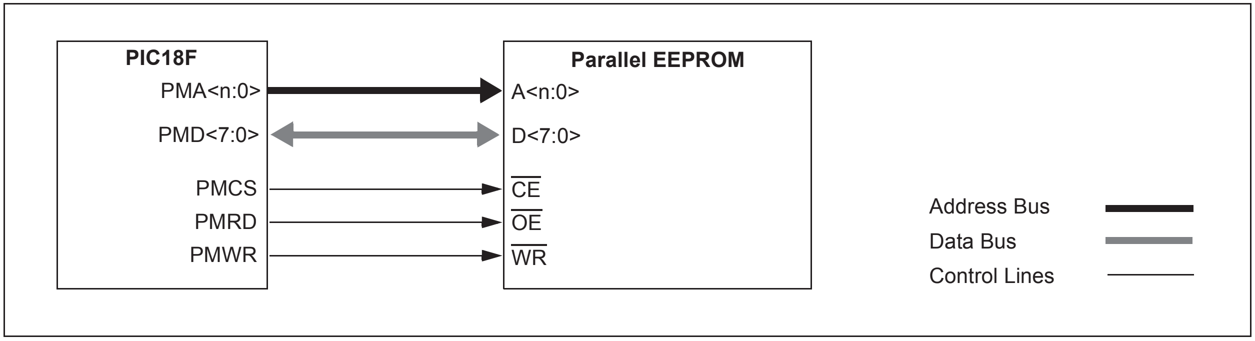
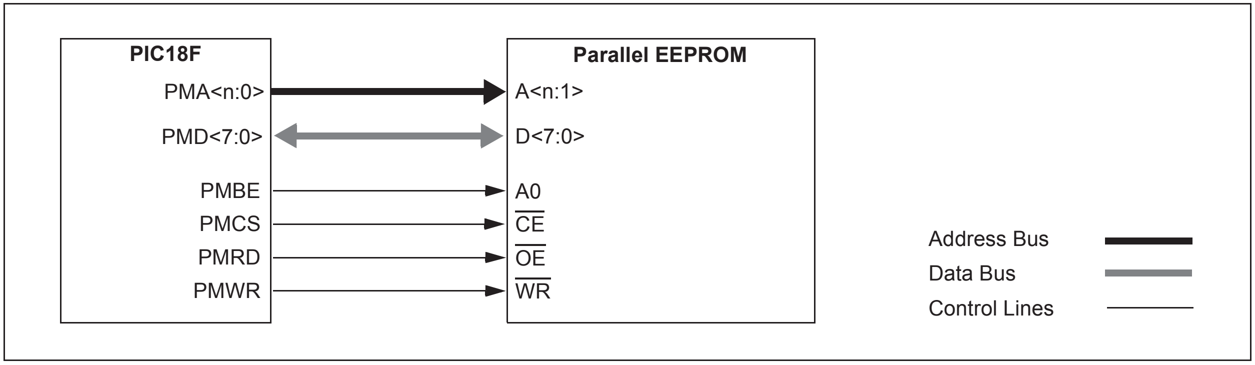

PIC18F
PMA<n:0>
PMD<7:0>
PMCS
PMRD
PMWR
Parallel EEPROM
A<n:0>
D<7:0>
$\overline{CE}$
$\overline{OE}$
$\overline{WR}$
Address Bus
Data Bus
Control Lines

FIGURE 11-31: PARALLEL EEPROM EXAMPLE (UP TO 15-BIT ADDRESS, 16-BIT DATA)


PIC18F
PMA<n:0>
PMD<7:0>
PMBE
PMCS
PMRD
PMWR
Parallel EEPROM
A<n:1>
D<7:0>
A0
$\overline{CE}$
$\overline{OE}$
$\overline{WR}$
Address Bus
Data Bus
Control Lines

11.4.4 LCD CONTROLLER EXAMPLE

The PMP module can be configured to connect to a typical LCD controller interface, as displayed in Figure 11-32. In this case the PMP module is configured for active-high control signals since common LCD displays require active-high control.

FIGURE 11-32: LCD CONTROL EXAMPLE (BYTE MODE OPERATION)

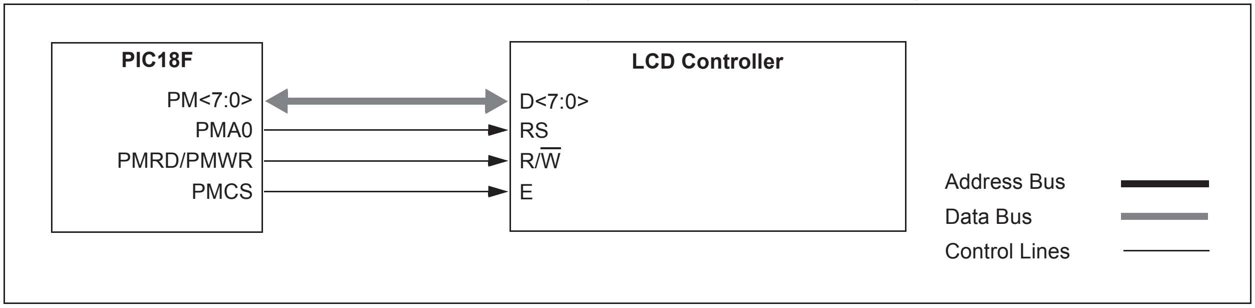

PIC18F
PM<7:0>
PMA0
PMRD/PMWR
PMCS
LCD Controller
D<7:0>
RS
R/$\overline{W}$
E
Address Bus
Data Bus
Control Lines

TABLE 11-2: REGISTERS ASSOCIATED WITH PMP 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	PMPIF(2)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(2)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(2)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PMCONH(2)	PMPEN	—	—	ADRMUX1	ADRMUX0	PTBEEN	PTWREN	PTRDEN
PMCONL(2)	CSF1	CSF0	ALP	—	CS1P	BEP	WRSP	RDSP
PMADDRH(1,2)/	—	CS1	Parallel Master Port Address High Byte					
PMDOUT1H(1,2)	Parallel Port Out Data High Byte (Buffer 1)							
PMADDRL(1,2)/	Parallel Master Port Address Low Byte							
PMDOUT1L(1,2)	Parallel Port Out Data Low Byte (Buffer 0)							
PMDOUT2H(2)	Parallel Port Out Data High Byte (Buffer 3)							
PMDOUT2L(2)	Parallel Port Out Data Low Byte (Buffer 2)							
PMDIN1H(2)	Parallel Port In Data High Byte (Buffer 1)							
PMDIN1L(2)	Parallel Port In Data Low Byte (Buffer 0)							
PMDIN2H(2)	Parallel Port In Data High Byte (Buffer 3)							
PMDIN2L(2)	Parallel Port In Data Low Byte (Buffer 2)							
PMMODEH(2)	BUSY	IRQM1	IRQM0	INCM1	INCM0	MODE16	MODE1	MODE0
PMMODEL(2)	WAITB1	WAITB0	WAITM3	WAITM2	WAITM1	WAITM0	WAITE1	WAITE0
PMEH(2)	PTEN15	PTEN14	PTEN13	PTEN12	PTEN11	PTEN10	PTEN9	PTEN8
PMEL(2)	PTEN7	PTEN6	PTEN5	PTEN4	PTEN3	PTEN2	PTEN1	PTEN0
PMSTATH(2)	IBF	IBOV	—	—	IB3F	IB2F	IB1F	IB0F
PMSTATL(2)	OBE	OBUF	—	—	OB3E	OB2E	OB1E	OB0E
PADCFG1	—	—	—	—	—	RTSECSEL1	RTSECSEL0	PMPTTL

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

Note 1: The PMADDRH/PMDOUT1H and PMADDRL/PMDOUT1L register pairs share the physical registers and addresses, but have different functions determined by the module's operating mode.

2: These bits and/or registers are only available in 44-pin devices.

12.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 12-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

Figure 12-1 provides a simplified block diagram of the Timer0 module in 8-bit mode. Figure 12-2 provides a simplified block diagram of the Timer0 module in 16-bit mode.

REGISTER 12-1: T0CON: TIMER0 CONTROL REGISTER (ACCESS FD5h)

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 the 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

12.1 Timer0 Operation

Timer0 can operate as either a timer or a counter. The mode is selected with the T0CS bit (T0CON<5>). In Timer mode (T0CS = `0`), the module increments on every clock by default unless a different prescaler value is selected (see **Section 12.3 "Prescaler"**). If the TMR0 register is written to, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

The Counter mode is selected by setting the T0CS bit (= `1`). In this mode, Timer0 increments either on every rising edge or falling edge of the pin, T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit, T0SE (T0CON<4>); clearing this bit selects the rising edge. Restrictions on the external clock input are discussed below.

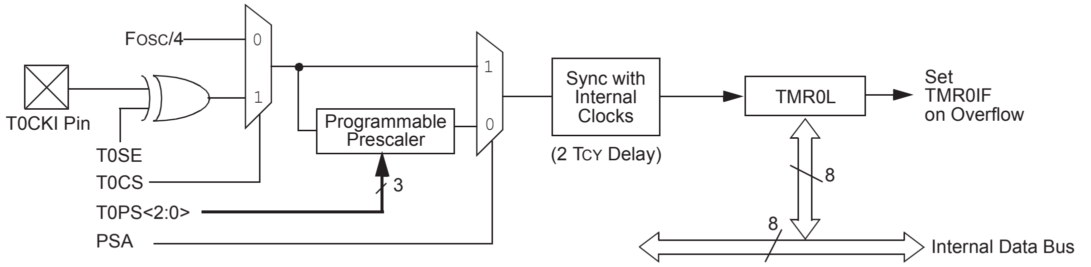
An external clock source can be used to drive Timer0; however, it must meet certain requirements to ensure that the external clock can be synchronized with the internal phase clock (TOSC). There is a delay between synchronization and the onset of incrementing the timer/counter.

12.2 Timer0 Reads and Writes in 16-Bit Mode

TMR0H is not the actual high byte of Timer0 in 16-bit mode. It is actually a buffered version of the real high byte of Timer0, which is not directly readable nor writable (refer to Figure 12-2). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte were valid, due to a rollover between successive reads of the high and low byte.

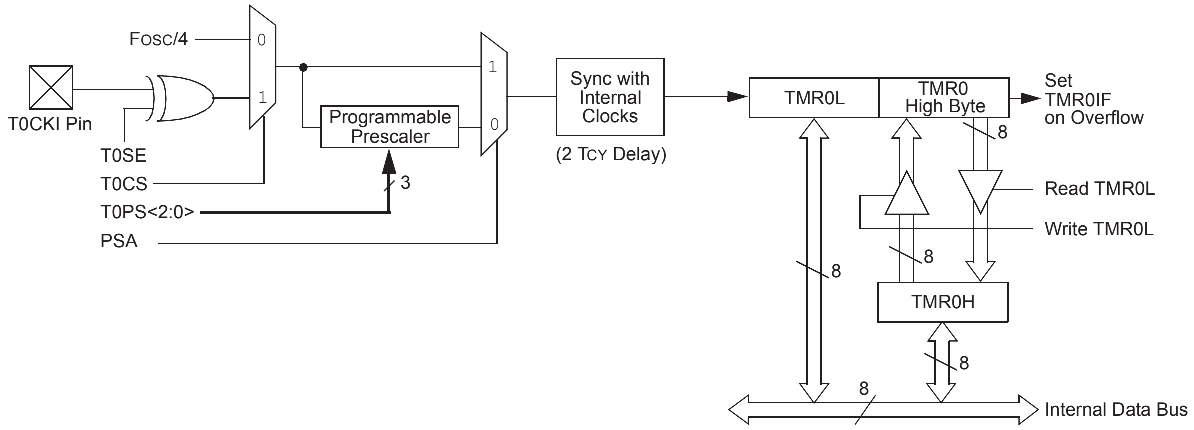
Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

FIGURE 12-1: TIMER0 BLOCK DIAGRAM (8-BIT MODE)


FOSC/4
T0CKI Pin
T0SE
T0CS
T0PS<2:0>
PSA
0
1
1
0
Programmable Prescaler
3
Sync with Internal Clocks
(2 TCY Delay)
TMR0L
Set TMR0IF on Overflow
8
8
Internal Data Bus

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

FIGURE 12-2: TIMER0 BLOCK DIAGRAM (16-BIT MODE)


FOSC/4
T0CKI Pin
T0SE
T0CS
T0PS<2:0>
PSA
0
1
1
0
Programmable Prescaler
3
Sync with Internal Clocks
(2 TCY Delay)
TMR0L
TMR0 High Byte
Set TMR0IF on Overflow
8
Read TMR0L
Write TMR0L
8
8
TMR0H
8
8
Internal Data Bus

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

12.3 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not directly readable or writable. Its value is set by the PSA and T0PS<2:0> bits (T0CON<3:0>), which determine the prescaler assignment and prescale ratio.

Clearing the PSA bit assigns the prescaler to the Timer0 module. When it is assigned, prescale values from 1:2 through 1:256, in power-of-2 increments, are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF TMR0`, `MOVWF TMR0`, `BSF TMR0`, etc.) clear the prescaler count.

> **Note:** Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count but will not change the prescaler assignment.

12.3.1 SWITCHING PRESCALER ASSIGNMENT

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

12.4 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from FFh to 00h in 8-bit mode, or from FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF flag bit. The interrupt can be masked by clearing the 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 12-1: 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: Shaded cells are not used by Timer0.

NOTES:

13.0 TIMER1 MODULE

The Timer1 timer/counter module incorporates these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMR1H and TMR1L)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Reset on ECCP Special Event Trigger
- Device clock status flag (SOSCRUN)
- Timer with gated control

Figure 13-1 displays a simplified block diagram of the Timer1 module.

The module incorporates its own low-power oscillator to provide an additional clocking option. The Timer1 oscillator can also be used as a low-power clock source for the microcontroller in power-managed operation.

Timer1 is controlled through the T1CON Control register (Register 13-1). It also contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

The FOSC clock source (TMR1CS<1:0> = `01`) 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 13-1: T1CON: TIMER1 CONTROL REGISTER (ACCESS FCDh)

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	T1OSCEN	$\overline{\text{T1SYNC}}$	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 the T1OSC or T1CKI 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 **T1OSCEN:** Timer1 Oscillator Source Select bit
When TMR1CS<1:0> = `10`:
`1` = Power up the Timer1 crystal driver and supply the Timer1 clock from the crystal output
`0` = Timer1 crystal driver is off, Timer1 clock is from the T1CKI input pin(2)
When TMR1CS<1:0> = `0x`:
`1` = Power up the Timer1 crystal driver
`0` = Timer1 crystal driver is off(2)

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> = `0x`.

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

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.

2: The Timer1 oscillator crystal driver is powered whenever T1OSCEN (T1CON<3>) or T3OSCEN (T3CON<3>) = `1`. The circuit is enabled by the logical OR of these two bits. When disabled, the inverter and feedback resistor are disabled to eliminate power drain. The TMR1ON and TMR3ON bits do not have to be enabled to power up the crystal driver.

13.1 Timer1 Gate Control Register

The Timer1 Gate Control register (T1GCON), displayed in Register 13-2, is used to control the Timer1 gate.

REGISTER 13-2: T1GCON: TIMER1 GATE CONTROL REGISTER (ACCESS F9Ah)(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 the 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 the 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

`00` = Timer1 gate pin
`01` = TMR2 to match PR2 output
`10` = Comparator 1 output
`11` = Comparator 2 output

Note 1: Programming the T1GCON prior to T1CON is recommended.

13.2 Timer1 Operation

The Timer1 module is an 8-bit or 16-bit incrementing counter, which 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 and increments on every selected edge of the external source.

Timer1 is enabled by configuring the TMR1ON and TMR1GE bits in the T1CON and T1GCON registers, respectively.

When Timer1 is enabled, the RC1/CCP8/T1OSI/RP12 and RC0/T1OSO/T1CKI/RP11 pins become inputs. This means the values of TRISC<1:0> are ignored and the pins are read as ‘`0`’.

13.3 Clock Source Selection

The TMR1CS<1:0> and T1OSCEN bits of the T1CON register are used to select the clock source for Timer1. Register 13-1 displays the clock source selections.

13.3.1 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.

13.3.2 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, or the capacitive sensing oscillator signal. Either of these external clock sources can be synchronized to the microcontroller system clock or they can run asynchronously.

When used as a timer with a clock oscillator, an external 32.768 kHz crystal can be used in conjunction with the dedicated internal oscillator circuit.

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:

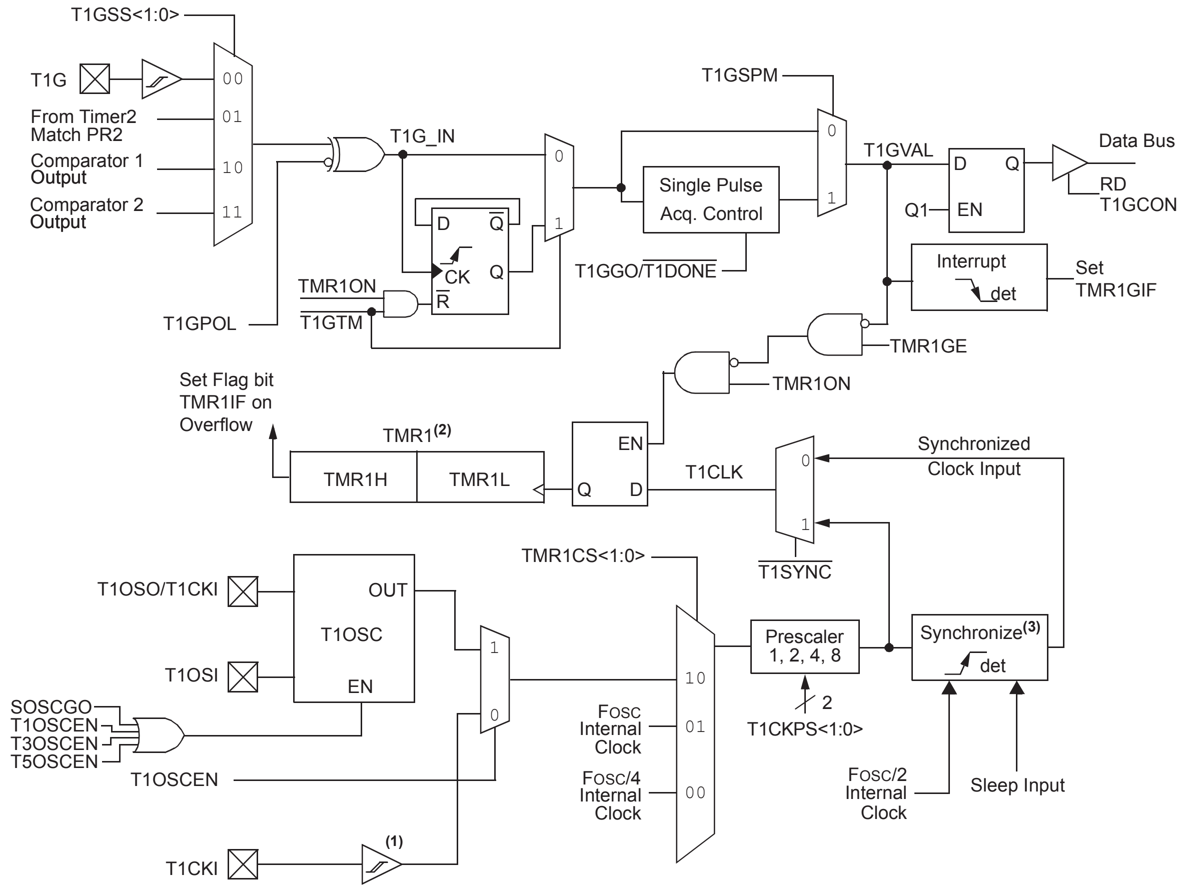
- Timer1 enabled after POR Reset
- Write to TMR1H or TMR1L
- Timer1 is disabled
- Timer1 is disabled (TMR1ON = `0`)

when T1CKI is high, then Timer1 is enabled (TMR1ON = `1`) when T1CKI is low.

TABLE 13-1: TIMER1 CLOCK SOURCE SELECTION

TMR1CS1	TMR1CS0	T1OSCEN	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 T1OSI/T1OSO Pin

FIGURE 13-1: TIMER1 BLOCK DIAGRAM


T1GSS<1:0>
T1G
00
From Timer2 Match PR2
01
Comparator 1 Output
10
Comparator 2 Output
11
T1G_IN
T1GPOL
TMR1ON
T1GTM
D
Q̅
CK
Q
R̅
0
1
T1GSPM
Single Pulse Acq. Control
T1GGO/T1DONE
0
1
T1GVAL
D
Q
Q1
EN
Data Bus
RD T1GCON
Interrupt det
Set TMR1GIF
TMR1GE
TMR1ON
Set Flag bit TMR1IF on Overflow
TMR1(2)
TMR1H
TMR1L
EN
Q
D
T1CLK
0
1
Synchronized Clock Input
T1SYNC
TMR1CS<1:0>
T1OSO/T1CKI
OUT
T1OSC
T1OSI
EN
1
0
SOSCGO
T1OSCEN
T3OSCEN
T5OSCEN
T1OSCEN
T1CKI
(1)
10
FOSC Internal Clock
01
FOSC/4 Internal Clock
00
Prescaler 1, 2, 4, 8
2
T1CKPS<1:0>
Synchronize(3) det
FOSC/2 Internal Clock
Sleep Input

Note 1: ST buffer is high-speed type when using T1CKI.

2: Timer1 register increments on the rising edge.

3: Synchronization does not operate while in Sleep.

13.4 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 to both the high and low bytes of Timer1 at once.

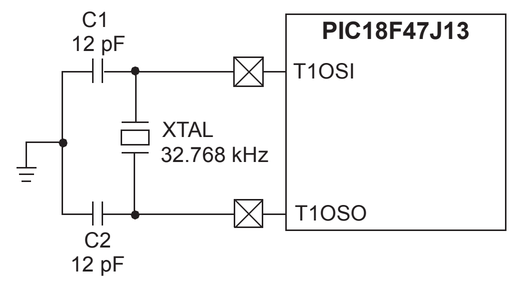
The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

13.5 Timer1 Oscillator

An on-chip crystal oscillator circuit is incorporated between pins, T1OSI (input) and T1OSO (amplifier output). It is enabled by setting the Timer1 Oscillator Enable bit, T1OSCEN (T1CON<3>). The oscillator is a low-power circuit rated for 32 kHz crystals. It will continue to run during all power-managed modes. The circuit for a typical LP oscillator is depicted in Figure 13-2. Table 13-2 provides the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

FIGURE 13-2: EXTERNAL COMPONENTS FOR THE TIMER1 LP OSCILLATOR



Note: See the Notes with Table 13-2 for additional information about capacitor selection.

TABLE 13-2: 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 C_L = 10 pF rated crystal when SOSCSEL = `0b11`.

5: Incorrect capacitance value may result in a frequency not meeting the crystal manufacturer's tolerance specification.

The Timer1 crystal oscillator drive level is determined based on the SOSCSEL (CONFIG2L<3:4>) Configuration bit. The Higher Drive Level mode, SOSCSEL = `0b11`, is intended to drive a wide variety of 32.768 kHz crystals with a variety of load capacitance (C_L) 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 T1OSI and T1OSO pins. This mode is only designed to work with discrete capacitances of approximately 3 pF-10 pF on each pin.

Crystal manufacturers usually specify a C_L (load capacitance) 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 13-2. See the crystal manufacturer's applications information for more details on how to select the optimum C1 and C2 for a given crystal. The optimum value depends in part on the amount of parasitic capacitance in the circuit, which is often unknown. Therefore, after values have been selected, it is highly recommended that thorough testing and validation of the oscillator be performed.

13.5.1 USING TIMER1 AS A CLOCK SOURCE

The Timer1 oscillator is also available as a clock source in power-managed modes. By setting the clock select bits, SCS<1:0> (OSCCON<1:0>), to '`01`', the device switches to SEC_RUN mode. Both the CPU and peripherals are clocked from the Timer1 oscillator. If the IDLEN bit (OSCCON<7>) is cleared and a `SLEEP` instruction is executed, the device enters SEC_IDLE mode. Additional details are available in **Section 4.0 "Low-Power Modes"**.

Whenever the Timer1 oscillator is providing the clock source, the Timer1 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. If the Clock Monitor is enabled and the Timer1 oscillator fails while providing the clock, polling the SOSCRUN bit will indicate whether the clock is being provided by the Timer1 oscillator or another source.

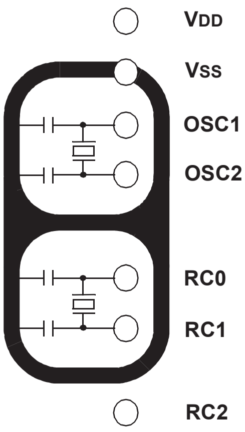
13.5.2 TIMER1 OSCILLATOR LAYOUT CONSIDERATIONS

The Timer1 oscillator circuit draws very little power during operation. Due to the low-power nature of the oscillator, it may also be sensitive to rapidly changing signals in close proximity. This is especially true when the oscillator is configured for extremely Low-Power mode (SOSCSEL = `0b01`).

The oscillator circuit, displayed in Figure 13-2, 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 (such as the ECCP1 pin in Output Compare or PWM mode, or the primary oscillator using the OSC2 pin), a grounded guard ring around the oscillator circuit, as displayed in Figure 13-3, may be helpful when used on a single-sided PCB or in addition to a ground plane.

FIGURE 13-3: OSCILLATOR CIRCUIT WITH GROUNDED GUARD RING



Note: Not drawn to scale.

In the Low Drive Level mode, SOSCSEL = `0b01`, it is critical that the 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 relatively good PCB layout. If possible, it is recommended to 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 = `0b11`) with many PCB layouts. Even in the High 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 also important to ensure that the circuit board is clean. Even a very small amount of conductive soldering flux residue can cause PCB leakage currents which can overwhelm the oscillator circuit.

13.6 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>).

13.7 Resetting Timer1 Using the ECCP Special Event Trigger

If ECCP1 or ECCP2 is configured to use Timer1 and to generate a Special Event Trigger in Compare mode (CCPxM<3:0> = `1011`), this signal will reset Timer3. The trigger from ECCP2 will also start an A/D conversion if the A/D module is enabled (see **Section 19.3.4 “Special Event Trigger”** for more information).

The module must be configured as either a timer or a synchronous counter to take advantage of this feature. When used this way, the CCPRxH:CCPRxL register pair effectively becomes a period register for Timer1.

If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a Special Event Trigger, the write operation will take precedence.

Note: The Special Event Trigger from the ECCPx module will not set the TMR1IF interrupt flag bit (PIR1<0>).

13.8 Timer1 Gate

The 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.

The Timer1 gate can also be driven by multiple selectable sources.

13.8.1 TIMER1 GATE COUNT ENABLE

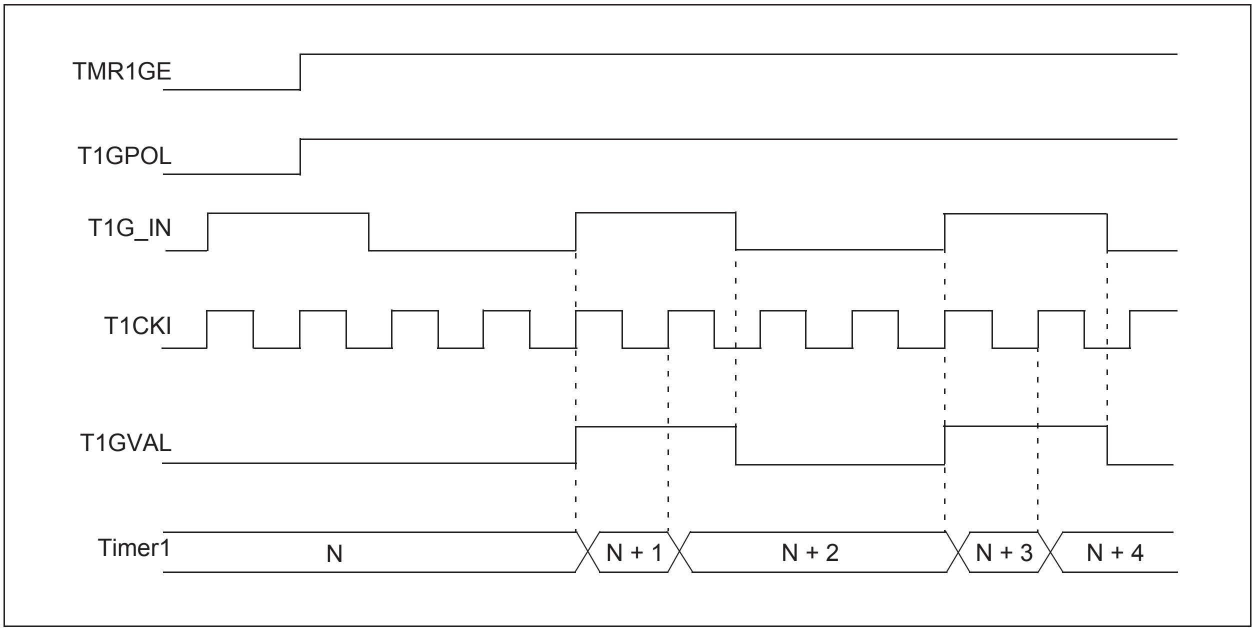
The Timer1 Gate Enable mode is enabled by setting the TMR1GE bit of the T1GCON register. The polarity of the Timer1 Gate Enable mode is configured using the T1GPOL bit of the T1GCON register.

When Timer1 Gate Enable mode is enabled, Timer1 will increment on the rising edge of the Timer1 clock source. When Timer1 Gate Enable mode is disabled, no incrementing will occur and Timer1 will hold the current count. See Figure 13-4 for timing details.

TABLE 13-3: TIMER1 GATE ENABLE SELECTIONS

T1CLK	T1GPOL	T1G	Timer1 Operation
↑	`0`	`0`	Counts
↑	`0`	`1`	Holds Count
↑	`1`	`0`	Holds Count
↑	`1`	`1`	Counts

FIGURE 13-4: TIMER1 GATE COUNT ENABLE MODE



13.8.2 TIMER1 GATE SOURCE SELECTION

The Timer1 gate source can be selected from one of four different sources. Source selection is controlled by the T1GSSx bits of the T1GCON register. The polarity for each available source is also selectable. Polarity selection is controlled by the T1GPOL bit of the T1GCON register.

TABLE 13-4: TIMER1 GATE SOURCES

T1GSS<1:0>	Timer1 Gate Source
`00`	Timer1 Gate Pin
`01`	TMR2 matches PR2
`10`	Comparator 1 output
`11`	Comparator 2 output

13.8.2.1 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.

13.8.2.2 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 remains high for one instruction cycle and returns to low until the next match.

When T1GPOL = `1`, Timer1 increments for a single instruction cycle following TMR2 matching PR2. With T1GPOL = `0`, Timer1 increments except during the cycle following the match.

13.8.3 TIMER1 GATE TOGGLE MODE

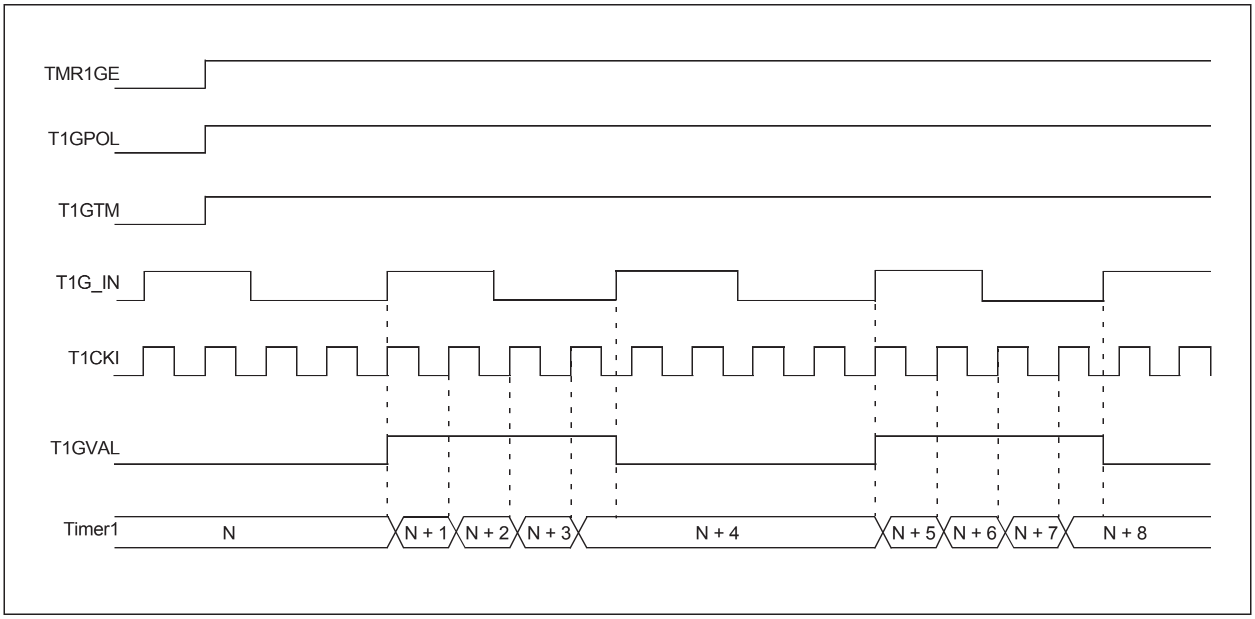
When Timer1 Gate Toggle mode is enabled, it is possible to measure the full cycle length of a Timer1 gate signal, as opposed to the duration of a single level pulse.

The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 13-5 for timing details.

The T1GVAL bit will indicate when the Toggled mode is active and the timer is counting.

The Timer1 Gate Toggle mode is enabled by setting the T1GTM bit of the T1GCON register. When the T1GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

FIGURE 13-5: TIMER1 GATE TOGGLE MODE



13.8.4 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 first enabled by setting the T1GSPM bit in the T1GCON register. Next, the T1GGO/$\overline{\text{T1DONE}}$ bit in the T1GCON register must be set. The Timer1 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the T1GGO/$\overline{\text{T1DONE}}$ 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. See Figure 13-6 for timing details.

Enabling the Toggle mode and the Single Pulse mode, simultaneously, will permit both sections to work together. This allows the cycle times on the Timer1 gate source to be measured. See Figure 13-7 for timing details.

13.8.5 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 in the T1GCON register. The T1GVAL bit is valid even when the Timer1 gate is not enabled (TMR1GE bit is cleared).

FIGURE 13-6: TIMER1 GATE SINGLE PULSE MODE

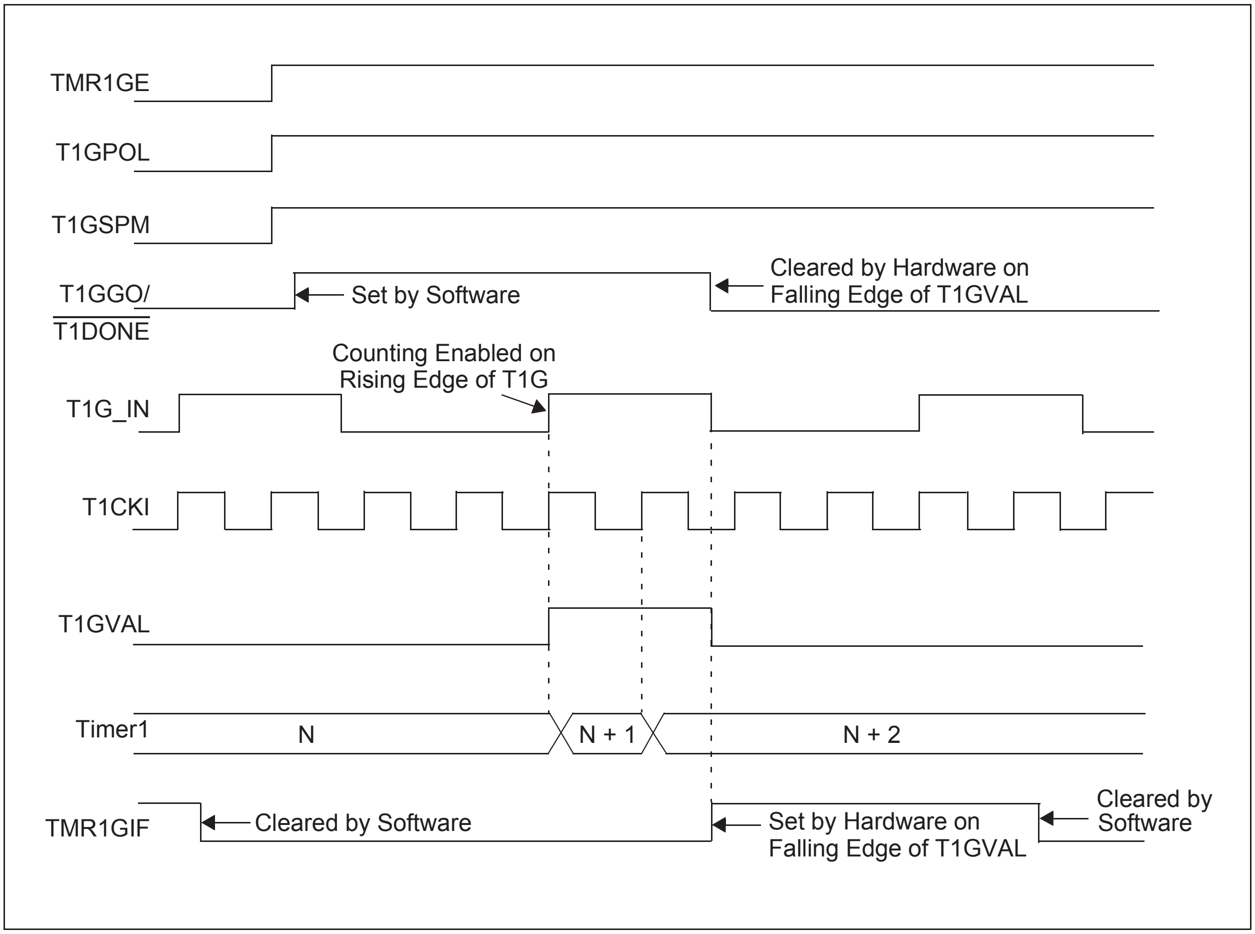

TMR1GE
T1GPOL
T1GSPM
T1GGO/
T1DONE
Set by Software
Cleared by Hardware on
Falling Edge of T1GVAL
Counting Enabled on
Rising Edge of T1G
T1G_IN
T1CKI
T1GVAL
Timer1
N
N + 1
N + 2
TMR1GIF
Cleared by Software
Set by Hardware on
Falling Edge of T1GVAL
Cleared by
Software

FIGURE 13-7: TIMER1 GATE SINGLE PULSE AND TOGGLE COMBINED MODE

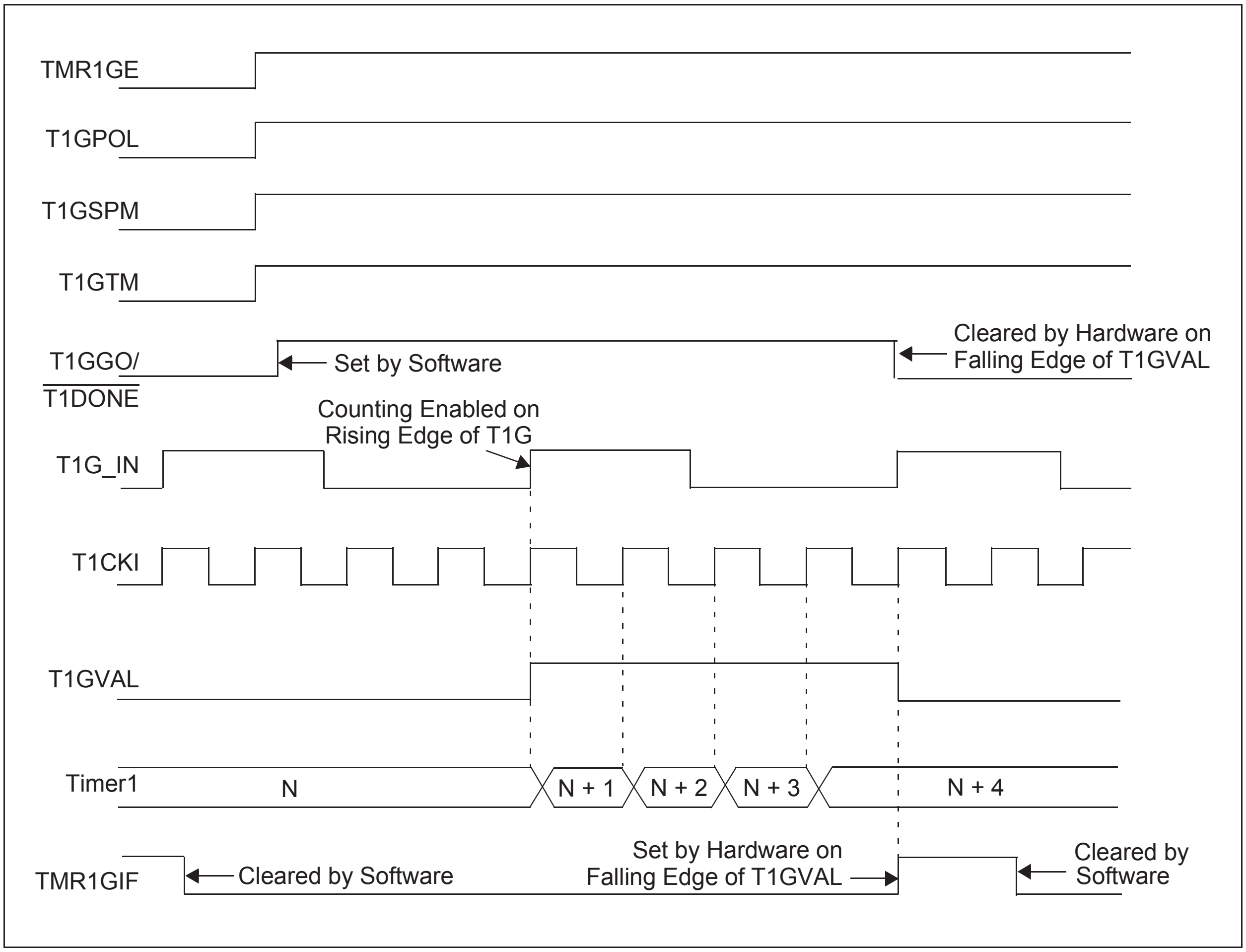

TMR1GE
T1GPOL
T1GSPM
T1GTM
T1GGO/
$\overline{\text{T1DONE}}$
Set by Software
Cleared by Hardware on
Falling Edge of T1GVAL
Counting Enabled on
Rising Edge of T1G
T1G_IN
T1CKI
T1GVAL
Timer1
N
N + 1
N + 2
N + 3
N + 4
TMR1GIF
Cleared by Software
Set by Hardware on
Falling Edge of T1GVAL
Cleared by
Software

TABLE 13-5: 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	PMPIF[1]	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE[1]	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP[1]	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
TMR1L	Timer1 Register Low Byte							
TMR1H	Timer1 Register High Byte							
T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	RD16	TMR1ON
T1GCON	TMR1GE	T1GPOL	T1GTM	T1GSPM	T1GGO/ $\overline{\text{T1DONE}}$	T1GVAL	T1GSS1	T1GSS0
OSCCON2	—	SOSCRUN	—	SOSCDRV	SOSCGO	—	—	—

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the Timer1 module.

Note 1: These bits are only available in 44-pin devices.

14.0 TIMER2 MODULE

The Timer2 module incorporates the following features:

- 8-bit Timer and Period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4 and 1:16)
- Software programmable postscaler (1:1 through 1:16)
- Interrupt on TMR2 to PR2 match
- Optional use as the shift clock for the MSSP modules

The module is controlled through the T2CON register (Register 14-1) which enables or disables the timer and configures the prescaler and postscaler. Timer2 can be shut off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption.

A simplified block diagram of the module is shown in Figure 14-1.

14.1 Timer2 Operation

In normal operation, TMR2 is incremented from 00h on each clock (FOSC/4). A 4-bit counter/prescaler on the clock input gives direct input, divide-by-4 and divide-by-16 prescale options. These are selected by the prescaler control bits, 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 14.2 "Timer2 Interrupt"**).

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, while the PR2 register initializes at FFh. Both the prescaler and postscaler counters are cleared on the following events:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (Power-on Reset (POR), $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset (WDTR) or Brown-out Reset (BOR))

TMR2 is not cleared when T2CON is written.

REGISTER 14-1: T2CON: TIMER2 CONTROL REGISTER (ACCESS FCAh)

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
10 = Prescaler is 16

14.2 Timer2 Interrupt

Timer2 can also generate an optional device interrupt. The Timer2 output signal (TMR2 to PR2 match) provides the input for the 4-bit output counter/postscaler. This counter generates the TMR2 Match Interrupt Flag, which is latched in TMR2IF (PIR1<1>). The interrupt is enabled by setting the TMR2 Match Interrupt Enable bit, TMR2IE (PIE1<1>).

A range of 16 postscaler options (from 1:1 through 1:16 inclusive) can be selected with the postscaler control bits, T2OUTPS<3:0> (T2CON<6:3>).

14.3 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 be optionally used as the shift clock source for the MSSP modules operating in SPI mode. Additional information is provided in **Section 20.0 "Master Synchronous Serial Port (MSSP) Module"**.

FIGURE 14-1: TIMER2 BLOCK DIAGRAM

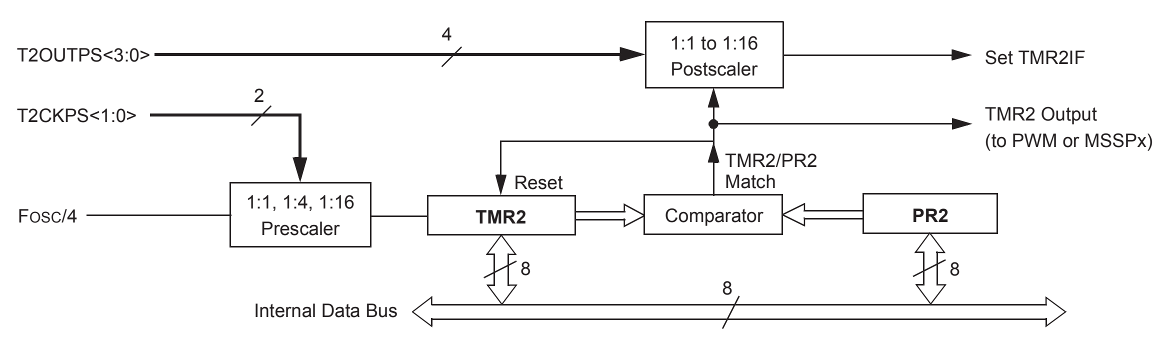


TABLE 14-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

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

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

Note 1: These bits are only available in 44-pin devices.

15.0 TIMER3/5 MODULE

The Timer3/5 timer/counter modules incorporate these features:

- Software selectable operation as a 16-bit timer or counter
- Readable and writable 8-bit registers (TMRxH and TMRxL)
- Selectable clock source (internal or external) with device clock or Timer1 oscillator internal options
- Interrupt-on-overflow
- Module Reset on ECCP Special Event Trigger

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 or Timer5 module. For example, the control register is named TxCON, and refers to T3CON and T5CON.

A simplified block diagram of the Timer3/5 module is shown in Figure 15-1.

The Timer3/5 module is controlled through the TxCON register (Register 15-1). It also selects the clock source options for the ECCP modules. (For more information, see **Section 19.1.1 “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.

REGISTER 15-1: TxCON: TIMER3/5 CONTROL REGISTER (ACCESS F79h, BANKED F22h)

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	TxOSCEN	$\overline{\text{TxSYNC}}$	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>:** Clock Source Select bits

`10` = Clock source is the pin or TxCKI input pin
`01` = Clock source is the system clock (FOSC)[1]
`00` = 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 **TxOSCEN**: Timer Oscillator Enable bit

`1` = T1OSC/SOSC oscillator used as clock source
`0` = TxCKI digital input pin used as clock source

bit 2 **$\overline{\text{TxSYNC}}$:** External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)

When TMRxCS1:TMRxCS0 = `10`:
`1` = Do not synchronize external clock input
`0` = Synchronize external clock input

When TMRxCS1:TMRxCS0 = `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 timer in one 16-bit operation
`0` = Enables register read/write of timer in two eight-bit operations

bit 0 **TMRxON:** Timer On bit

`1` = Enables Timer
`0` = Stops Timer

Note 1: The FOSC clock source should not be selected if the timer will be used with the ECCP capture/compare features.

15.1 Timer3/5 Gate Control Register

The Timer3/5 Gate Control register (TxGCON), provided in Register 14-2, is used to control the Timerx gate.

REGISTER 15-2: TxGCON: TIMER3/5 GATE CONTROL REGISTER(1) (ACCESS F97h, BANKED F21h)

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:** Timer Gate Enable bit

If TMRxON = `0`:
This bit is ignored.

If TMRxON = `1`:
`1` = Timer counting is controlled by the Timerx gate function
`0` = Timer counts regardless of the Timerx gate function

bit 6 **TxGPOL:** Gate Polarity bit

`1` = Timer gate is active-high (Timerx counts when the gate is high)
`0` = Timer gate is active-low (Timerx counts when the gate is low)

bit 5 **TxGTM:** Gate Toggle Mode bit

`1` = Timer Gate Toggle mode is enabled.
`0` = Timer Gate Toggle mode is disabled and toggle flip-flop is cleared
Timerx gate flip-flop toggles on every rising edge.

bit 4 **TxGSPM:** Timer Gate Single Pulse Mode bit

`1` = Timer Gate Single Pulse mode is enabled and is controlling Timerx gate
`0` = Timer Gate Single Pulse mode is disabled

bit 3 **TxGGO/$\overline{\text{TxDONE}}$:** Timer Gate Single Pulse Acquisition Status bit

`1` = Timer gate single pulse acquisition is ready, waiting for an edge
`0` = Timer gate single pulse acquisition has completed or has not been started
This bit is automatically cleared when TxGSPM is cleared.

bit 2 **TxGVAL:** Timer Gate Current State bit

Indicates the current state of the Timer gate that could be provided to TMRxH:TMRxL. Unaffected by the Timer Gate Enable bit (TMRxGE).

bit 1-0 **TxGSS<1:0>:** Timer Gate Source Select bits

`11` = Comparator 2 output
`10` = Comparator 1 output
`01` = TMR4/6 to match PR4/6 output
`00` = T3G/T5G gate input pin

Note 1: Programming the TxGCON prior to TxCON is recommended.

REGISTER 15-3: OSCCON2: OSCILLATOR CONTROL REGISTER 2 (ACCESS F87h)

U-0	R-0^(2)	U-0	R/W-1	R/W-0^(2)	R/W-1	U-0	U-0
—	SOSCRUN	—	SOSCDRV	SOSCGO^(3)	PRISD	—	—
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 secondary SOSC
0 = System clock comes from an oscillator other than SOSC

bit 5 **Unimplemented:** Read as ‘0’

bit 4 **SOSCDRV:** SOSC Drive Control bit

1 = T1OSC/SOSC oscillator drive circuit is selected by Configuration bits, CONFIG2L<4:3>
0 = Low-power T1OSC/SOSC circuit is selected

bit 3 **SOSCGO:** Oscillator Start Control bit

1 = Turns on the oscillator, even if no peripherals are requesting it
0 = Oscillator is shut off unless peripherals are requesting it

bit 2 **PRISD:** Primary Oscillator Drive Circuit Shutdown bit

1 = Oscillator drive circuit is on
0 = Oscillator drive circuit is off (zero power)

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

Note 1: Reset value is ‘0’ when Two-Speed Start-up is enabled and ‘1’ if disabled.
2: Default output frequency of INTOSC on Reset (4 MHz).
3: When the SOSC is selected to run from a digital clock input, rather than an external crystal, this bit has no effect.

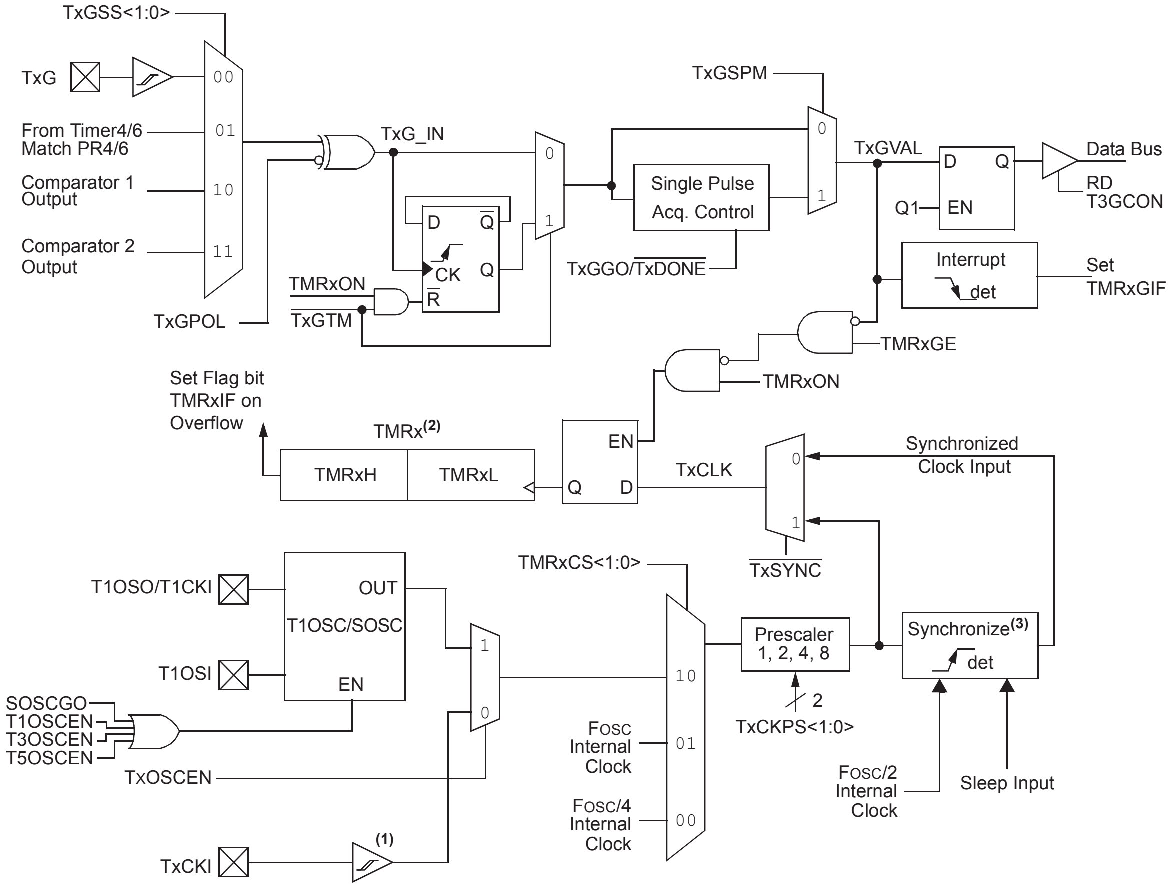
15.2 Timer3/5 Operation

Timer3 and Timer5 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 increments on every internal instruction cycle (FOSC/4). When TMRxCSx = `01`, the Timer3/5 clock source is the system clock (FOSC), and when it is ‘`10`’, Timer3/5 works as a counter from the external clock from the TxCKI pin (on the rising edge after the first falling edge) or the Timer1 oscillator.

FIGURE 15-1: TIMER3/5 BLOCK DIAGRAM


TxGSS<1:0>
TxG
00
From Timer4/6 Match PR4/6
01
Comparator 1 Output
10
Comparator 2 Output
11
TxGPOL
TxG_IN
0
1
D
Q
CK
Q
R
TMRxON
TxGTM
TxGSPM
Single Pulse Acq. Control
TxGGO/TxDONE
TxGVAL
D
Q
Q1
EN
Data Bus
RD T3GCON
Interrupt det
Set TMRxGIF
TMRxGE
TMRxON
Set Flag bit TMRxIF on Overflow
TMRx(2)
TMRxH
TMRxL
EN
Q
D
TxCLK
0
1
TxSYNC
Synchronized Clock Input
T1OSO/T1CKI
OUT
T1OSC/SOSC
EN
T1OSI
SOSCGO
T1OSCEN
T3OSCEN
T5OSCEN
TxOSCEN
TxCKI
(1)
TMRxCS<1:0>
10
01
00
FOSC Internal Clock
FOSC/4 Internal Clock
Prescaler 1, 2, 4, 8
2
TxCKPS<1:0>
Synchronize(3) det
FOSC/2 Internal Clock
Sleep Input

Note 1: ST buffer is a high-speed type when using T1CKI.

2: Timerx registers increment on rising edge.

3: Synchronization does not operate while in Sleep.

15.3 Timer3/5 16-Bit Read/Write Mode

Timer3/5 can be configured for 16-bit reads and writes (see Figure 15.3). 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. A read from TMRxL will load the contents of the high byte of Timer3/5 into the Timerx High Byte Buffer register. This provides users with the ability to accurately read all 16 bits of Timer3/5 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 must also take place through the TMRxH Buffer register. The Timer3/5 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 at once.

The high byte of Timer3/5 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 prescaler. The prescaler is only cleared on writes to TMRxL.

15.4 Using the Timer1 Oscillator as the Timer3/5 Clock Source

The Timer1 internal oscillator may be used as the clock source for Timer3/5. The Timer1 oscillator is enabled by setting the T1OSCEN (T1CON<3>) bit. To use it as the Timer3/5 clock source, the TMRxCS bits must also be set. As previously noted, this also configures Timer3/5 to increment on every rising edge of the oscillator source.

The Timer1 oscillator is described in **Section 13.0 "Timer1 Module"**.

15.5 Timer3/5 Gates

Timer3/5 can be configured to count freely or the count can be enabled and disabled using the Timer3/5 gate circuitry. This is also referred to as the Timer3/5 gate count enable.

The Timer3/5 gate can also be driven by multiple selectable sources.

15.5.1 TIMER3/5 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>).

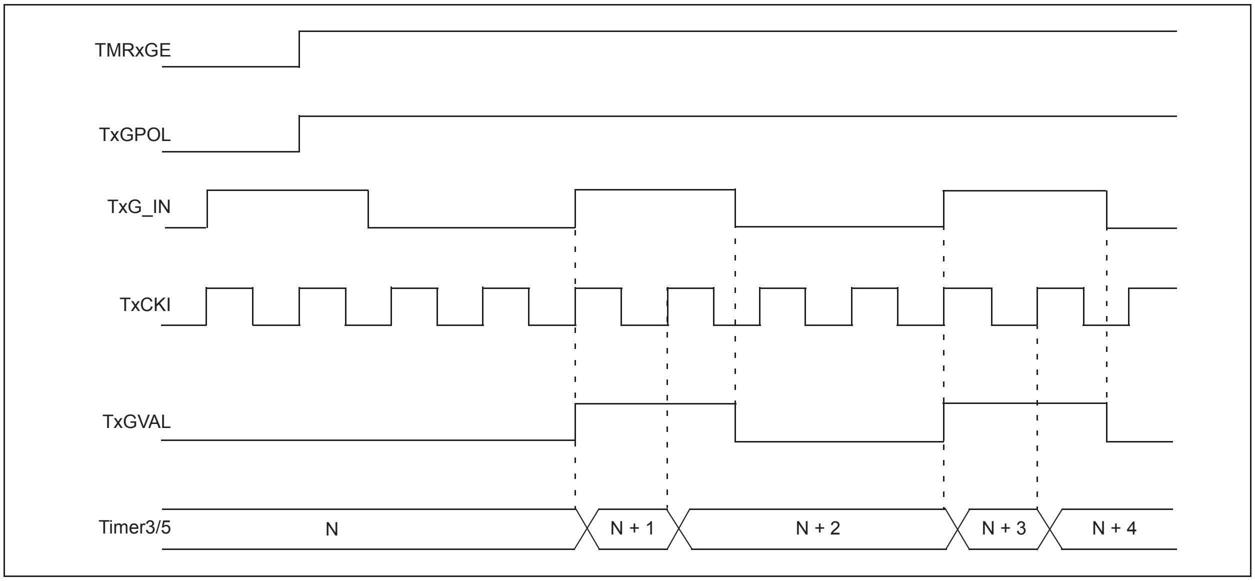
When Timerx Gate Enable mode is enabled, Timer3/5 will increment on the rising edge of the Timer3/5 clock source. When Timerx Gate Enable mode is disabled, no incrementing will occur and Timer3/5 will hold the current count. See Figure 15-2 for timing details.

TABLE 15-1: TIMER3/5 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 is running. For more information, see TxCLK in Figure 15-1.

FIGURE 15-2: TIMER3/5 GATE COUNT ENABLE MODE



15.5.2 TIMER3/5 GATE SOURCE SELECTION

The Timer3/5 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 15-2: TIMER3/5 GATE SOURCES

TxGSS<1:0>	Timerx Gate Source
`00`	TxG timer gate pin
`01`	TMR4/6 matches PR4/6
`10`	Comparator 1 output
`11`	Comparator 2 output

15.5.2.1 TxG Pin Gate Operation

The TxG pin is one source for Timer3/5 gate control. It can be used to supply an external source to the gate circuitry.

15.5.2.2 Timer4/6 Match Gate Operation

The TMR4/6 register will increment until it matches the value in the PR4/6 register. On the very next increment cycle, TMR4/6 will be reset to 00h. When this Reset occurs, a low-to-high pulse will automatically be generated and internally supplied to the Timer3/5 gate circuitry.

15.5.3 TIMER3/5 GATE-TOGGLE MODE

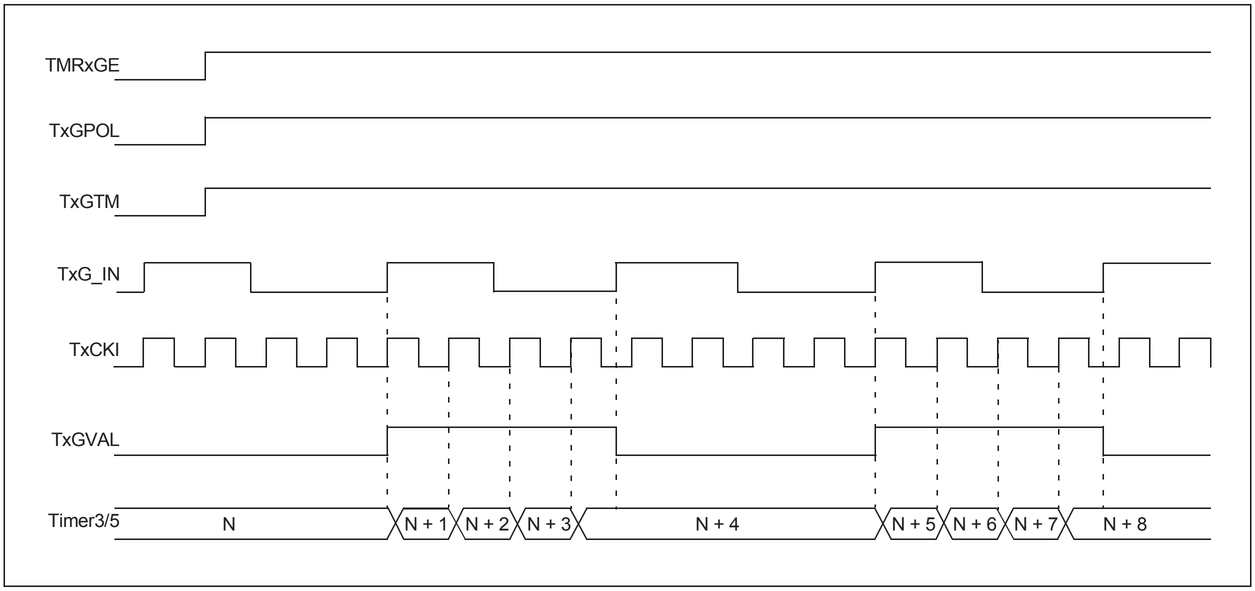
When Timer3/5 Gate Toggle mode is enabled, it is possible to measure the full cycle length of a Timer3/5 gate signal, as opposed to the duration of a single level pulse.

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 15-3.)

The TxGVAL bit will indicate when the Toggled mode is active and the timer is counting.

Timer3/5 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 15-3: TIMER3/5 GATE TOGGLE MODE



15.5.4 TIMER3/5 GATE SINGLE PULSE MODE

When Timer3/5 Gate Single Pulse mode is enabled, it is possible to capture a single pulse gate event. Timer3/5 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 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 until the TxGGO/$\overline{\text{TxDONE}}$ bit is once again set in software.

Clearing the TxGSPM bit will also clear the TxGGO/$\overline{\text{TxDONE}}$ bit. (For timing details, see Figure 15-4.)

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 gate source to be measured. (For timing details, see Figure 15-5.)

FIGURE 15-4: TIMER3/5 GATE SINGLE PULSE MODE

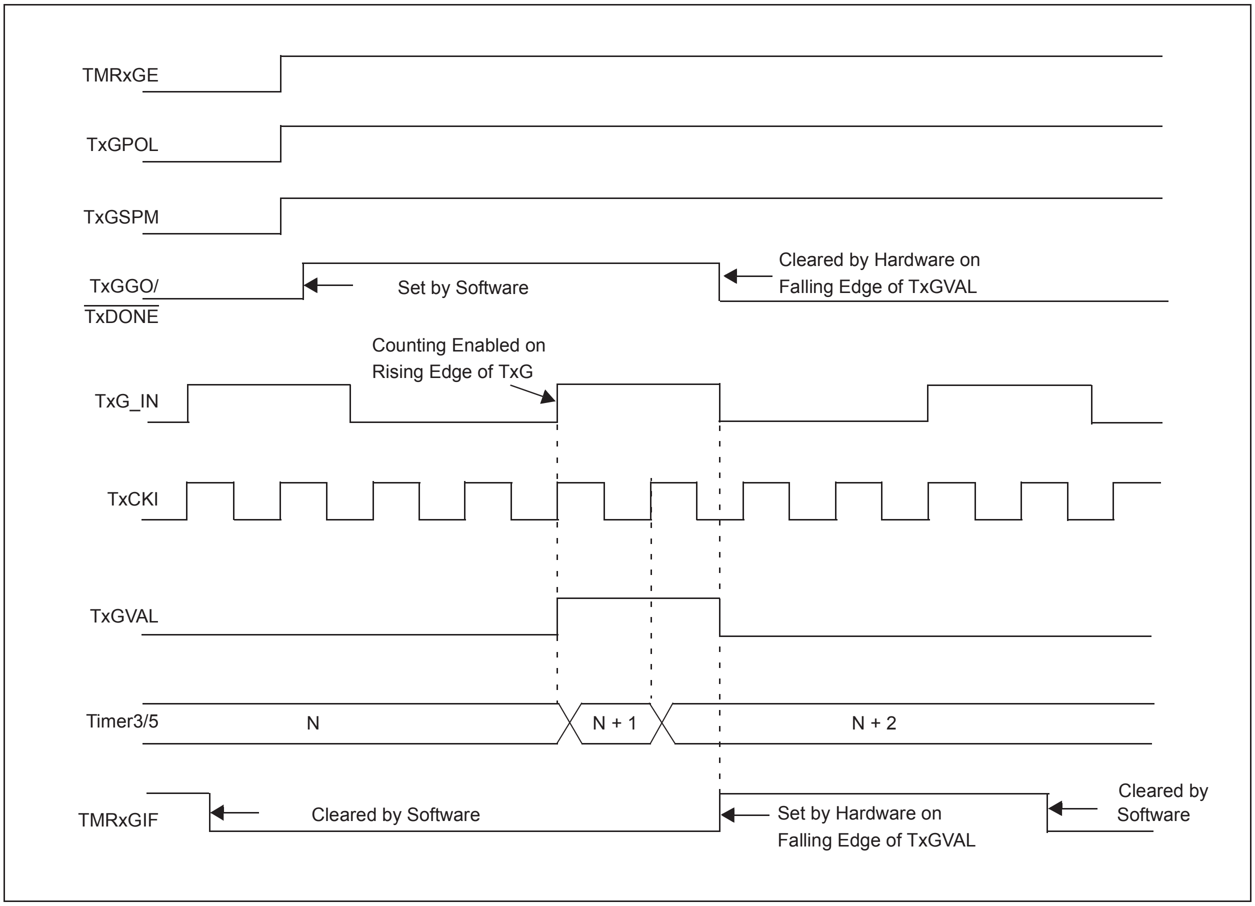
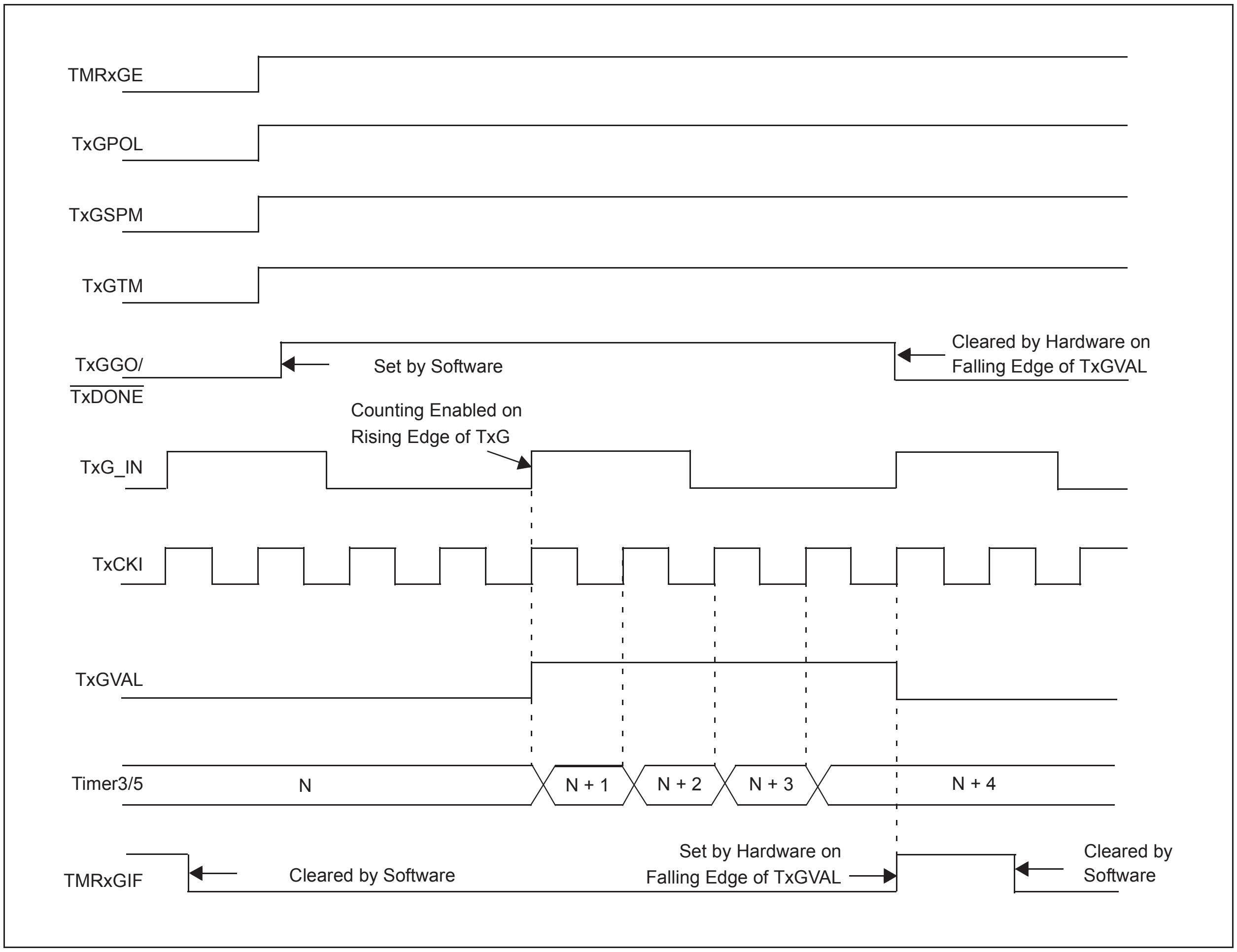

TMRxGE
TxGPOL
TxGSPM
TxGGO/
TxDONE
Set by Software
Cleared by Hardware on
Falling Edge of TxGVAL
Counting Enabled on
Rising Edge of TxG
TxG_IN
TxCKI
TxGVAL
Timer3/5
N
N + 1
N + 2
TMRxGIF
Cleared by Software
Set by Hardware on
Falling Edge of TxGVAL
Cleared by
Software

FIGURE 15-5: TIMER3/5 GATE SINGLE PULSE AND TOGGLE COMBINED MODE


TMRxGE
TxGPOL
TxGSPM
TxGTM
TxGGO/
$\overline{\text{TxDONE}}$
Set by Software
Cleared by Hardware on
Falling Edge of TxGVAL
Counting Enabled on
Rising Edge of TxG
TxG_IN
TxCKI
TxGVAL
Timer3/5
N
N + 1
N + 2
N + 3
N + 4
TMRxGIF
Cleared by Software
Set by Hardware on
Falling Edge of TxGVAL
Cleared by
Software

15.5.5 TIMER3/5 GATE VALUE STATUS

When Timer3/5 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 gate is not enabled (TMRxGE bit is cleared).

15.5.6 TIMER3/5 GATE EVENT INTERRUPT

When the Timer3/5 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 gate is not enabled (TMRxGE bit is cleared).

15.6 Timer3/5 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 15-3 gives each module's flag bit.

TABLE 15-3: TIMER3/5 INTERRUPT FLAG BITS

Timer Module	Flag Bit
3	PIR2<1>
5	PIR5<1>

This interrupt can be enabled or disabled by setting or clearing the TMRxIE bit, respectively. Table 15-4 gives each module's enable bit.

TABLE 15-4: TIMER3/5 INTERRUPT ENABLE BITS

Timer Module	Flag Bit
3	PIE2<1>
5	PIE5<2>

15.7 Resetting Timer3/5 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 19.3.4 "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 18-3 and Register 19-2.

TABLE 15-5: REGISTERS ASSOCIATED WITH TIMER3/5 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	—	—	CM3IF	TMR8IF	TMR6IF	TMR5IF	TMR5GIF	TMR1GIF
PIE5	—	—	CM3IE	TMR8IE	TMR6IE	TMR5IE	TMR5GIE	TMR1GIE
PIR2	OSCFIF	CM2IF	CM1IF	—	BCL1IF	HLVDIF	TMR3IF	CCP2IF
PIE2	OSCFIE	CM2IE	CM1IE	—	BCL1IE	HLVDIE	TMR3IE	CCP2IE
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	T3OSCEN	$\overline{\text{T3SYNC}}$	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	T5OSCEN	$\overline{\text{T5SYNC}}$	RD16	TMR5ON
OSCCON2	—	SOSCRUN	—	SOSCDRV	SOSCGO	—	—	—
CCPTMRS0	C3TSEL1	C3TSEL0	C2TSEL2	C2TSEL1	C2TSEL0	C1TSEL2	C1TSEL1	C1TSEL0
CCPTMRS1	C7TSEL1	C7TSEL0	—	C6TSEL0	—	C5TSEL0	C4TSEL1	C4TSEL0
CCPTMRS1	—	—	—	C10TSEL0	—	C9TSEL0	C8TSEL1	C8TSEL0
CCPTMRS2	—	—	—	C10TSEL0	—	C9TSEL0	C8TSEL1	C8TSEL0

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the Timer3 module.

NOTES:

16.0 TIMER4/6/8 MODULE

The Timer4/6/8 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

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 or Timer8 module. For example, the control register is named TxCON and refers to T4CON, T6CON and T8CON.

The Timer4/6/8 modules have a control register shown in Register 16-1. Timer4/6/8 can be shut off by clearing control bit, TMRxON (TxCON<2>), to minimize power consumption. The prescaler and postscaler selection of Timer4/6/8 is also controlled by this register. Figure 16-1 is a simplified block diagram of the Timer4/6/8 modules.

16.1 Timer4/6/8 Operation

Timer4/6/8 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 16-1 gives each module's flag bit.

TABLE 16-1: TIMER4/6/8 FLAG BITS

Timer Module	Flag Bit
4	PIR3<3>
6	PIR5<3>
8	PIR5<4>

The interrupt can be enabled or disabled by setting or clearing the Timerx Interrupt Enable bit (TMRxIE), shown in Table 16-2.

TABLE 16-2: TIMER4/6/8 INTERRUPT ENABLE BITS

Timer Module	Flag Bit
4	PIE3<3>
6	PIE5<3>
8	PIE5<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), $\overline{\text{MCLR}}$ 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 18-2, Register 18-3 and Register 19-2.

REGISTER 16-1: TxCON: TIMER4/6/8 CONTROL REGISTER (ACCESS F76h, BANKED F1Eh, BANKED F1Bh)

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

16.2 Timer4/6/8 Interrupt

The Timer4/6/8 modules have 8-bit Period registers, PRx, that are both readable and writable. Timer4/6/8 increment from 00h until they match PR4/6/8 and then reset to 00h on the next increment cycle. The PRx registers are initialized to FFh upon Reset.

16.3 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 16-1: TIMER4 BLOCK DIAGRAM

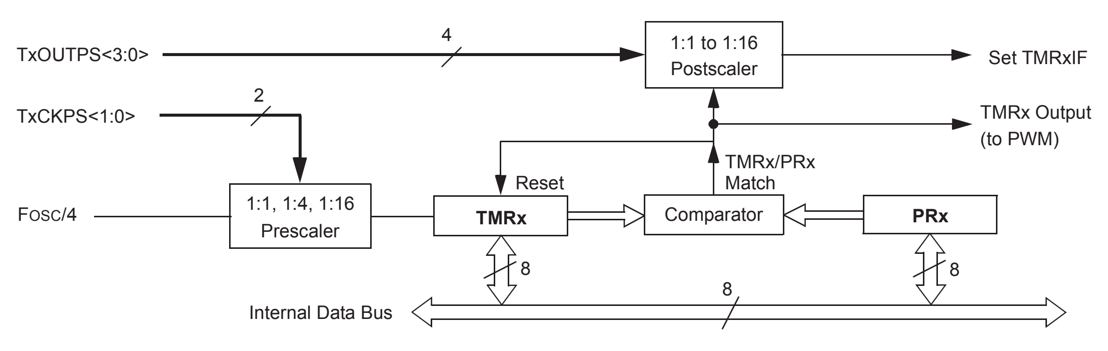

TxOUTPS<3:0>
4
1:1 to 1:16
Postscaler
Set TMRxIF
TxCKPS<1:0>
2
TMRx Output
(to PWM)
TMRx/PRx
Match
Reset
Fosc/4
1:1, 1:4, 1:16
Prescaler
TMRx
Comparator
PRx
8
8
8
Internal Data Bus

TABLE 16-3: REGISTERS ASSOCIATED WITH TIMER4/6/8 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	—	—	CM3IP	TMR8IP	TMR6IP	TMR5IP	TMR5GIP	TMR1GIP
PIR5	—	—	CM3IF	TMR8IF	TMR6IF	TMR5IF	TMR5GIF	TMR1GIF
PIE5	—	—	CM3IE	TMR8IE	TMR6IE	TMR5IE	TMR5GIE	TMR1GIE
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							
CCPTMRS0	C3TSEL1	C3TSEL0	C2TSEL2	C2TSEL1	C2TSEL0	C1TSEL2	C1TSEL1	C1TSEL0
CCPTMRS1	C7TSEL1	C7TSEL0	—	C6TSEL0	—	C5TSEL0	C4TSEL1	C4TSEL0
CCPTMRS2	—	—	—	C10TSEL0	—	C9TSEL0	C8TSEL1	C8TSEL0

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the Timer4/6/8 modules.

17.0 REAL-TIME CLOCK AND CALENDAR (RTCC)

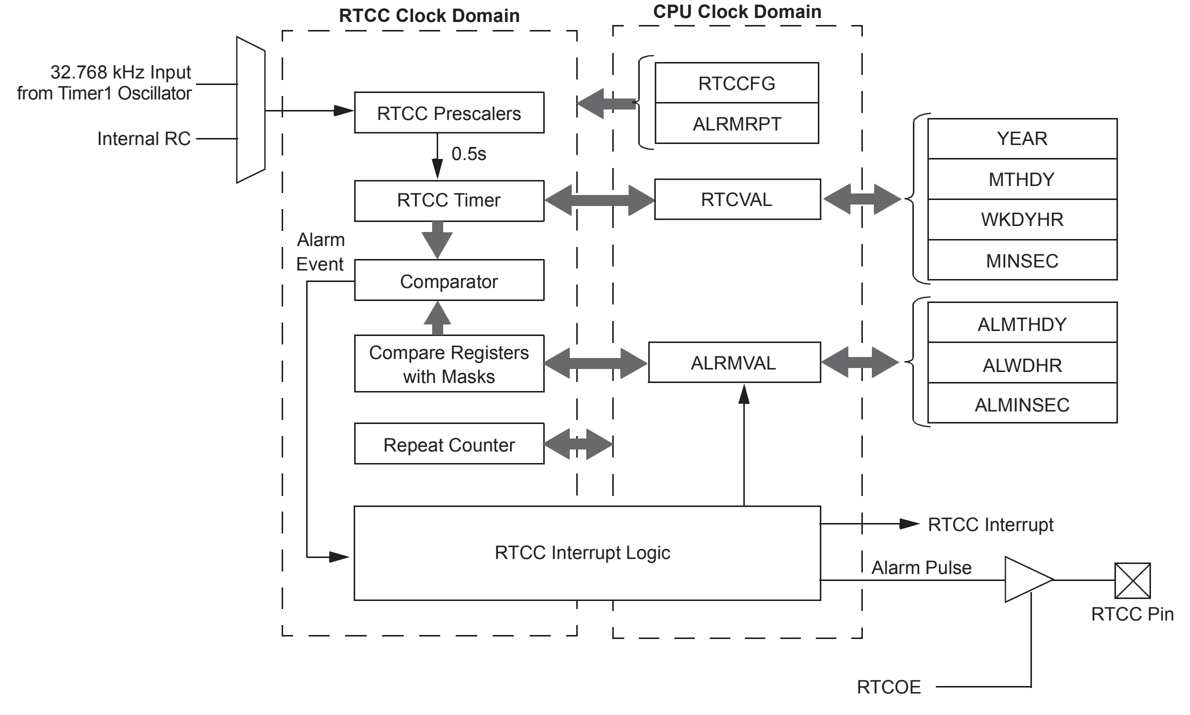
The key features of the Real-Time Clock and Calendar (RTCC) module are:

- Time: hours, minutes and seconds
- 24-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 an 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 17-1: RTCC BLOCK DIAGRAM


RTCC Clock Domain
CPU Clock Domain
32.768 kHz Input
from Timer1 Oscillator
Internal RC
RTCC Prescalers
0.5s
RTCC Timer
Alarm
Event
Comparator
Compare Registers
with Masks
Repeat Counter
RTCC Interrupt Logic
RTCCFG
ALRMRPT
RTCVAL
ALRMVAL
YEAR
MTHDY
WKDYHR
MINSEC
ALMTHDY
ALWDHR
ALMINSEC
RTCC Interrupt
Alarm Pulse
RTCC Pin
RTCOE

17.1 RTCC MODULE REGISTERS

The RTCC module registers are divided into following categories:

RTCC Control Registers

- RTCCFG
- RTCCAL
- PADCFG1
- ALRMCFG
- ALRMRPT

RTCC Value Registers

- RTCVALH and RTCVALL – Can access the following registers
  - YEAR
  - MONTH
  - DAY
  - WEEKDAY
  - HOUR
  - MINUTE
  - SECOND

Alarm Value Registers

- ALRMVALH and ALRMVALL – Can access the following registers:
  - ALRMMNTH
  - ALRMDAY
  - ALRMWD
  - ALRMHR
  - ALRMMIN
  - ALRMSEC

Note: The RTCVALH and RTCVALL registers can be accessed through RTCRPT<1:0>. ALRMVALH and ALRMVALL can be accessed through ALRMPTR<1:0>.

17.1.1 RTCC CONTROL REGISTERS

REGISTER 17-1: RTCCFG: RTCC CONFIGURATION REGISTER (BANKED F3Fh)[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	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

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 ALCFGRPT registers can change while reading due to a rollover ripple resulting 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 or 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 enabled
0 = RTCC clock output 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 until it reaches ‘00’.

RTCVAL<15:8>:
00 = Minutes
01 = Weekday
10 = Month
11 = Reserved

RTCVAL<7:0>:
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.

REGISTER 17-2: RTCCAL: RTCC CALIBRATION REGISTER (BANKED F3Eh)

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>:** RTCC Drift Calibration bits

`01111111` = Maximum positive adjustment; adds 508 RTC clock pulses every minute
.
.
.
`00000001` = Minimum positive adjustment; adds four RTCC clock pulses every minute
`00000000` = No adjustment
`11111111` = Minimum negative adjustment; subtracts four RTCC clock pulses every minute
.
.
.
`10000000` = Maximum negative adjustment; subtracts 512 RTCC clock pulses every minute

REGISTER 17-3: PADCFG1: PAD CONFIGURATION REGISTER 1 (BANKED F3Ch)

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	RTSECSEL1(1)	RTSECSEL0(1)	PMPTTL(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 7-3 **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 (pin can be INTRC or T1OSC, depending on the RTCOSC (CONFIG3L<1>) setting)
`01` = RTCC seconds clock is selected for the RTCC pin
`00` = RTCC alarm pulse is selected for the RTCC pin

bit 0 **PMPTTL:** PMP Module TTL Input Buffer Select bit(2)

`1` = PMP module uses TTL input buffers
`0` = PMP module uses Schmitt input buffers

Note 1: To enable the actual RTCC output, the RTCOE (RTCCFG<2>) bit must be set.

2: Available only on 44-pin devices (PIC18F46J13, PIC18F47J13, PIC18LF46J13 and PIC18LF47J13). For 28-pin devices, the bit is U-0.

REGISTER 17-4: ALRMCFG: ALARM CONFIGURATION REGISTER (ACCESS F47h)

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> = `0000 0000` and CHIME = `0`)
0 = Alarm is disabled

bit 6 **CHIME:** Chime Enable bit

1 = Chime is enabled; ARPT<7:0> bits are allowed to roll over from 00h to FFh
0 = Chime is disabled; ARPT<7:0> 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’.

<u>ALRMVAL<15:8>:</u>
00 = ALRMMIN
01 = ALRMWD
10 = ALRMMNTH
11 = Unimplemented

<u>ALRMVAL<7:0>:</u>
00 = ALRMSEC
01 = ALRMHR
10 = ALRMDAY
11 = Unimplemented

REGISTER 17-5: ALRMRPT: ALARM REPEAT COUNTER (ACCESS F46h)

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`.

17.1.2 RTCVALH AND RTCVALL REGISTER MAPPINGS

REGISTER 17-6: 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 17-7: 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`.

REGISTER 17-8: 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 17-9: 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 17-10: WKDY: 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 17-11: HOURS: 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 17-12: MINUTES: 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 17-13: SECONDS: 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.

17.1.3 ALRMVALH AND ALRMVALL REGISTER MAPPINGS

REGISTER 17-14: 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 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 17-15: 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 17-16: 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 17-17: 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 17-18: 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 17-19: 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.

17.1.4 RTCEN BIT WRITE

An attempt to write to the RTCEN bit while RTCWREN = `0` will be ignored. RTCWREN must be set before a write to RTCEN can take place.

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.

17.2 Operation

17.2.1 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 17-2 and Figure 17-3).

FIGURE 17-2: TIMER DIGIT FORMAT

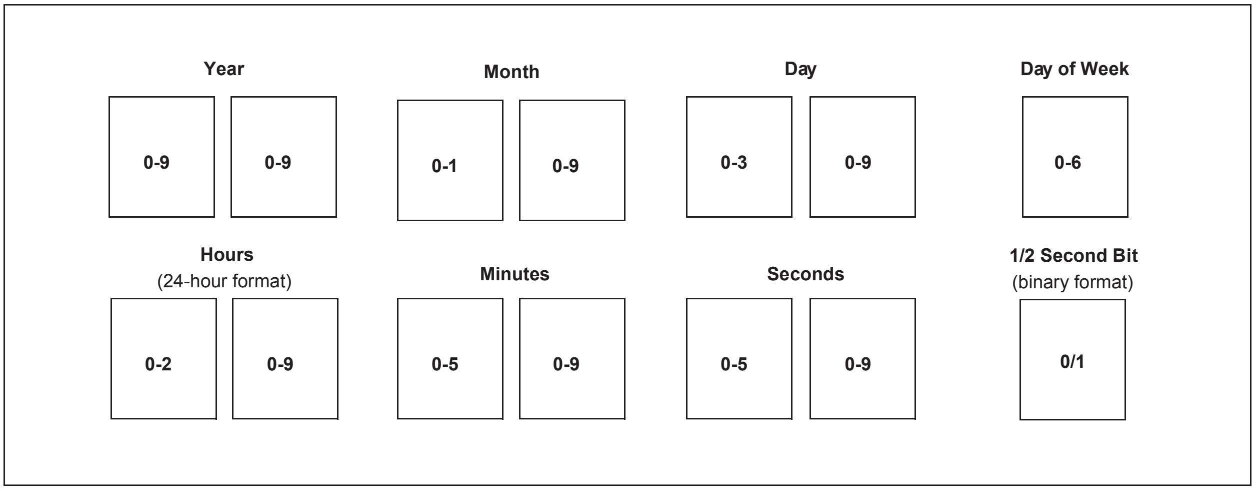
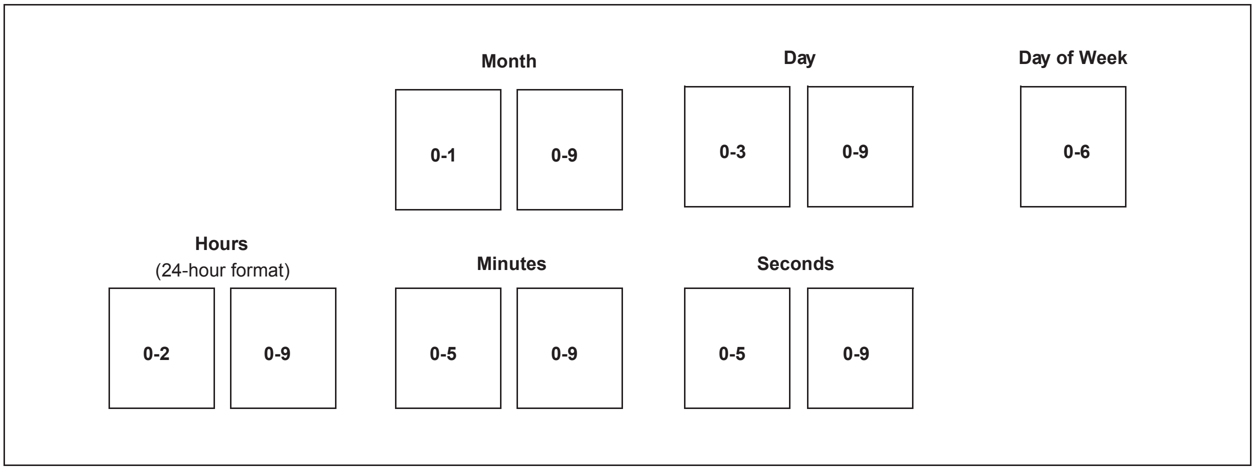

Year
0-9
0-9
Month
0-1
0-9
Day
0-3
0-9
Day of Week
0-6
Hours
(24-hour format)
0-2
0-9
Minutes
0-5
0-9
Seconds
0-5
0-9
1/2 Second Bit
(binary format)
0/1

FIGURE 17-3: ALARM DIGIT FORMAT

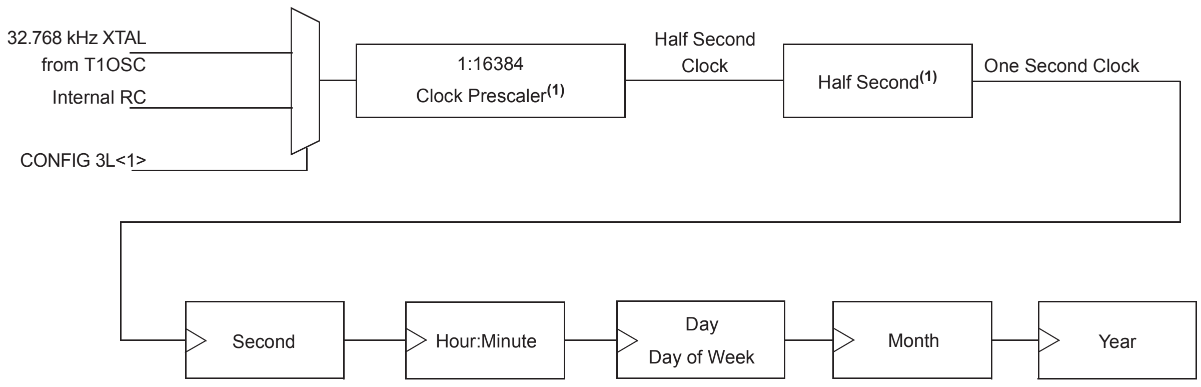

Month
0-1
0-9
Day
0-3
0-9
Day of Week
0-6
Hours
(24-hour format)
0-2
0-9
Minutes
0-5
0-9
Seconds
0-5
0-9

17.2.2 CLOCK SOURCE

As mentioned earlier, the RTCC module is intended to be clocked by an external Real-Time Clock (RTC) crystal oscillating at 32.768 kHz, but can also be clocked by the INTRC. The RTCC clock selection is decided by the RTCOSC bit (CONFIG3L<1>).

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 17.2.9 “Calibration”**.)

FIGURE 17-4: CLOCK SOURCE MULTIPLEXING



Note 1: Writing to the lower half of the MINSEC register resets all counters, allowing fraction of a second synchronization. The clock prescaler is held in Reset when RTCEN = `0`.

17.2.2.1 Real-Time Clock Enable

The RTCC module can be clocked by an external, 32.768 kHz crystal (Timer1 oscillator or T1CKI input) or the INTRC oscillator, which can be selected in CONFIG3L<1>.

If the Timer1 oscillator will be used as the clock source for the RTCC, make sure to enable it by setting T1CON<3> (T1OSCEN). The selected RTC clock can be brought out to the RTCC pin by the RTSECSEL<1:0> bits in the PADCFG1 register.

17.2.3 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 17-1)
- Year Carry: From 99 to 00; this also surpasses the use of the RTCC

For the day to month rollover schedule, see Table 17-2.

Considering that the following values are in BCD format, the carry to the upper BCD digit will occur at a count of 10 and not at 16 (SECONDS, MINUTES, HOURS, WEEKDAY, DAYS and MONTHS).

TABLE 17-1: DAY OF WEEK SCHEDULE

Day of Week	
Sunday	0
Monday	1
Tuesday	2
Wednesday	3
Thursday	4
Friday	5
Saturday	6

TABLE 17-2: 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 17.2.4 "Leap Year"**.

17.2.4 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 effected in a leap year.

February will have 29 days in a leap year and 28 days in any other year.

17.2.5 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 17.2.8 "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 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.

17.2.6 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.

17.2.7 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 at any time other than while writing to it. 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 17-1.

EXAMPLE 17-1: SETTING THE RTCWREN BIT

```
movlb   0x0F            ;RTCCFG is banked
bcf     INTCON, GIE     ;Disable interrupts
movlw   0x55
movwf   EECON2
movlw   0xAA
movwf   EECON2
bsf     RTCCFG, RTCWREN
```

17.2.8 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<15:8> and RTCVALL<7:0>) uses the RTCPTR 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`'. Once it reaches '`00`', the MINUTES and SECONDS value will be accessible through RTCVALH and RTCVALL until the pointer value is manually changed.

TABLE 17-3: RTCVALH AND RTCVALL REGISTER MAPPING

RTCPTR<1:0>	RTCC Value Register Window	
	RTCVAL<15:8>	RTCVAL<7:0>
`00`	MINUTES	SECONDS
`01`	WEEKDAY	HOURS
`10`	MONTH	DAY
`11`	—	YEAR

The Alarm Value register window (ALRMVALH and ALRMVALL) uses 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 1 until it reaches '`00`'. Once it reaches '`00`', the ALRMMIN and ALRMSEC value will be accessible through ALRMVALH and ALRMVALL until the pointer value is manually changed.

TABLE 17-4: ALRMVAL REGISTER MAPPING

ALRMPTR<1:0>	Alarm Value Register Window	
	ALRMVAL<15:8>	ALRMVAL<7:0>
`00`	ALRMMIN	ALRMSEC
`01`	ALRMWD	ALRMHR
`10`	ALRMMNTH	ALRMDAY
`11`	—	—

17.2.9 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 in the lower half of the RTCCAL register. The 8-bit, signed value, loaded into RTCCAL, is multiplied by four and will either be 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 17-1).

EQUATION 17-1: CONVERTING ERROR CLOCK PULSES

$$\text{(Ideal Frequency (32,768) – Measured Frequency)} * 60 = \text{Error Clocks per Minute}$$

- If the oscillator is *faster* than ideal (negative result from Step 2), the RTCCFG 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 RTCCFG 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.

17.3 Alarm

The alarm features and characteristics are:

- Configurable from half a second to one year
- Enabled using the ALRMEN bit (ALRMCFG<7>, Register 17-4)
- Offers one time and repeat alarm options

17.3.1 CONFIGURING THE ALARM

The alarm feature is enabled using the ALRMEN bit.

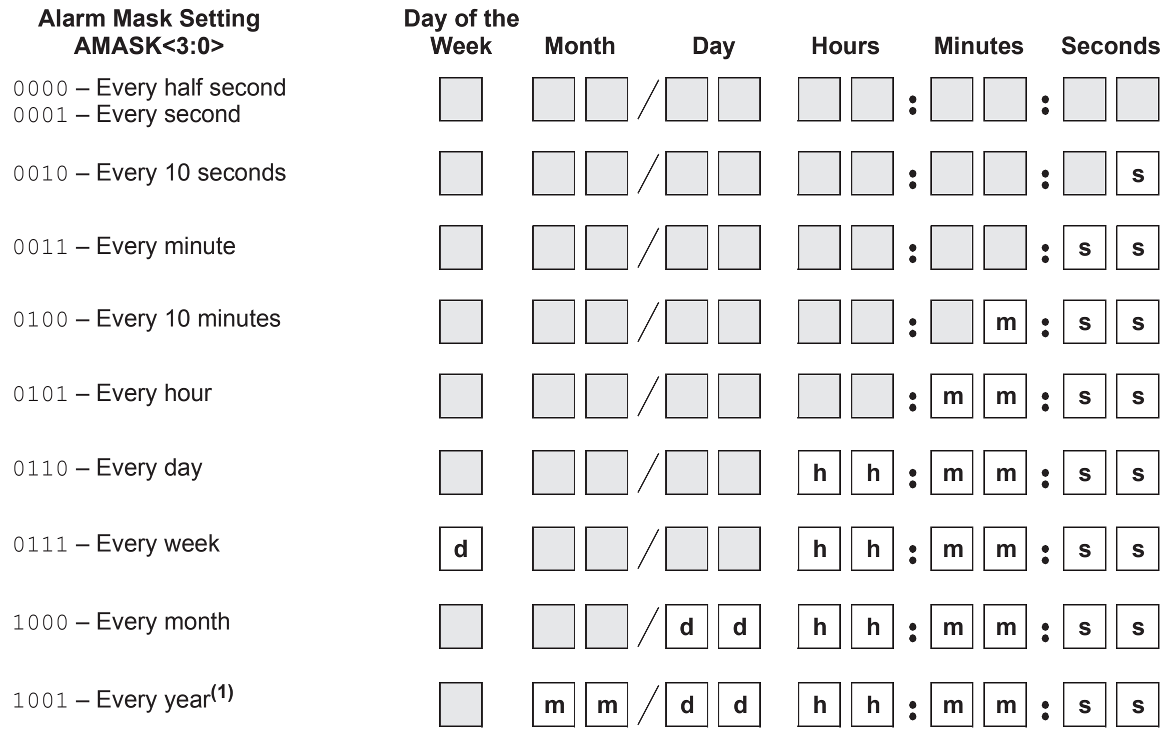
This bit is cleared when an alarm is issued. The bit will not be cleared if the CHIME bit = `1` or if ALRMRPT ≠ `0`.

The interval selection of the alarm is configured through the ALRMCFG (AMASK<3:0>) bits (see Figure 17-5). 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`.

FIGURE 17-5: ALARM MASK SETTINGS



Alarm Mask Setting AMASK<3:0>	Day of the Week	Month		Day		Hours		Minutes		Seconds	
`0000` – Every half second `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 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.

17.3.2 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.

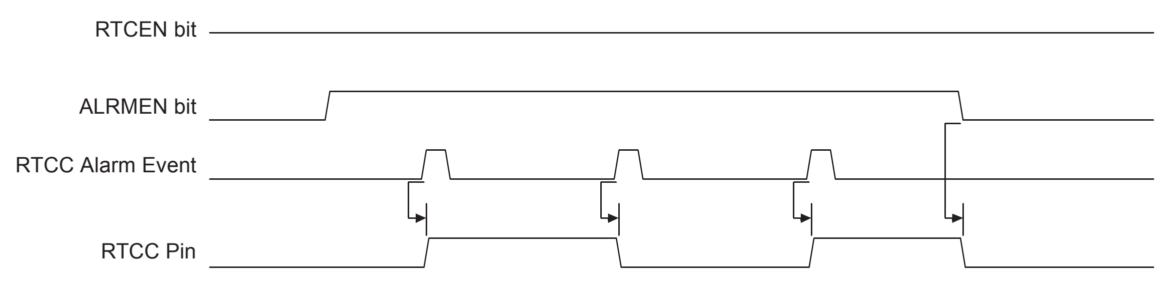
The alarm pulse output is completely synchronous with the RTCC clock and can be used as a trigger clock to other peripherals. This output is available on the RTCC pin. The output pulse is a clock with a 50% duty cycle and a frequency half that of the alarm event (see Figure 17-6).

The RTCC pin can also output the seconds clock. The user can select between the alarm pulse, generated by the RTCC module, or the seconds clock output.

The RTSECSEL (PADCFG1<2:1>) bits select between these two outputs:

- Alarm pulse – RTSECSEL<2:1> = `00`
- Seconds clock – RTSECSEL<2:1> = `01`

FIGURE 17-6: TIMER PULSE GENERATION



17.4 Low-Power Modes

The timer and alarm can optionally continue to operate while in Sleep, Idle and even Deep Sleep mode. An alarm event can be used to wake-up the microcontroller from any of these Low-Power modes.

17.5 Reset

17.5.1 DEVICE RESET

When a device Reset occurs, the ALCFGRPT 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.

17.5.2 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.

17.6 Register Maps

Table 17-5, Table 17-6 and Table 17-7 summarize the registers associated with the RTCC module.

TABLE 17-5: RTCC CONTROL REGISTERS

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
RTCCFG	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0	0000
RTCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	0000
PADCFG1	—	—	—	—	—	RTSECSEL1	RTSECSEL0	PMPTTL	0000
ALRMCFG	ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0	0000
ALRMRPT	ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0	0000
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCCIF	0000
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCCIE	0000
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCCIP	0000

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal for 44-pin devices.

TABLE 17-6: RTCC VALUE REGISTERS

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
RTCVALH	RTCC Value Register Window High Byte, Based on RTCPTR<1:0>								xxxx
RTCVALL	RTCC Value Register Window Low Byte, Based on RTCPTR<1:0>								xxxx
RTCCFG	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0	0000
ALRMCFG	ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0	0000
ALRMVALH	Alarm Value Register Window High Byte, Based on ALRMPTR<1:0>								xxxx
ALRMVALL	Alarm Value Register Window Low Byte, Based on ALRMPTR<1:0>								xxxx

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal for 44-pin devices.

TABLE 17-7: ALARM VALUE REGISTERS

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
ALRMRPT	ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0	0000
ALRMVALH	Alarm Value Register Window High Byte, Based on ALRMPTR<1:0>								xxxx
ALRMVALL	Alarm Value Register Window Low Byte, Based on ALRMPTR<1:0>								xxxx
RTCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	0000
RTCVALH	RTCC Value Register Window High Byte, Based on RTCPTR<1:0>								xxxx
RTCVALL	RTCC Value Register Window Low Byte, Based on RTCPTR<1:0>								xxxx

Legend: — = unimplemented, read as '0'. Reset values are shown in hexadecimal for 44-pin devices.

18.0 CAPTURE/COMPARE/PWM (CCP) MODULES

PIC18F47J13 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.

REGISTER 18-1: CCPxCON: CCP4-10 CONTROL REGISTER (4, BANKED F12h; 5, F0Fh; 6, F0Ch; 7, F09h; 8, F06h; 9, F03h; 10, F00h)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3(1)	CCPxM2(1)	CCPxM1(1)	CCPxM0(1)
bit 7							bit 0

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

bit 7-6 **Unimplemented:** Read as '`0`'

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

bit 3-0 **CCPxM<3:0>**: CCPx Module Mode Select bits(1)

`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: CCPxM<3:0> = `1011` will only reset the timer and not start an A/D conversion on CCPx match.

REGISTER 18-2: CCPTMRS1: CCP4-10 TIMER SELECT 1 REGISTER (BANKED F51h)

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 bit `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 18-3: CCPTMRS2: CCP4-10 TIMER SELECT 2 REGISTER (BANKED F50h)

U-0	U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	C10TSEL0	—	C9TSEL0	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

0 = CCP10 is based off of TMR1/TMR2
1 = Reserved; do not use

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **C9TSEL0:** CCP9 Timer Selection bit

0 = CCP9 is based off of TMR1/TMR2
1 = CCP9 is based off of TMR1/TMR4

bit 1-0 **C8TSEL<1:0>:** CCP8 Timer Selection bits

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

REGISTER 18-4: CCPRxL: CCP4-10 PERIOD LOW BYTE REGISTER (4, BANKED F13h; 5, F10h; 6, F0Dh; 7, F0Ah; 8, F07h; 9, F04h; 10, F01h)

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: PWM Period register low byte

REGISTER 18-5: CCPRxH: CCP4-10 PERIOD HIGH BYTE REGISTER (4, BANKED F14h; 5, F11h; 6, F0Eh; 7, F0Bh; 8, F08h; 9, F05h; 10, F02h)

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: PWM Period register high byte

18.1 CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (generically, CCPxCON) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte). All registers are both readable and writable.

18.1.1 CCP MODULES AND TIMER RESOURCES

The CCP modules utilize Timers, 1 through 8, varying with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in Table 18-1.

TABLE 18-1: CCP MODE – TIMER RESOURCE

CCP Mode	Timer Resource
Capture	Timer1, Timer3 or Timer5
Compare	
PWM	Timer2, Timer4, Timer6 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 18-2 and Register 18-3.) 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 18-2 and Table 18-3.

TABLE 18-2: 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 bit configuration.

TABLE 18-3: TIMER ASSIGNMENTS FOR CCP MODULES 8, 9 AND 10

CCPTMRS2 Register											
CCP8			CCP8 Devices with 64 Kbyte			CCP9			CCP10		
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	TMR1	TMR4	0 1	TMR1	TMR4	1	TMR1	TMR4	1	Reserved(1)	
1 0	TMR1	TMR6	1 0	TMR1	TMR6						
1 1	Reserved(1)		1 1	Reserved(1)							

Note 1: Do not use the reserved bit configuration.

18.1.2 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 (ODCON1<7:0> and ODCON2<3:2>). Setting the appropriate bit configures the pin for the corresponding module for open-drain operation.

18.2 Capture Mode

In Capture mode, the CCPR4H:CCPR4L register pair captures the 16-bit value of the TMR1 or TMR3 register when an event occurs on the CCP4 pin, RB4. 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 register, CCPR4, is read, the old captured value is overwritten by the new captured value.

Figure 18-1 shows the Capture mode block diagram.

18.2.1 CCP PIN CONFIGURATION

In Capture mode, the appropriate CCPx pin should be configured as an input by setting the corresponding TRISx direction bit.

Note: If RB4 is configured as a CCP4 output, a write to the port causes a capture condition.

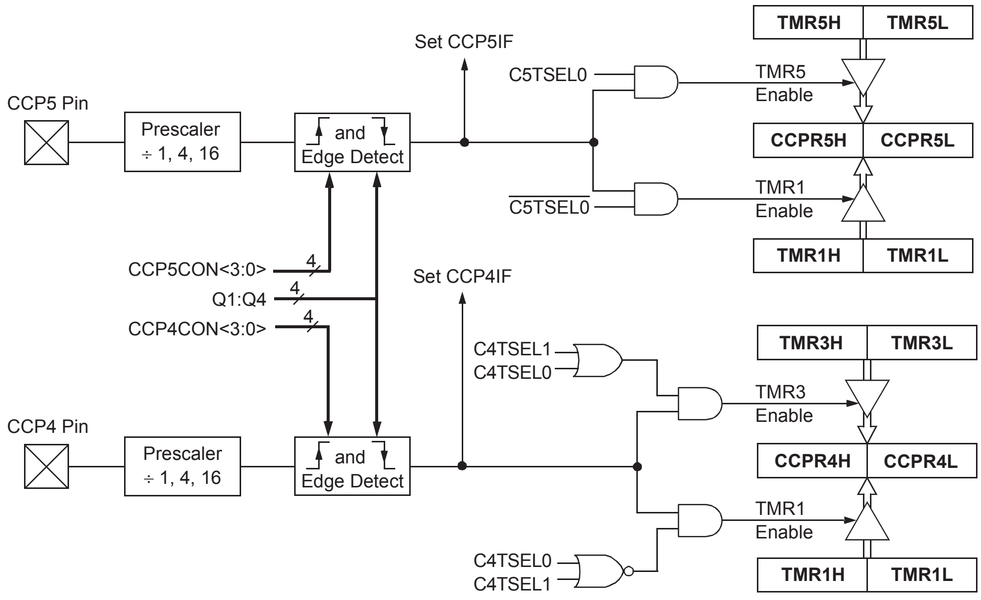
18.2.2 TIMER1/3/5 MODE SELECTION

For the available timers (1/3/5) 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 18.1.1 "CCP Modules and Timer Resources"**.)

Details of the timer assignments for the CCP modules are given in Table 18-2 and Table 18-3.

FIGURE 18-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



TMR5H
TMR5L
Set CCP5IF
C5TSEL0
TMR5 Enable
CCP5 Pin
Prescaler ÷ 1, 4, 16
and Edge Detect
CCPR5H
CCPR5L
$\overline{\text{C5TSEL0}}$
TMR1 Enable
TMR1H
TMR1L
CCP5CON<3:0>
4
Q1:Q4
4
CCP4CON<3:0>
4
Set CCP4IF
C4TSEL1
C4TSEL0
TMR3H
TMR3L
TMR3 Enable
CCP4 Pin
Prescaler ÷ 1, 4, 16
and Edge Detect
CCPR4H
CCPR4L
TMR1 Enable
C4TSEL0
C4TSEL1
TMR1H
TMR1L

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 18-2 and Table 18-3.

18.2.3 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.

18.2.4 CCP PRESCALER

There are four prescaler settings in Capture mode. They are specified as part of the operating mode selected by the mode select bits (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 also will not clear the prescaler counter – meaning the first capture may be from a non-zero prescaler.

Example 18-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

EXAMPLE 18-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
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
```

18.3 Compare Mode

In Compare mode, the 16-bit CCPR4 register value is constantly compared against either the TMR1 or TMR3 register pair value. 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 18-2 gives the Compare mode block diagram

18.3.1 CCP PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the appropriate TRISx bit.

Note:	Clearing the CCP4CON register will force the RB4 compare output latch (depending on device configuration) to the default low level. This is not the PORTB I/O data latch.

18.3.2 TIMER1/3/5 MODE SELECTION

If the CCP module is using the compare feature in conjunction with any of the Timer1/3/5 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 18-2 and Table 18-3.

18.3.3 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.

18.3.4 SPECIAL EVENT TRIGGER

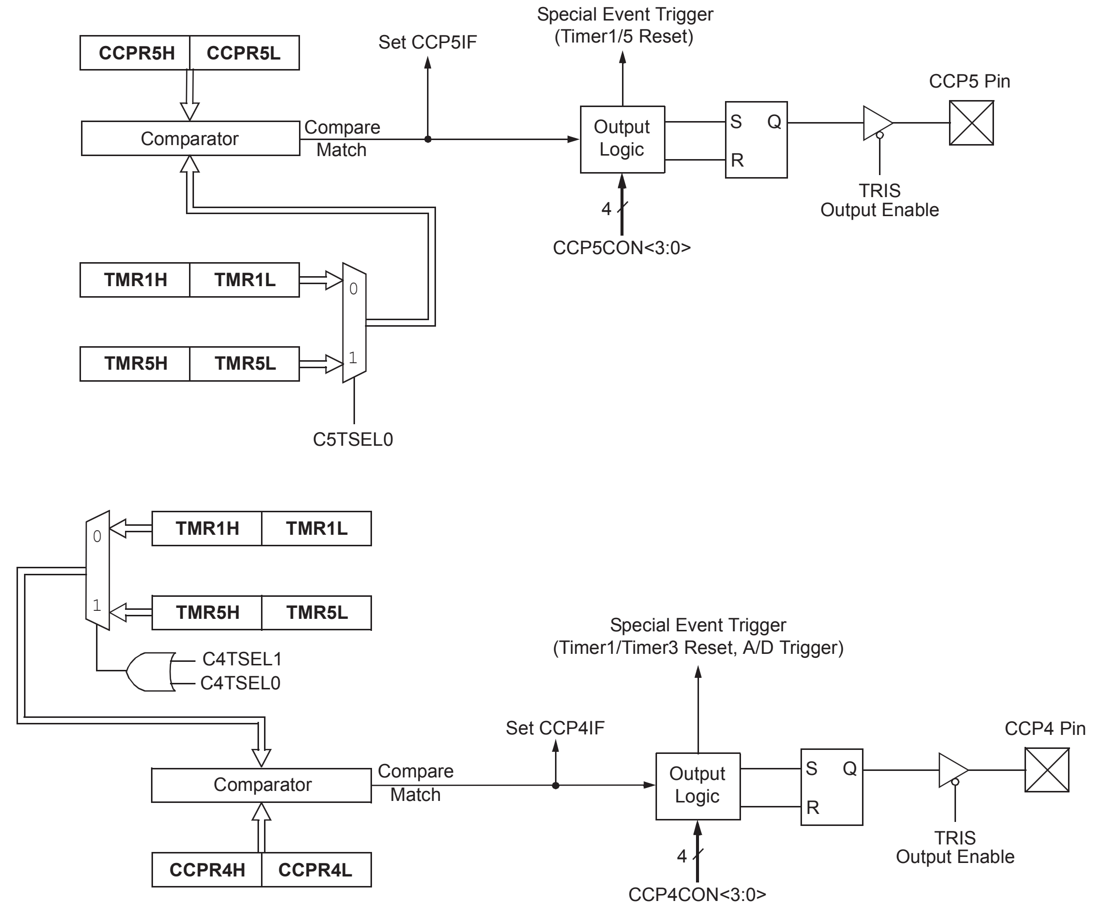
Both CCP modules are equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode (CCP4M<3:0> = `1011`).

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 ECCP1 can start an A/D conversion, but the A/D Converter must be enabled. For more information, see **Section 19.0 “Enhanced Capture/Compare/PWM (ECCP) Module”**.

FIGURE 18-2: COMPARE MODE OPERATION BLOCK DIAGRAM


CCPR5H
CCPR5L
Set CCP5IF
Special Event Trigger
(Timer1/5 Reset)
CCP5 Pin
Comparator
Compare
Match
Output
Logic
S
Q
R
TRIS
Output Enable
4
CCP5CON<3:0>
TMR1H
TMR1L
0
TMR5H
TMR5L
1
C5TSEL0
TMR1H
TMR1L
0
TMR5H
TMR5L
1
C4TSEL1
C4TSEL0
Special Event Trigger
(Timer1/Timer3 Reset, A/D Trigger)
Set CCP4IF
CCP4 Pin
Comparator
Compare
Match
Output
Logic
S
Q
R
TRIS
Output Enable
4
CCP4CON<3:0>
CCPR4H
CCPR4L

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 18-2 and Table 18-3.

TABLE 18-4: 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	—	$\overline{\text{CM}}$	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
PIR4	CCP10IF	CCP9IF	CCP8IF	CCP7IF	CCP6IF	CCP5IF	CCP4IF	CCP3IF
PIE4	CCP10IE	CCP9IE	CCP8IE	CCP7IE	CCP6IE	CCP5IE	CCP4IE	CCP3IE
IPR4	CCP10IP	CCP9IP	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	$\overline{\text{RDPU}}$	$\overline{\text{REPU}}$	—	—	—	TRISE2	TRISE1	TRISE0
TMR1L	Timer1 Register Low Byte							
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							
T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	RD16	TMR1ON
T3CON	TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	T3OSCEN	$\overline{\text{T3SYNC}}$	RD16	TMR3ON
T5CON	TMR5CS1	TMR5CS0	T5CKPS1	T5CKPS0	T5OSCEN	$\overline{\text{T5SYNC}}$	RD16	TMR5ON
CCPR4L	CCPR4L7	CCPR4L6	CCPR4L5	CCPR4L4	CCPR4L3	CCPR4L2	CCPR4L1	CCPR4L0
CCPR4H	CCPR4H7	CCPR4H6	CCPR4H5	CCPR4H4	CCPR4H3	CCPR4H2	CCPR4H1	CCPR4H0
CCPR5L	CCPR5L7	CCPR5L6	CCPR5L5	CCPR5L4	CCPR5L3	CCPR5L2	CCPR5L1	CCPR5L0
CCPR5H	CCPR5H7	CCPR5H6	CCPR5H5	CCPR5H4	CCPR5H3	CCPR5H2	CCPR5H1	CCPR5H0
CCPR6L	CCPR6L7	CCPR6L6	CCPR6L5	CCPR6L4	CCPR6L3	CCPR6L2	CCPR6L1	CCPR6L0
CCPR6H	CCPR6H7	CCPR6H6	CCPR6H5	CCPR6H4	CCPR6H3	CCPR6H2	CCPR6H1	CCPR6H0
CCPR7L	CCPR7L7	CCPR7L6	CCPR7L5	CCPR7L4	CCPR7L3	CCPR7L2	CCPR7L1	CCPR7L0
CCPR7H	CCPR7H7	CCPR7H6	CCPR7H5	CCPR7H4	CCPR7H3	CCPR7H2	CCPR7H1	CCPR7H0
CCPR8L	CCPR8L7	CCPR8L6	CCPR8L5	CCPR8L4	CCPR8L3	CCPR8L2	CCPR8L1	CCPR8L0
CCPR8H	CCPR8H7	CCPR8H6	CCPR8H5	CCPR8H4	CCPR8H3	CCPR8H2	CCPR8H1	CCPR8H0
CCPR9L	CCPR9L7	CCPR9L6	CCPR9L5	CCPR9L4	CCPR9L3	CCPR9L2	CCPR9L1	CCPR9L0
CCPR9H	CCPR9H7	CCPR9H6	CCPR9H5	CCPR9H4	CCPR9H3	CCPR9H2	CCPR9H1	CCPR9H0
CCPR10L	CCPR10L7	CCPR10L6	CCPR10L5	CCPR10L4	CCPR10L3	CCPR10L2	CCPR10L1	CCPR10L0
CCPR10H	CCPR10H7	CCPR10H6	CCPR10H5	CCPR10H4	CCPR10H3	CCPR10H2	CCPR10H1	CCPR10H0
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	—	—	DC9B1	DC9B0	CCP9M3	CCP9M2	CCP9M1	CCP9M0
CCP10CON	—	—	DC10B1	DC10B0	CCP10M3	CCP10M2	CCP10M1	CCP10M0
CCPTMRS1	C7TSEL1	C7TSEL0	—	C6TSEL0	—	C5TSEL0	C4TSEL1	C4TSEL0
CCPTMRS2	—	—	—	C10TSEL0	—	C9TSEL0	C8TSEL1	C8TSEL0

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by capture/compare or Timer1/3/5.

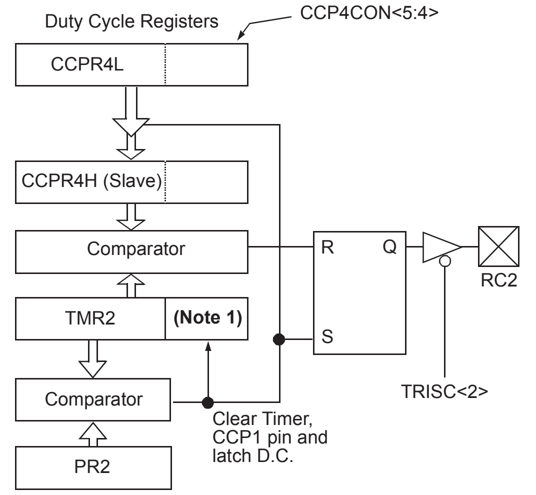
18.4 PWM Mode

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 PORTB data latch, the appropriate TRISx bit must be cleared to make the CCP4 pin an output.

Figure 18-3 shows a simplified block diagram of the CCP1 module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see **Section 18.4.3 "Setup for PWM Operation"**.

FIGURE 18-3: SIMPLIFIED PWM BLOCK DIAGRAM[(2)]

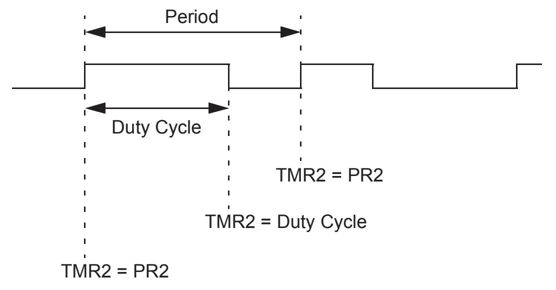

Duty Cycle Registers
CCP4CON<5:4>
CCPR4L
CCPR4H (Slave)
Comparator
TMR2
(Note 1)
Comparator
PR2
R
Q
S
RC2
TRISC<2>
Clear Timer, CCP1 pin and latch D.C.

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 18-2 and Table 18-3.

A PWM output (Figure 18-4) 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 18-4: PWM OUTPUT


Period
Duty Cycle
TMR2 = PR2
TMR2 = Duty Cycle
TMR2 = PR2

18.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

EQUATION 18-1:

$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

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 the 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 14.0 "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.

18.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified 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> contains 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 18-2:

$$\text{PWM Duty Cycle} = (\text{CCPR4L:CCP4CON<5:4>}) \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

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 given by the equation:

EQUATION 18-3:

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)}\text{bits}$$

Note: If the PWM duty cycle value is longer than the PWM period, the CCP4 pin will not be cleared.

TABLE 18-5: 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

18.4.3 SETUP FOR PWM OPERATION

To configure the CCP module for PWM operation:

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 TRISx bit.
4. Set the TMR2 prescale value, then enable Timer2 by writing to T2CON.
5. Configure the CCP4 module for PWM operation.

TABLE 18-6: 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	—	$\overline{\text{CM}}$	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
PIR4	CCP10IF	CCP9IF	CCP8IF	CCP7IF	CCP6IF	CCP5IF	CCP4IF	CCP3IF
PIE4	CCP10IE	CCP9IE	CCP8IE	CCP7IE	CCP6IE	CCP5IE	CCP4IE	CCP3IE
IPR4	CCP10IP	CCP9IP	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	$\overline{\text{RDPU}}$	$\overline{\text{REPU}}$	—	—	—	TRISE2	TRISE1	TRISE0
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	CCPR4L7	CCPR4L6	CCPR4L5	CCPR4L4	CCPR4L3	CCPR4L2	CCPR4L1	CCPR4L0
CCPR4H	CCPR4H7	CCPR4H6	CCPR4H5	CCPR4H4	CCPR4H3	CCPR4H2	CCPR4H1	CCPR4H0
CCPR5L	CCPR5L7	CCPR5L6	CCPR5L5	CCPR5L4	CCPR5L3	CCPR5L2	CCPR5L1	CCPR5L0
CCPR5H	CCPR5H7	CCPR5H6	CCPR5H5	CCPR5H4	CCPR5H3	CCPR5H2	CCPR5H1	CCPR5H0
CCPR6L	CCPR6L7	CCPR6L6	CCPR6L5	CCPR6L4	CCPR6L3	CCPR6L2	CCPR6L1	CCPR6L0
CCPR6H	CCPR6H7	CCPR6H6	CCPR6H5	CCPR6H4	CCPR6H3	CCPR6H2	CCPR6H1	CCPR6H0
CCPR7L	CCPR7L7	CCPR7L6	CCPR7L5	CCPR7L4	CCPR7L3	CCPR7L2	CCPR7L1	CCPR7L0
CCPR7H	CCPR7H7	CCPR7H6	CCPR7H5	CCPR7H4	CCPR7H3	CCPR7H2	CCPR7H1	CCPR7H0
CCPR8L	CCPR8L7	CCPR8L6	CCPR8L5	CCPR8L4	CCPR8L3	CCPR8L2	CCPR8L1	CCPR8L0
CCPR8H	CCPR8H7	CCPR8H6	CCPR8H5	CCPR8H4	CCPR8H3	CCPR8H2	CCPR8H1	CCPR8H0
CCPR9L	CCPR9L7	CCPR9L6	CCPR9L5	CCPR9L4	CCPR9L3	CCPR9L2	CCPR9L1	CCPR9L0
CCPR9H	CCPR9H7	CCPR9H6	CCPR9H5	CCPR9H4	CCPR9H3	CCPR9H2	CCPR9H1	CCPR9H0
CCPR10L	CCPR10L7	CCPR10L6	CCPR10L5	CCPR10L4	CCPR10L3	CCPR10L2	CCPR10L1	CCPR10L0
CCPR10H	CCPR10H7	CCPR10H6	CCPR10H5	CCPR10H4	CCPR10H3	CCPR10H2	CCPR10H1	CCPR10H0
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	—	—	DC9B1	DC9B0	CCP9M3	CCP9M2	CCP9M1	CCP9M0
CCP10CON	—	—	DC10B1	DC10B0	CCP10M3	CCP10M2	CCP10M1	CCP10M0
CCPTMRS1	C7TSEL1	C7TSEL0	—	C6TSEL0	—	C5TSEL0	C4TSEL1	C4TSEL0
CCPTMRS2	—	—	—	C10TSEL0	—	C9TSEL0	C8TSEL1	C8TSEL0

Legend: — = unimplemented, read as ‘`0`’. Shaded cells are not used by PWM or Timer2/4/6/8.

19.0 ENHANCED CAPTURE/COMPARE/PWM (ECCP) MODULE

PIC18F47J13 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 upwardly 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.

The ECCP modules 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 19.4 "PWM (Enhanced Mode)"**.

REGISTER 19-1: CCPxCON: ECCP1/2/3 CONTROL (1, ACCESS FBAh; 2, FB4h; 3, BANKED F15h)

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 assigned as capture/compare input/output; PxB, PxC and PxD assigned as port pins

If CCPxM<3:2> = `11`:

`00` = Single output: PxA, PxB, PxC and PxD are controlled by steering (see **Section 19.4.7 “Pulse Steering Mode”**)
`01` = Full-bridge output forward: PxD modulated; PxA active; PxB, PxC inactive
`10` = Half-bridge output: PxA, PxB modulated with dead-band control; PxC and PxD assigned as port pins
`11` = Full-bridge output reverse: PxB modulated; PxC active; PxA and PxD 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 ECCPRxL.

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 16th rising edge
`1000` = Compare mode; initialize ECCPx pin low, set output on compare match (set CCPxIF)
`1001` = Compare mode; initialize ECCPx pin high, clear output on compare match (set CCPxIF)
`1010` = Compare mode; generate software interrupt only, ECCPx pin reverts to I/O state
`1011` = Compare mode; trigger special event (ECCPx resets TMR1 or TMR3, starts A/D conversion, sets CCPxIF bit)
`1100` = PWM mode; PxA and PxC active-high; PxB and PxD active-high
`1101` = PWM mode; PxA and PxC active-high; PxB and PxD active-low
`1110` = PWM mode; PxA and PxC active-low; PxB and PxD active-high
`1111` = PWM mode; PxA and PxC active-low; PxB and PxD active-low

REGISTER 19-2: CCPTMRS0: ECCP1/2/3 TIMER SELECT 0 REGISTER (BANKED F52h)

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
`1xx` = 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
`1xx` = 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

19.1 ECCP Outputs and Configuration

The Enhanced CCP module may have up to four PWM outputs, depending on the selected operating mode. These outputs, designated PxA through PxD, are routed through the Peripheral Pin Select (PPS) module. Therefore, individual functions can be mapped to any of the remappable I/O pins (RPn).

The outputs that are active depend on the ECCP operating mode selected. The pin assignments are summarized in Table 19-3.

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.

19.1.1 ECCP MODULE AND TIMER RESOURCES

The ECCP modules use Timer1, 2, 3, 4, 6 or 8, depending on the mode selected. These timers are available to CCP modules in Capture, Compare or PWM modes, as shown in Table 19-1.

TABLE 19-1: ECCP MODE – TIMER RESOURCE

ECCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
PWM	Timer2, Timer4, Timer6 or Timer8

The assignment of a particular timer to a module is determined by the Timer to ECCP enable bits in the CCPTMRS0 register (Register 19-2). The interactions between the two modules are depicted in Figure 19-1. 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.

19.2 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 register<3:0>). When a capture is made, the interrupt request flag bit, CCPxIF, is set (see Table 19-2). The flag must be cleared by software. If another capture occurs before the value in the CCPRxH/L register pair is read, the old captured value is overwritten by the new captured value.

TABLE 19-2: ECCP1/2/3 INTERRUPT FLAG BITS

ECCP Module	Flag-Bit
1	PIR1<2>
2	PIR2<0>
3	PIR4<0>

19.2.1 ECCP PIN CONFIGURATION

In Capture mode, the appropriate ECCPx pin should be configured as an input by setting the corresponding TRISx direction bit.

Additionally, the ECCPx input function needs to be assigned to an I/O pin through the Peripheral Pin Select module. For details on setting up the remappable pins, see **Section 10.7 “Peripheral Pin Select (PPS)”**.

Note: If the ECCPx pin is configured as an output, a write to the port can cause a capture condition.

19.2.2 TIMER1/TIMER3 MODE SELECTION

The timers that are to be used with the capture feature (Timer1 and/or Timer3) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work. The timer to be used with each ECCP module is selected in the CCPTMRS0 register (Register 19-2).

19.2.3 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.

19.2.4 ECCP PRESCALER

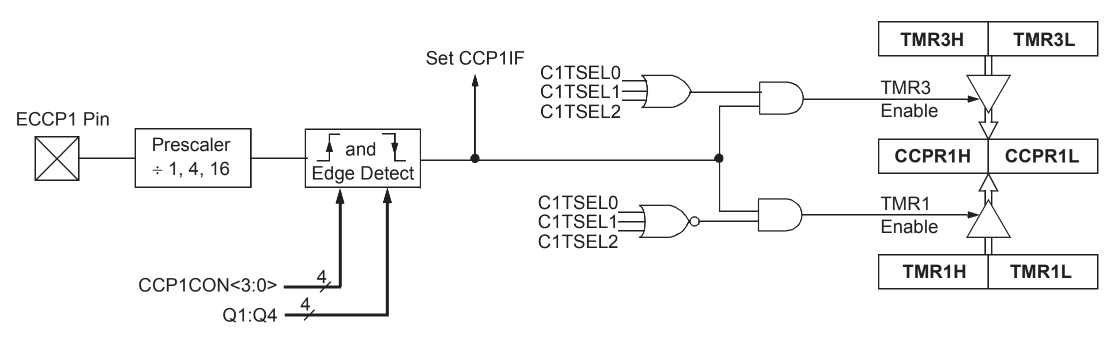
There are four prescaler settings in Capture mode; they are specified as part of the operating mode selected by the mode select bits (CCPxM<3:0>). Whenever the ECCP module is turned off, or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler 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 19-1 provides the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

EXAMPLE 19-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF   ECCP1CON     ; Turn ECCP module off
MOVLW  NEW_CAPT_PS  ; Load WREG with the
                    ; new prescaler mode
                    ; value and ECCP ON
MOVWF  CCP1CON      ; Load ECCP1CON with
                    ; this value
```

FIGURE 19-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



TMR3H
TMR3L
Set CCP1IF
C1TSEL0
C1TSEL1
C1TSEL2
TMR3 Enable
ECCP1 Pin
Prescaler ÷ 1, 4, 16
and Edge Detect
CCPR1H
CCPR1L
C1TSEL0
C1TSEL1
C1TSEL2
TMR1 Enable
TMR1H
TMR1L
CCP1CON<3:0>
4
Q1:Q4
4

19.3 Compare Mode

In Compare mode, the 16-bit CCPRx register pair value is constantly compared against either the TMR1 or TMR3 register pair value. 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.

19.3.1 ECCP PIN CONFIGURATION

Users must configure the ECCPx pin as an output by clearing the appropriate TRISx 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.

19.3.2 TIMER1/TIMER3 MODE SELECTION

Timer1 and/or Timer3 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.

19.3.3 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.

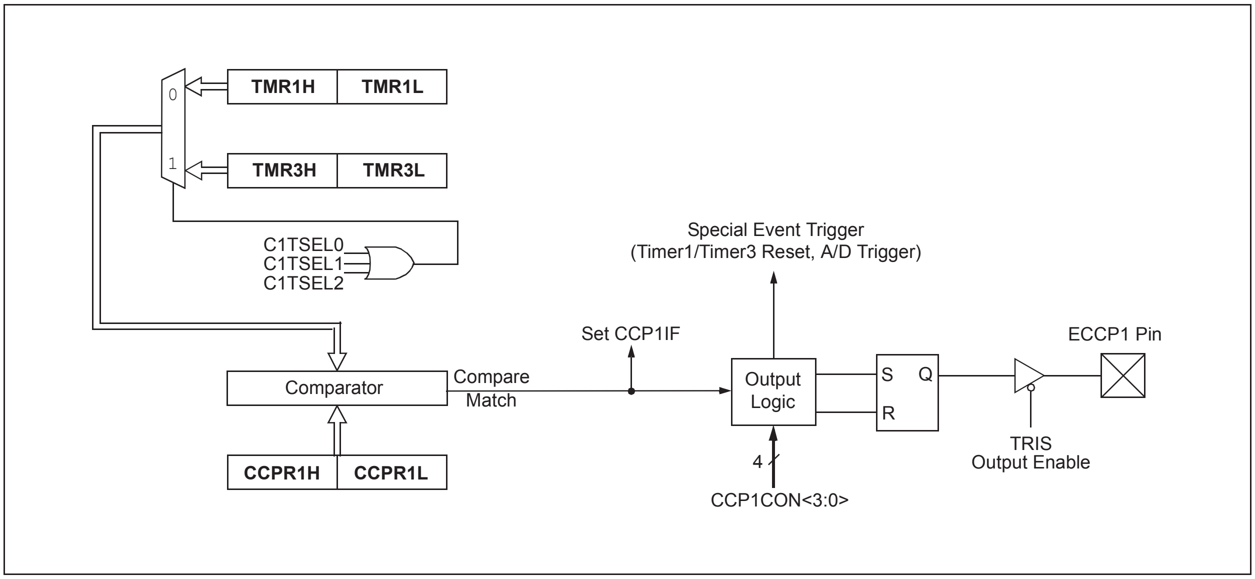
19.3.4 SPECIAL EVENT TRIGGER

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 19-2: COMPARE MODE OPERATION BLOCK DIAGRAM


0
1
TMR1H
TMR1L
TMR3H
TMR3L
C1TSEL0
C1TSEL1
C1TSEL2
Special Event Trigger
(Timer1/Timer3 Reset, A/D Trigger)
Set CCP1IF
ECCP1 Pin
Comparator
Compare
Match
Output
Logic
S
Q
R
TRIS
Output Enable
4
CCP1CON<3:0>
CCPR1H
CCPR1L

19.4 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 mode
- Half-Bridge PWM mode
- 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.

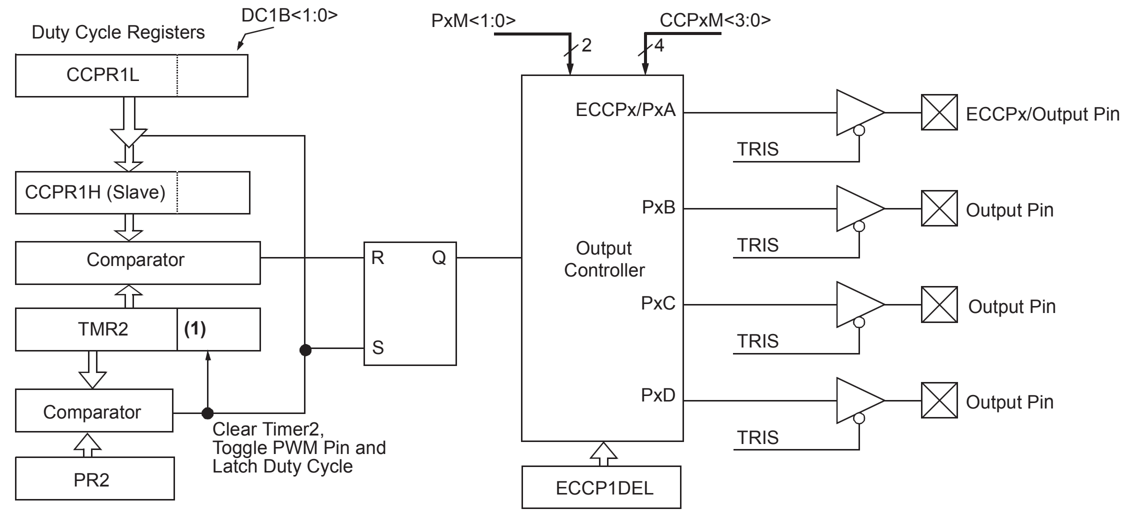
The PWM outputs are multiplexed with I/O pins and are designated: PxA, PxB, PxC and PxD. The polarity of the PWM pins is configurable and is selected by setting the CCPxM bits in the CCPxCON register appropriately.

Table 19-1 provides the pin assignments for each Enhanced PWM mode.

Figure 19-3 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 19-3: SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODE EXAMPLE


Duty Cycle Registers
DC1B<1:0>
PxM<1:0>
CCPxM<3:0>
2
4
CCPR1L
CCPR1H (Slave)
Comparator
TMR2
(1)
Comparator
PR2
R
Q
S
Clear Timer2,
Toggle PWM Pin and
Latch Duty Cycle
ECCPx/PxA
PxB
Output
Controller
PxC
PxD
ECCP1DEL
TRIS
TRIS
TRIS
TRIS
ECCPx/Output Pin
Output Pin
Output Pin
Output Pin

Note 1: The 8-bit timer, 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.

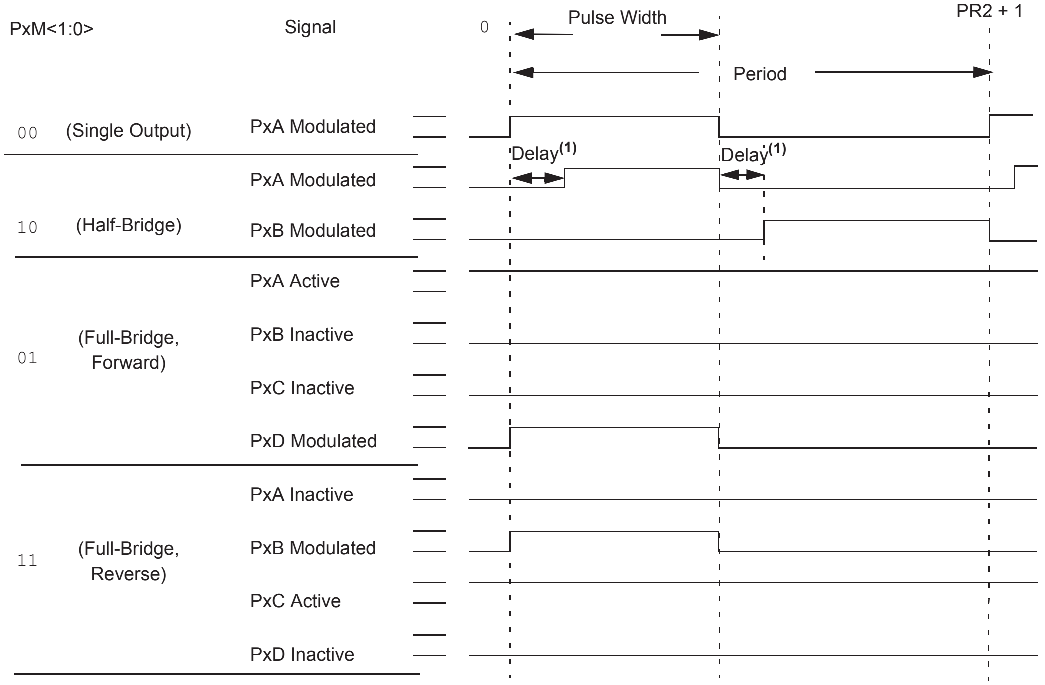
TABLE 19-3: EXAMPLE PIN ASSIGNMENTS FOR VARIOUS PWM ENHANCED MODES

TABLE 19-3:

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 19-6).

FIGURE 19-4: PWM (ENHANCED MODE) OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE) EXAMPLE

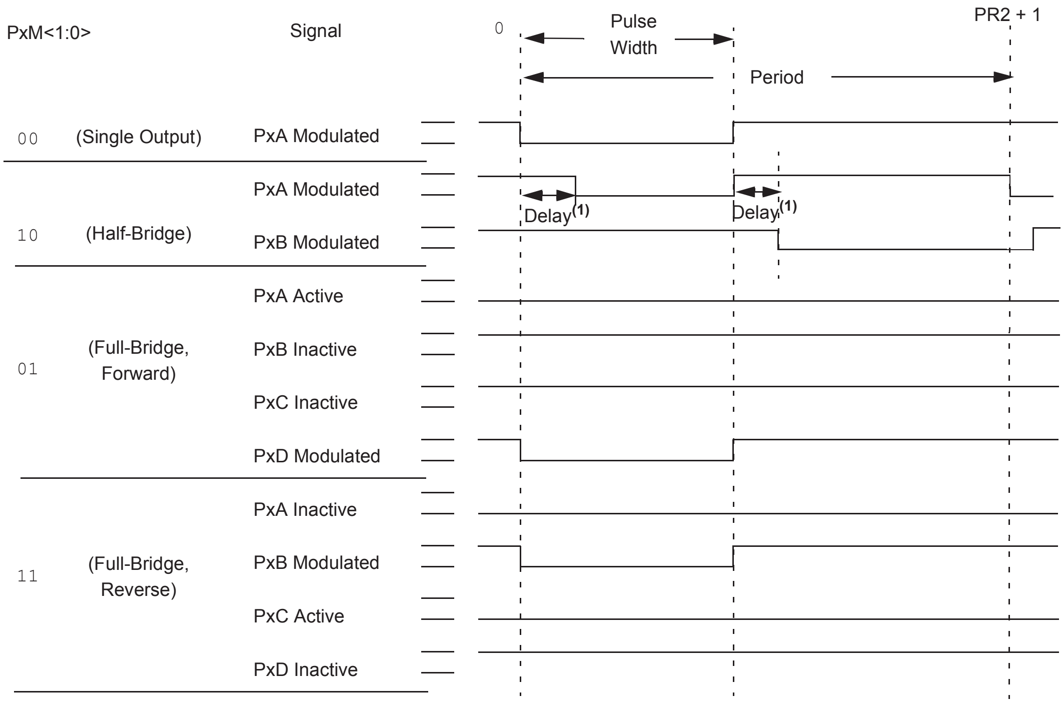

PxM<1:0>
Signal
0
Pulse Width
PR2 + 1
Period
00
(Single Output)
PxA Modulated
Delay(1)
Delay(1)
PxA Modulated
10
(Half-Bridge)
PxB Modulated
PxA Active
PxB Inactive
01
(Full-Bridge, Forward)
PxC Inactive
PxD Modulated
PxA Inactive
11
(Full-Bridge, Reverse)
PxB Modulated
PxC Active
PxD Inactive

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 19.4.6 “Programmable Dead-Band Delay Mode”**).

FIGURE 19-5: ENHANCED PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE) EXAMPLE


PxM<1:0>
Signal
0
Pulse Width
PR2 + 1
Period
00 (Single Output)
PxA Modulated
10 (Half-Bridge)
PxA Modulated
Delay(1)
Delay(1)
PxB Modulated
01 (Full-Bridge, Forward)
PxA Active
PxB Inactive
PxC Inactive
PxD Modulated
11 (Full-Bridge, Reverse)
PxA Inactive
PxB Modulated
PxC Active
PxD Inactive

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 19.4.6 “Programmable Dead-Band Delay Mode”**).

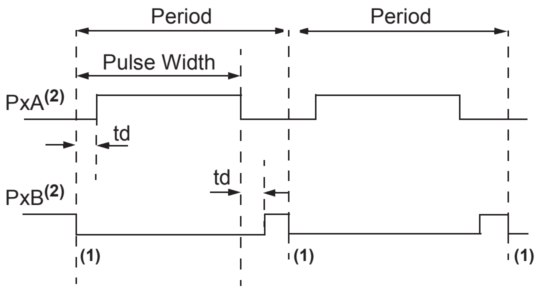
19.4.1 HALF-BRIDGE MODE

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 19-6). This mode can be used for half-bridge applications, as shown in Figure 19-7, 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 19.4.6 "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 19-6: EXAMPLE OF HALF-BRIDGE PWM OUTPUT

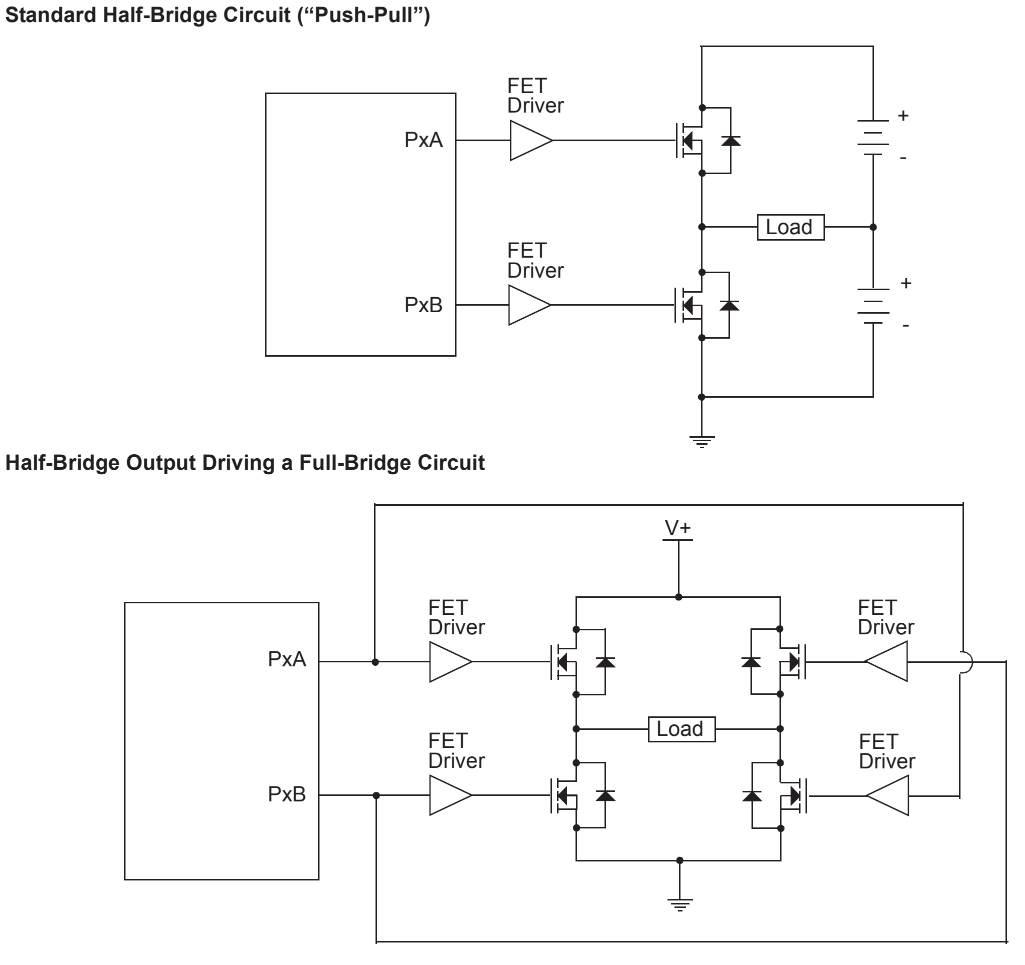


td = Dead-Band Delay

Note 1: At this time, the TMR2 register is equal to the PR2 register.

2: Output signals are shown as active-high.

FIGURE 19-7: EXAMPLE OF HALF-BRIDGE APPLICATIONS


Standard Half-Bridge Circuit ("Push-Pull")
FET Driver
PxA
FET Driver
PxB
+
-
Load
+
-
Half-Bridge Output Driving a Full-Bridge Circuit
V+
FET Driver
FET Driver
PxA
Load
FET Driver
FET Driver
PxB

19.4.2 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 19-8.

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 shown in Figure 19-9.

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 shown in Figure 19-9.

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 19-8: EXAMPLE OF FULL-BRIDGE APPLICATION

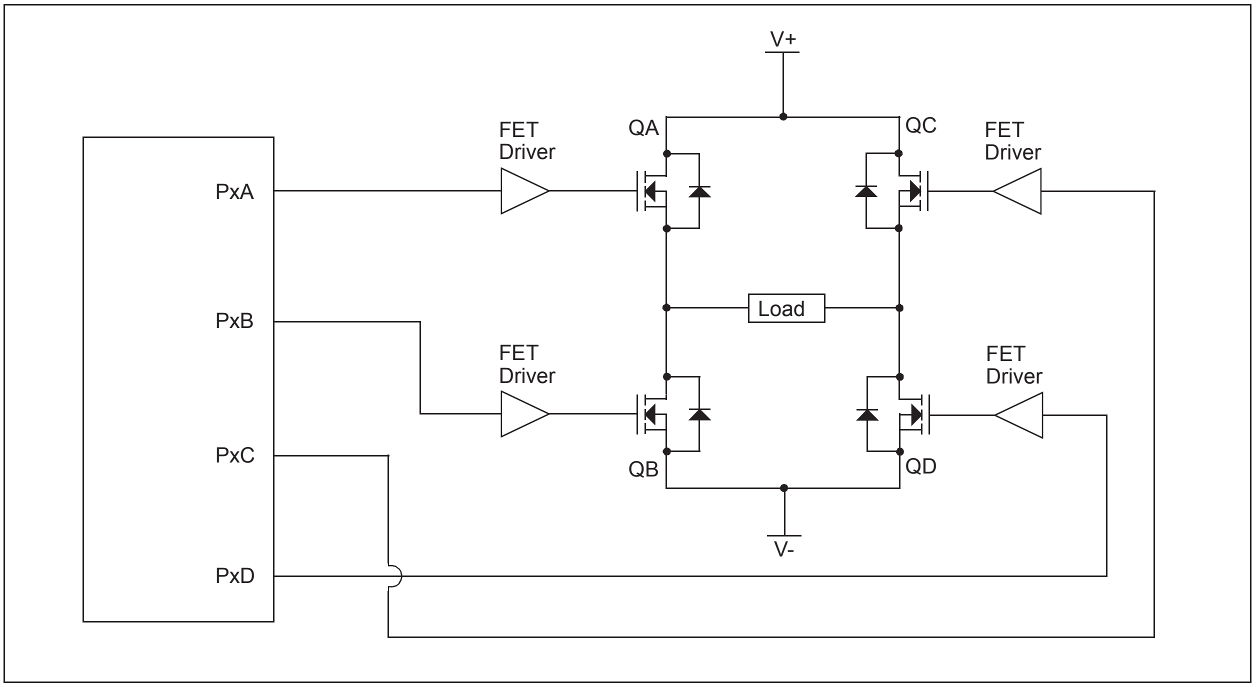
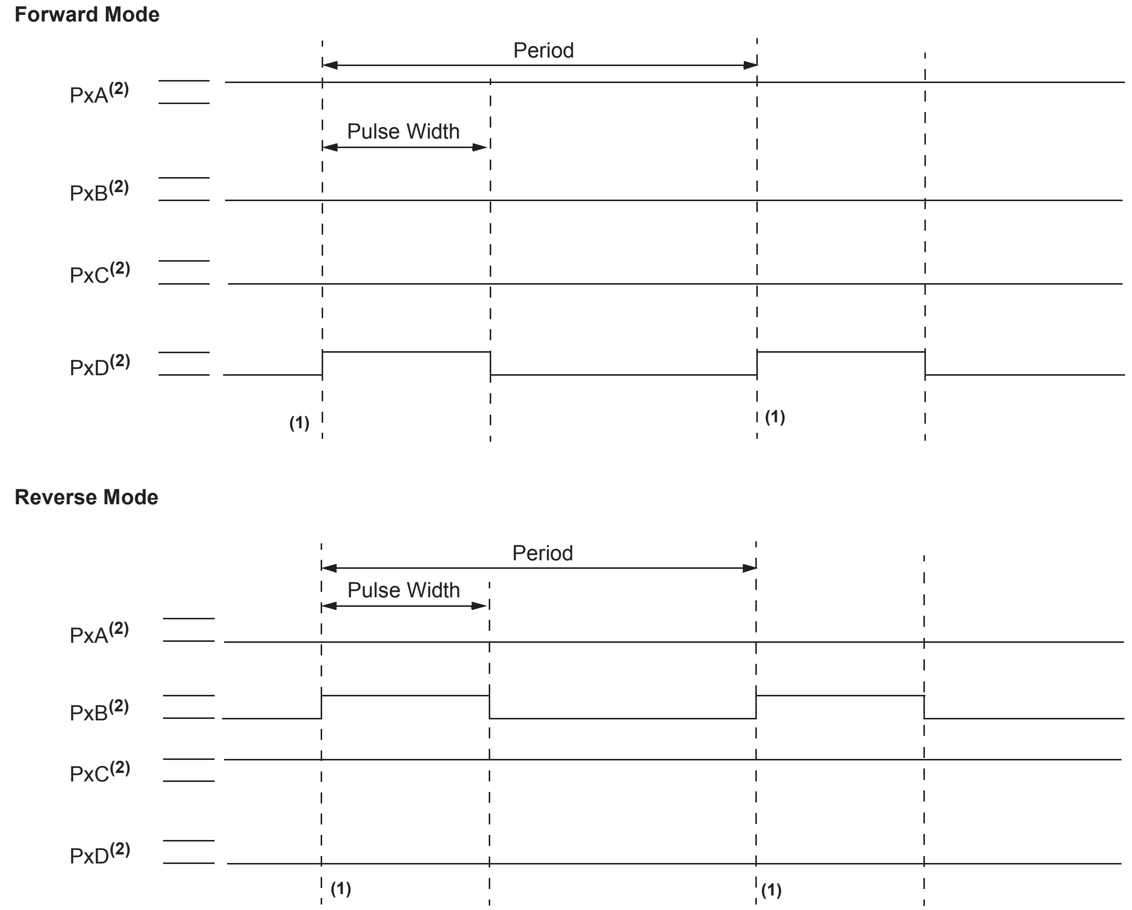


FIGURE 19-9: EXAMPLE OF FULL-BRIDGE PWM OUTPUT


Forward Mode
Period
PxA(2)
Pulse Width
PxB(2)
PxC(2)
PxD(2)
(1)
(1)
Reverse Mode
Period
Pulse Width
PxA(2)
PxB(2)
PxC(2)
PxD(2)
(1)
(1)

Note 1: At this time, the TMR2 register is equal to the PR2 register.

2: The output signal is shown as active-high.

19.4.2.1 Direction Change in Full-Bridge Mode

In the Full-Bridge mode, the PxM1 bit in the CCPxCON register allows users to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will change to the new direction on the next PWM cycle.

A direction change is initiated in software by changing the PxM1 bit of the CCPxCON register. The following sequence occurs 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 19-10.

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.

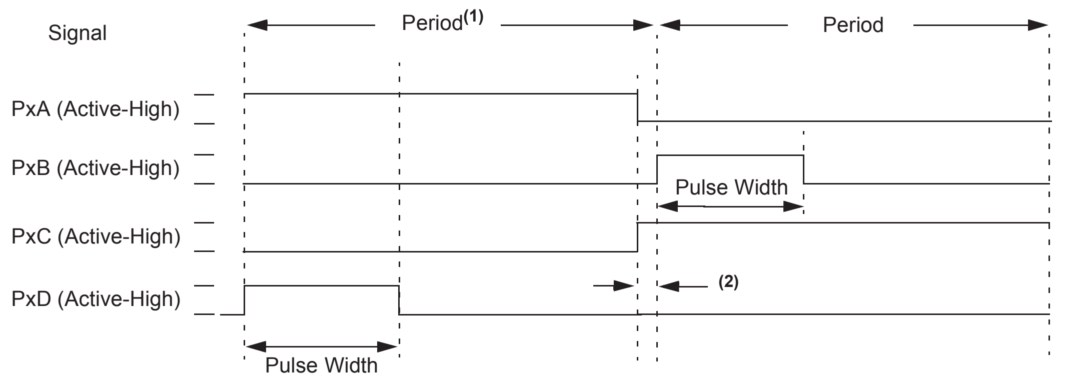
Figure 19-11 shows an example of the PWM direction changing from forward to reverse at a near 100% duty cycle. In this example, at time t1, the PxA and PxD outputs become inactive, while the PxC output becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through power devices, QC and QD (see Figure 19-8), 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.

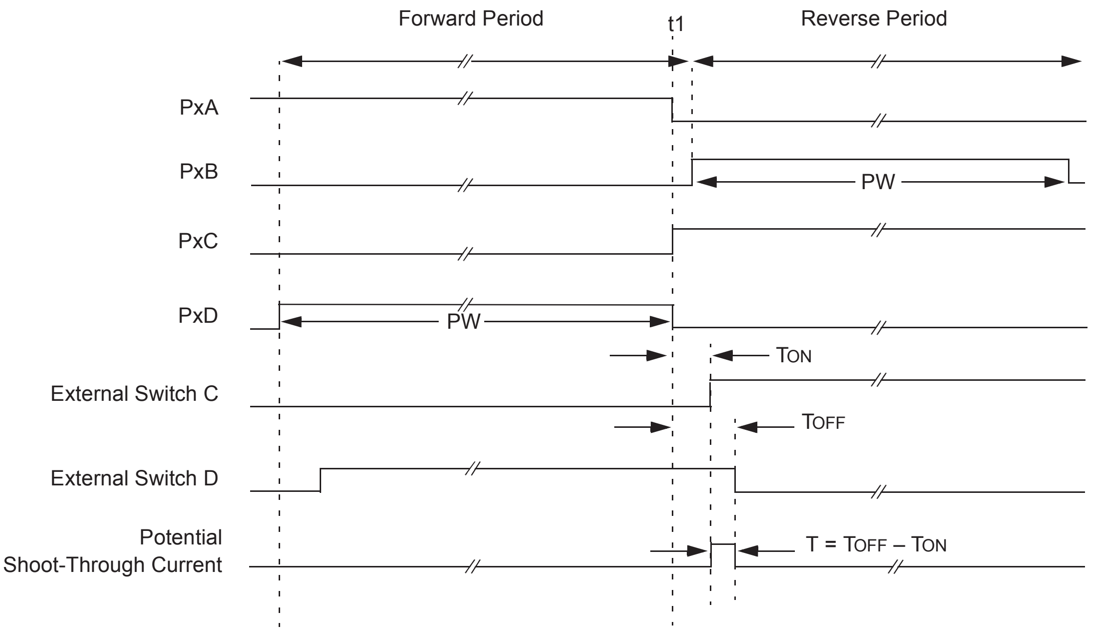
FIGURE 19-10: EXAMPLE OF PWM DIRECTION CHANGE



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.

FIGURE 19-11: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



Note 1: All signals are shown as active-high.

2: TON is the turn-on delay of power switch, QC, and its driver.

3: TOFF is the turn-off delay of power switch, QD, and its driver.

19.4.3 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 PIR3 register being set as the second PWM period begins.

19.4.4 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 to the FLT0 input function
- Comparator C1
- Comparator C2
- 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 19.4.5 "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<3:0>).

Each pin pair may be placed into one of three states:

- Drive logic '`1`'
- Drive logic '`0`'
- Tri-state (high-impedance)

REGISTER 19-4: ECCPxAS: ECCP1/2/3 AUTO-SHUTDOWN CONTROL REGISTER (1, ACCESS FBEh; 2, FB8h; 3, BANKED F19h)

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

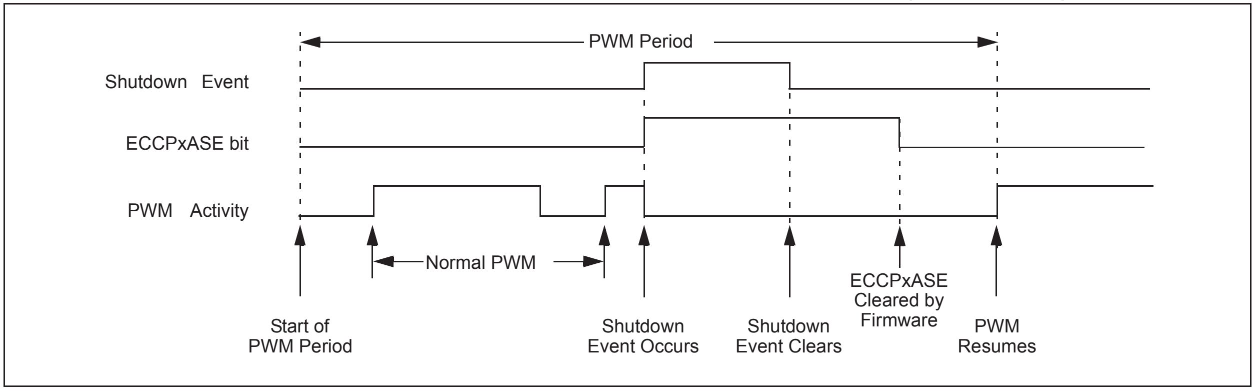
bit 7	**ECCPxASE:** ECCP Auto-Shutdown Event Status bit `1` = A shutdown event has occurred; ECCP outputs are in a shutdown state `0` = ECCP outputs are operating
bit 6-4	**ECCPxAS<2:0>:** ECCP Auto-Shutdown Source Select bits `000` = Auto-shutdown is disabled `001` = Comparator, C1OUT, output is high `010` = Comparator, C2OUT, output is high `011` = Either comparator, C1OUT or C2OUT, is high `100` = V_{IL} on FLT0 pin `101` = V_{IL} on FLT0 pin or comparator, C1OUT, output is high `110` = V_{IL} on FLT0 pin or comparator, C2OUT, output is high `111` = V_{IL} 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>:** PxB and PxD Pins 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.

FIGURE 19-12: PWM AUTO-SHUTDOWN WITH FIRMWARE RESTART (PxRSEN = 0)


PWM Period
Shutdown Event
ECCPxASE bit
PWM Activity
Normal PWM
ECCPxASE Cleared by Firmware
Start of PWM Period
Shutdown Event Occurs
Shutdown Event Clears
PWM Resumes

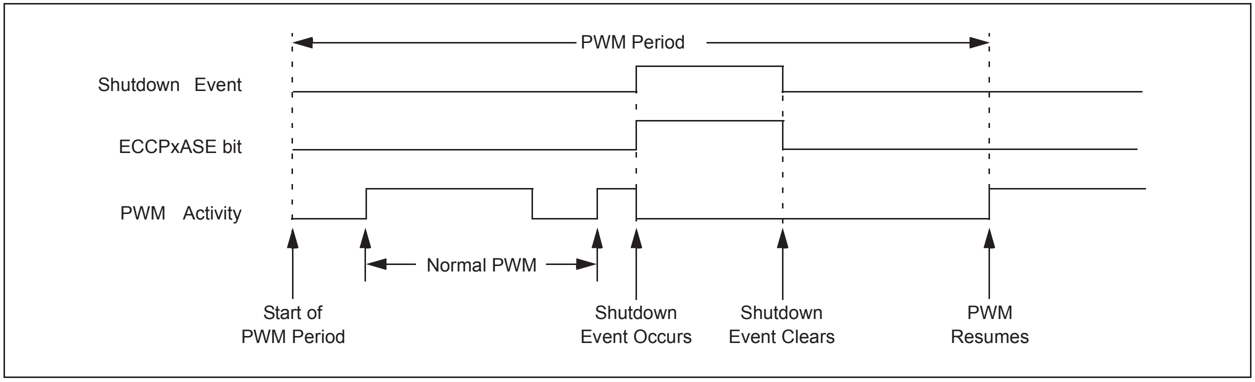
19.4.5 AUTO-RESTART MODE

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 the current mode of PWM control.

FIGURE 19-13: PWM AUTO-SHUTDOWN WITH AUTO-RESTART ENABLED (PxRSEN = 1)

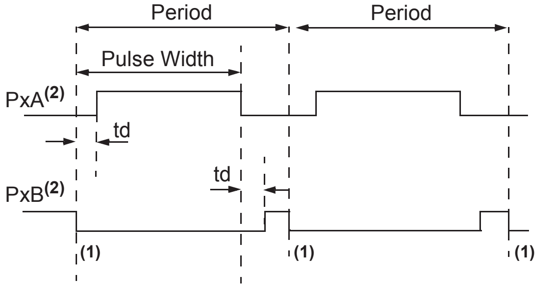

PWM Period
Shutdown Event
ECCPxASE bit
PWM Activity
Normal PWM
Start of PWM Period
Shutdown Event Occurs
Shutdown Event Clears
PWM Resumes

19.4.6 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 19-14. The lower seven bits of the associated ECCPxDEL register (Register 19-5) set the delay period in terms of microcontroller instruction cycles (TCY or 4 TOSC).

FIGURE 19-14: EXAMPLE OF HALF-BRIDGE PWM OUTPUT

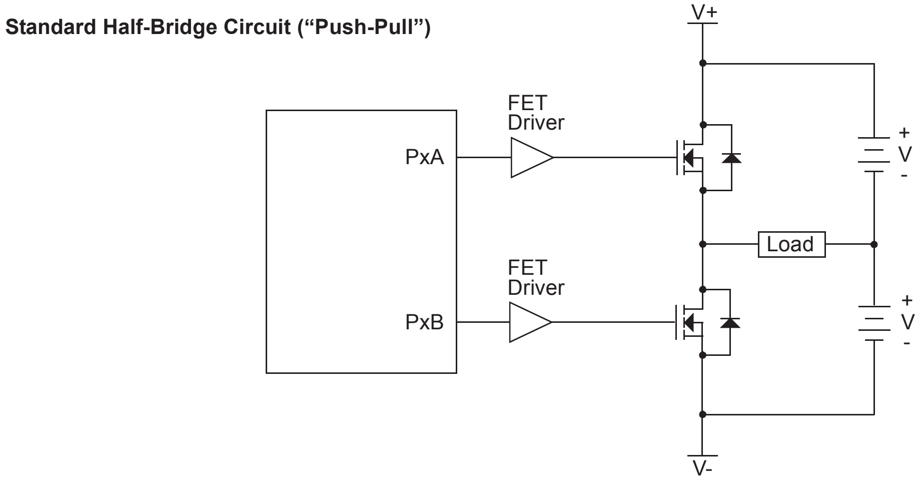


td = Dead-Band Delay

Note 1: At this time, the TMR2 register is equal to the PR2 register.

2: Output signals are shown as active-high.

FIGURE 19-15: EXAMPLE OF HALF-BRIDGE APPLICATIONS



REGISTER 19-5: ECCPxDEL: ECCP1/2/3 ENHANCED PWM CONTROL REGISTER (1, ACCESS FBDh; 2, FB7h; 3, BANKED F18h)

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.

19.4.7 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 19-3.

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 PWM Steering mode, as described in **Section 19.4.4 “Enhanced PWM Auto-Shutdown Mode”**. An auto-shutdown event will only affect pins that have PWM outputs enabled.

REGISTER 19-6: PSTRxCON: PULSE STEERING CONTROL (1, ACCESS FBFh; 2, FB9h; 3, BANKED F1Ah)[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

`1` = Modulated output pin toggles between PxA and PxB for each period
`0` = Complementary output assignment is disabled; the STRD:STRA bits are used to determine Steering mode

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 D bit

`1` = PxD pin has the PWM waveform with polarity control from CCPxM<1:0>
`0` = PxD pin is assigned to a port pin

bit 2 **STRC:** Steering Enable C bit

`1` = PxC pin has the PWM waveform with polarity control from CCPxM<1:0>
`0` = PxC pin is assigned to a port pin

bit 1 **STRB:** Steering Enable B bit

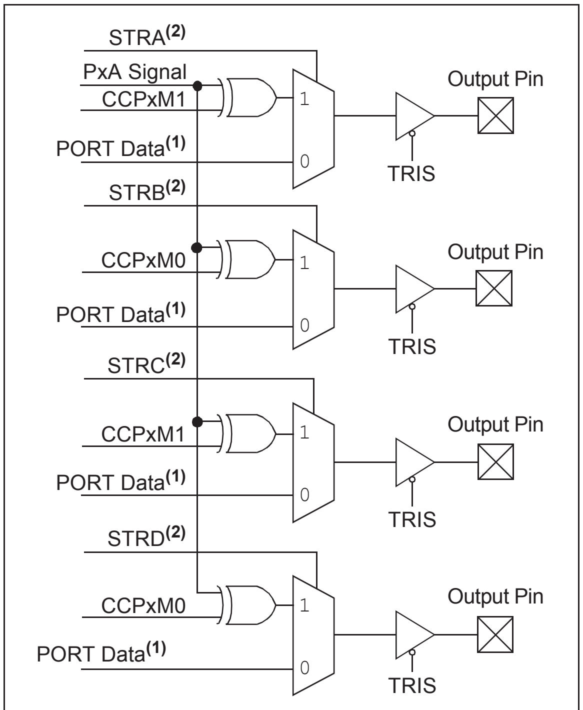
`1` = PxB pin has the PWM waveform with polarity control from CCPxM<1:0>
`0` = PxB pin is assigned to a port pin

bit 0 **STRA:** Steering Enable A bit

`1` = PxA pin has the PWM waveform with polarity control from CCPxM<1:0>
`0` = PxA pin is assigned to a port pin

Note 1: The PWM Steering mode is available only when the CCPxCON register bits, CCPxM<3:2> = `11` and PxM<1:0> = `00`.

FIGURE 19-16: SIMPLIFIED STEERING BLOCK DIAGRAM


STRA(2)
PxA Signal
CCPxM1
PORT Data(1)
STRB(2)
CCPxM0
PORT Data(1)
STRC(2)
CCPxM1
PORT Data(1)
STRD(2)
CCPxM0
PORT Data(1)
1
0
Output Pin
TRIS

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 STR<D:A> bits.

19.4.7.1 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.

Figure 19-17 and Figure 19-18 illustrate the timing diagrams of the PWM steering depending on the STRSYNC setting.

FIGURE 19-17: EXAMPLE OF STEERING EVENT AT END OF INSTRUCTION (STRSYNC = `0`)

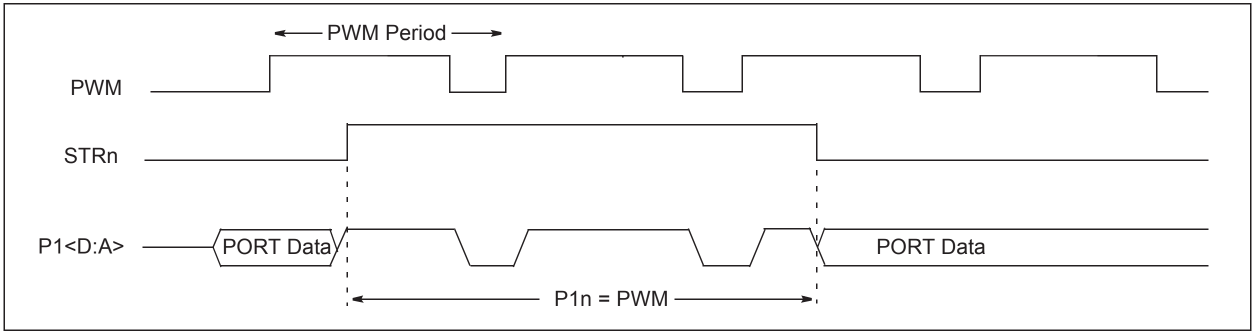
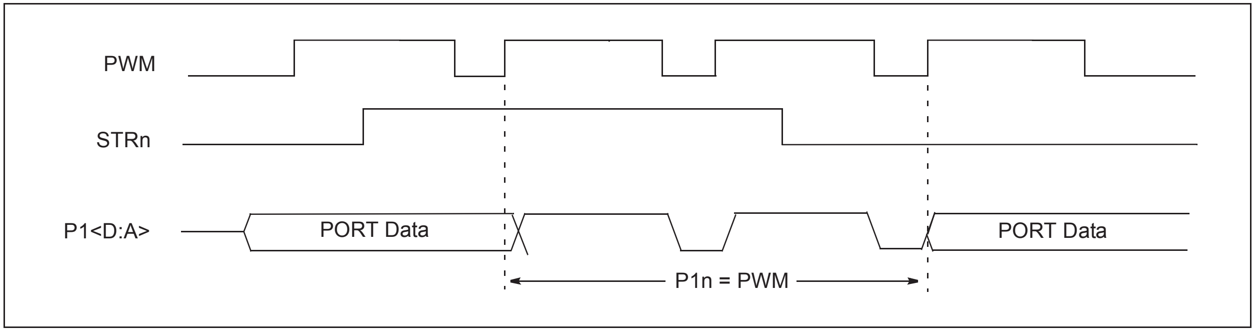

PWM Period
PWM
STRn
P1<D:A>
PORT Data
PORT Data
P1n = PWM

FIGURE 19-18: EXAMPLE OF STEERING EVENT AT BEGINNING OF INSTRUCTION (STRSYNC = `1`)


PWM
STRn
P1<D:A>
PORT Data
PORT Data
P1n = PWM

19.4.8 OPERATION IN POWER-MANAGED MODES

In Sleep mode, all clock sources are disabled. Timer2 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 HFINTOSC and the postscaler may not be immediately stable.

In PRI_IDLE mode, the primary clock will continue to clock the ECCPx module without change.

19.4.8.1 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.

19.4.9 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 19-4: REGISTERS ASSOCIATED WITH ECCP1/2/3 MODULE AND TIMER1/2/3/4/6/8

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	—	$\overline{\text{CM}}$	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
PIR1	PMPIF	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIR2	OSCFIF	CM2IF	CM1IF	—	BCL1IF	HLVDIF	TMR3IF	CCP2IF
PIR4	CCP10IF	CCP9IF	CCP8IF	CCP7IF	CCP6IF	CCP5IF	CCP4IF	CCP3IF
PIE1	PMPIE	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
PIE2	OSCFIE	CM2IE	CM1IE	—	BCL1IE	HLVDIE	TMR3IE	CCP2IE
PIE4	CCP10IE	CCP9IE	CCP8IE	CCP7IE	CCP6IE	CCP5IE	CCP4IE	CCP3IE
IPR1	PMPIP	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
IPR2	OSCFIP	CM2IP	CM1IP	—	BCL1IP	HLVDIP	TMR3IP	CCP2IP
IPR4	CCP10IP	CCP9IP	CCP8IP	CCP7IP	CCP6IP	CCP5IP	CCP4IP	CCP3IP
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	TRISC7	TRISC6	—	—	—	TRISC2	TRISC1	TRISC0
TRISE	$\overline{\text{RDPU}}$	$\overline{\text{REPU}}$	—	—	—	TRISE2	TRISE1	TRISE0
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							
PR2	Timer2 Period Register							
PR4	Timer4 Period Register							
PR6	Timer6 Period Register							
PR8	Timer8 Period Register							
T1CON	TMR1CS1	TMR1CS0	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	RD16	TMR1ON
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
T3CON	TMR3CS1	TMR3CS0	T3CKPS1	T3CKPS0	T3OSCEN	$\overline{\text{T3SYNC}}$	RD16	TMR3ON
T4CON	—	T4OUTPS3	T4OUTPS2	T4OUTPS1	T4OUTPS0	TMR4ON	T4CKPS1	T4CKPS0
T6CON	—	T6OUTPS3	T6OUTPS2	T6OUTPS1	T6OUTPS0	TMR6ON	T6CKPS1	T6CKPS0

TABLE 19-4: REGISTERS ASSOCIATED WITH ECCP1/2/3 MODULE AND TIMER1/2/3/4/6/8 (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T8CON	—	T8OUTPS3	T8OUTPS2	T8OUTPS1	T8OUTPS0	TMR8ON	T8CKPS1	T8CKPS0
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	P3M1	P3M0	DC3B1	DC3B0	CCP3M3	CCP3M2	CCP3M1	CCP3M0

20.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices include serial EEPROMs, shift registers, display drivers and A/D Converters.

20.1 Master SSP (MSSP) Module Overview

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 PIC18F47J13 Family have two MSSP modules, designated as MSSP1 and MSSP2. The modules operate independently:

- PIC18F4XJ13 devices – Both modules can be configured for either I^2C or SPI communication
- PIC18F2XJ13 devices:
  - MSSP1 can be used for either I^2C or SPI communication
  - MSSP2 can be used only for SPI communication

All of the MSSP1 module related SPI and I^2C I/O functions are hard-mapped to specific I/O pins.

For MSSP2 functions:

- SPI I/O functions (SDO2, SDI2, SCK2 and $\overline{\text{SS2}}$) are all routed through the Peripheral Pin Select (PPS) module.

  These functions may be configured to use any of the RPn remappable pins, as described in **Section 10.7 "Peripheral Pin Select (PPS)"**.
- I^2C functions (SCL2 and SDA2) have fixed pin locations.

On all PIC18F47J13 Family devices, the SPI DMA capability can only be used in conjunction with MSSP2. The SPI DMA feature is described in **Section 20.4 "SPI DMA Module"**.

Note:	Throughout this section, generic references to an MSSP module in any of its operating modes may be interpreted as being equally applicable to MSSP1 or MSSP2. Register names and module I/O signals use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module when required. Control bit names are not individuated.

20.2 Control Registers

Each MSSP module has three associated control registers. These include a status register (SSPxSTAT) and two control registers (SSPxCON1 and SSPxCON2). The use of these registers and their individual Configuration bits differs significantly depending on whether the MSSP module is operated in SPI or I^{2}C mode.

Additional details are provided under the individual sections.

Note: In devices with more than one MSSP module, it is very important to pay close attention to the SSPxCON register names. SSP1CON1 and SSP1CON2 control different operational aspects of the same module, while SSP1CON1 and SSP2CON1 control the same features for two different modules.

20.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported.

When MSSP2 is used in SPI mode, it can optionally be configured to work with the SPI DMA submodule described in **Section 20.4 "SPI DMA Module"**.

To accomplish communication, typically three pins are used:

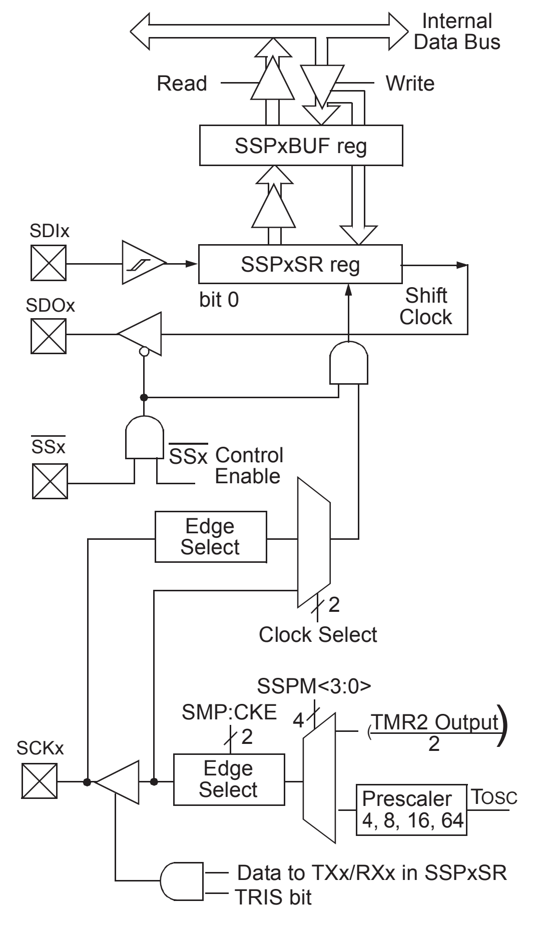
- Serial Data Out (SDOx) – RC5/SDO1/RP16 or SDO2/Remappable
- Serial Data In (SDIx) – RC4/SDI1/SDA1/RP15 or SDI2/Remappable
- Serial Clock (SCKx) – RC3/SCK1/SCL1/RP14 or SCK2/Remappable

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{\text{SSx}}$) – RA5/AN4/C1INC/$\overline{\text{SS1}}$/HLVDIN/RP2 or $\overline{\text{SS2}}$/Remappable

Figure 20-1 depicts the block diagram of the MSSP module when operating in SPI mode.

FIGURE 20-1: MSSPx BLOCK DIAGRAM (SPI MODE)


Internal
Data Bus
Read
Write
SSPxBUF reg
SDIx
SSPxSR reg
bit 0
Shift
Clock
SDOx
SSx
SSx Control
Enable
Edge
Select
2
Clock Select
SSPM<3:0>
SMP:CKE
4
2
TMR2 Output
2
SCKx
Edge
Select
Prescaler
4, 8, 16, 64
TOSC
Data to TXx/RXx in SSPxSR
TRIS bit

Note: Only port I/O names are used in this diagram for the sake of brevity. Refer to the text for a full list of multiplexed functions.

20.3.1 REGISTERS

Each MSSP module has four registers for SPI mode operation. These are:

- MSSPx Control Register 1 (SSPxCON1)
- MSSPx Status Register (SSPxSTAT)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSPx Shift Register (SSPxSR) – Not directly accessible

SSPxCON1 and SSPxSTAT are the control and status registers in SPI mode operation. The SSPxCON1 register is readable and writable. The lower six bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

SSPxSR is the shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPxSR and SSPxBUF together, create a double-buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

Note:	Because the SSPxBUF register is double-buffered, using read-modify-write instructions such as `BCF`, `COMF`, etc., will not work. Similarly, when debugging under an in-circuit debugger, performing actions that cause reads of SSPxBUF (mouse hovering, watch, etc.) can consume data that the application code was expecting to receive.

REGISTER 20-1: SSPxSTAT: MSSPx STATUS REGISTER (SPI MODE) (ACCESS 1, FC7h; 2, F73h)

R/W-1	R/W-1	R-1	R-1	R-1	R-1	R-1	R-1
SMP	CKE[(1)]	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	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 sampled at the end of data output time
`0` = Input data sampled at the middle of data output time

SPI Slave mode:
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Select bit[(1)]

`1` = Transmit occurs on transition from active to Idle clock state
`0` = Transmit occurs on transition from Idle to active clock state

bit 5 **D/$\overline{\text{A}}$:** Data/$\overline{\text{Address}}$ bit

Used in I^{2}C mode only.

bit 4 **P:** Stop bit

Used in I^{2}C mode only; this bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** Start bit

Used in I^{2}C mode only.

bit 2 **R/$\overline{\text{W}}$:** Read/$\overline{\text{Write}}$ Information bit

Used in I^{2}C mode only.

bit 1 **UA:** Update Address bit

Used in I^{2}C mode only.

bit 0 **BF:** Buffer Full Status bit

`1` = Receive complete, SSPxBUF is full
`0` = Receive not complete, SSPxBUF is empty

Note 1: Polarity of the clock state is set by the CKP bit (SSPxCON1<4>).

REGISTER 20-2: SSPxCON1: MSSPx CONTROL REGISTER 1 (SPI MODE) (1, ACCESS FC6h; 2, F72h)

R/W-0	R/C-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:
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 **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 $\overline{\text{SSx}}$ as serial port pins
`0` = Disables serial port and configures these pins as I/O port pins

bit 4 **CKP:** Clock Polarity Select bit

`1` = Idle state for clock is a high level
`0` = Idle state for clock is a low level

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits(3)

`0101` = SPI Slave mode, clock = SCKx pin; $\overline{\text{SSx}}$ pin control disabled, $\overline{\text{SSx}}$ can be used as I/O pin
`0100` = SPI Slave mode, clock = SCKx pin; $\overline{\text{SSx}}$ 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
`1010` = SPI Master mode, clock = FOSC/8
`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, this pin must be properly configured as input or output.
3: Bit combinations not specifically listed here are either reserved or implemented in I^{2}C mode only.

20.3.2 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPxCON1<5:0> and SSPxSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCKx is the clock output)
- Slave mode (SCKx is the clock input)
- Clock Polarity (Idle state of SCKx)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCKx)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

Each MSSP module consists of a Transmit/Receive Shift register (SSPxSR) and a Buffer register (SSPxBUF). The SSPxSR shifts the data in and out of the device, MSb first. The SSPxBUF holds the data that was written to the SSPxSR until the received data is ready. Once the 8 bits of data have been received, that byte is moved to the SSPxBUF register. Then, the Buffer Full (BF) detect bit (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 or 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.

Note: When the application software is expecting to receive valid data, the SSPxBUF should be read before the next byte of transfer data is written to the SSPxBUF. Application software should follow this process even when the current contents of SSPxBUF are not important.

The Buffer Full bit, BF (SSPxSTAT<0>), indicates when SSPxBUF has been loaded with the received data (transmission is complete). When the SSPxBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur.

Example 20-1 provides the loading of the SSPxBUF (SSPxSR) for data transmission.

The SSPxSR is not directly readable or writable and can only be accessed by addressing the SSPxBUF register. Additionally, the SSPxSTAT register indicates the various status conditions.

20.3.3 OPEN-DRAIN OUTPUT OPTION

The drivers for the SDOx output and SCKx clock pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor, provided the SDOx or SCKx pin is not multiplexed with an ANx analog function. This allows the output to communicate with external circuits without the need for additional level shifters. For more information, see **Section 10.1.4 "Open-Drain Outputs"**.

The open-drain output option is controlled by the SPI2OD and SPI1OD bits (ODCON3<1:0>). Setting an SPIxOD bit configures both the SDOx and SCKx pins for the corresponding open-drain operation.

EXAMPLE 20-1: LOADING THE SSP1BUF (SSP1SR) REGISTER

```
LOOP    BTFSS   SSP1STAT, BF    ;Has data been received (transmit complete)?
        BRA     LOOP            ;No
        MOVF    SSP1BUF, W      ;WREG reg = contents of SSP1BUF

        MOVWF   RXDATA          ;Save in user RAM, if data is meaningful

        MOVF    TXDATA, W       ;W reg = contents of TXDATA
        MOVWF   SSP1BUF         ;New data to xmit
```

20.3.4 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPxCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPxCON1 registers and then set the SSPEN bit. This configures the SDIx, SDOx, SCKx and $\overline{\text{SSx}}$ pins as serial port pins. For the pins to behave as the serial port function, the appropriate TRISx bits, PCFGx bits and Peripheral Pin Select registers (if using MSSP2) should be correctly initialized prior to setting the SSPEN bit.

A typical SPI serial port initialization process follows:

- Initialize the ODCON3 register (optional open-drain output control)
- Initialize the remappable pin functions (if using MSSP2, see **Section 10.7 “Peripheral Pin Select (PPS)”**)
- Initialize the SCKx/LAT value to the desired Idle SCKx level (if master device)
- Initialize the SCKx/PCFGx bit (if in Slave mode and multiplexed with the ANx function)
- Initialize the SCKx/TRISx bit as an output (Master mode) or input (Slave mode)
- Initialize the SDIx/PCFGx bit (if SDIx is multiplexed with the ANx function)
- Initialize the SDIx/TRISx bit
- Initialize the $\overline{\text{SSx}}$/PCFG bit (if in Slave mode and multiplexed with the ANx function)
- Initialize the $\overline{\text{SSx}}$/TRISx bit (Slave modes)
- Initialize the SDOx/TRISx bit
- Initialize the SSPxSTAT register
- Initialize the SSPxCON1 register
- Set the SSPEN bit to enable the module

Any MSSP1 serial port function that is not desired may be overridden by programming the corresponding Data Direction (TRIS) register to the opposite value. If individual MSSP2 serial port functions will not be used, they may be left unmapped.

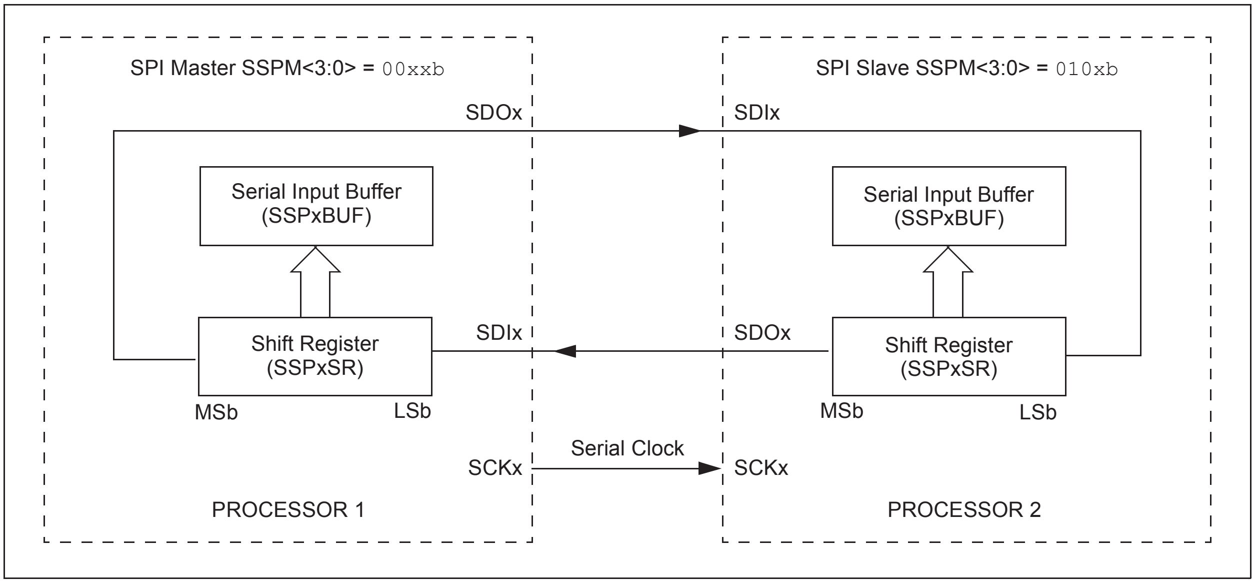
Note: When MSSP2 is used in SPI Master mode, the SCK2 function must be configured as both an output and an input in the PPS module. SCK2 must be initialized as an output pin (by writing 0x0A to one of the RPORx registers). Additionally, SCK2IN must also be mapped to the same pin by initializing the RPINR22 register. Failure to initialize SCK2/SCK2IN as both output and input will prevent the module from receiving data on the SDI2 pin, as the module uses the SCK2IN signal to latch the received data.

20.3.5 TYPICAL CONNECTION

Figure 20-2 illustrates 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 valid data – Slave sends dummy data
- Master sends valid data – Slave sends valid data
- Master sends dummy data – Slave sends valid data

FIGURE 20-2: SPI MASTER/SLAVE CONNECTION



20.3.6 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCKx. The master determines when the slave (Processor 2, Figure 20-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPxBUF register is written to. If the SPI is only going to receive, the SDOx output could be disabled (programmed as an input). The SSPxSR register will continue to shift in the signal present on the SDIx pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPxBUF register as if it is a normal received byte (interrupts and status bits are appropriately set). This could be useful in receiver applications as a “Line Activity Monitor” mode.

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.

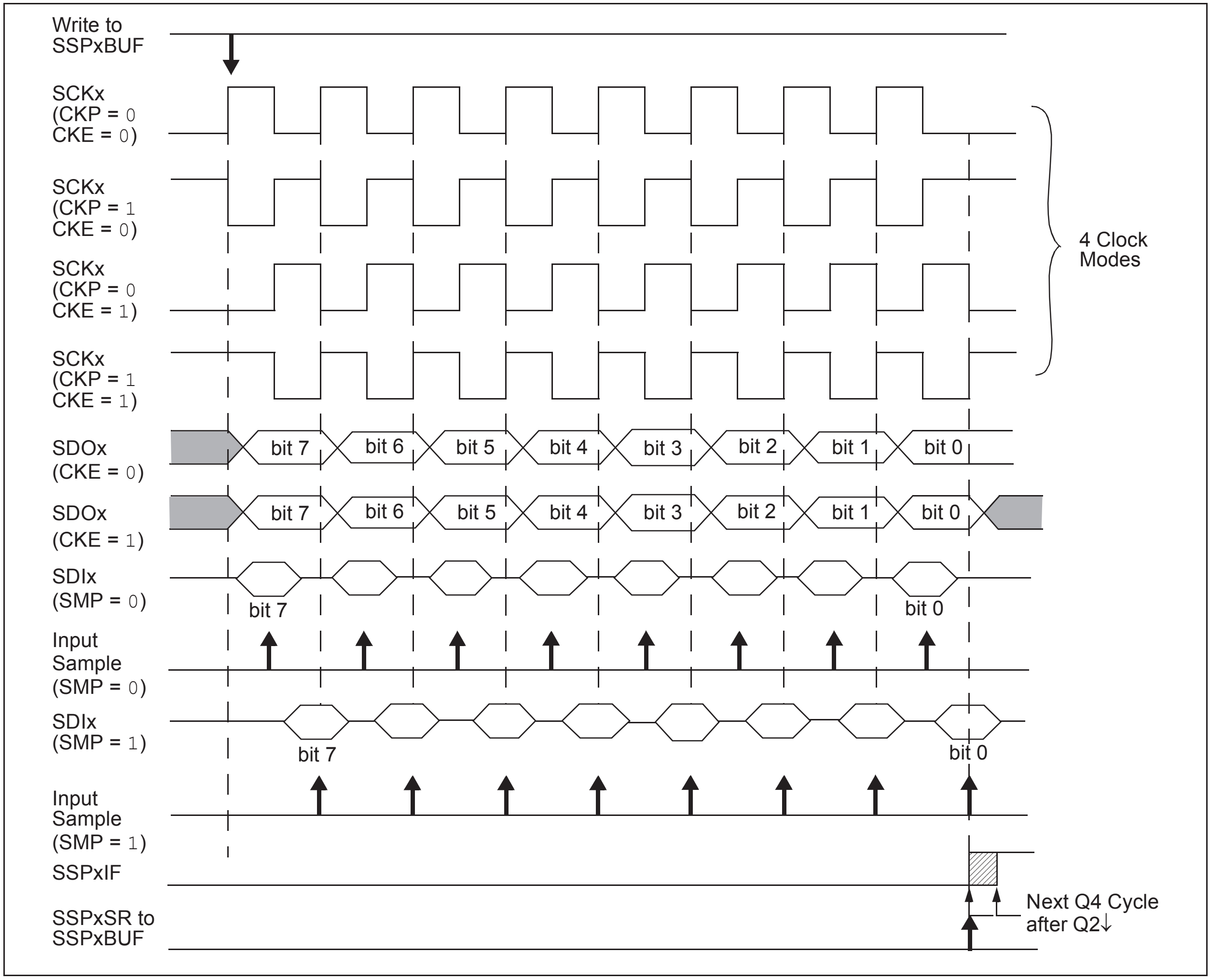
The CKP is selected by appropriately programming the CKP bit (SSPxCON1<4>). This then, would give waveforms for SPI communication, as illustrated in Figure 20-3, Figure 20-5 and Figure 20-6, where the Most Significant Byte (MSB) is transmitted first. In Master mode, the SPI clock rate (bit rate) is user-programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/8 (or 2 • TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

When using the Timer2 output/2 option, the Period Register 2 (PR2) can be used to determine the SPI bit rate. However, only PR2 values of 0x01 to 0xFF are valid in this mode.

Figure 20-3 illustrates the waveforms for Master mode. When the CKE bit is set, the SDOx data is valid before there is a clock edge on SCKx. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPxBUF is loaded with the received data is shown.

FIGURE 20-3: SPI MODE WAVEFORM (MASTER MODE)


Write to SSPxBUF
SCKx (CKP = 0 CKE = 0)
SCKx (CKP = 1 CKE = 0)
SCKx (CKP = 0 CKE = 1)
SCKx (CKP = 1 CKE = 1)
4 Clock Modes
SDOx (CKE = 0)
SDOx (CKE = 1)
bit 7
bit 6
bit 5
bit 4
bit 3
bit 2
bit 1
bit 0
SDIx (SMP = 0)
Input Sample (SMP = 0)
SDIx (SMP = 1)
Input Sample (SMP = 1)
SSPxIF
SSPxSR to SSPxBUF
Next Q4 Cycle after Q2↓

20.3.7 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCKx. When the last bit is latched, the SSPxIF interrupt flag bit is set.

While in Slave mode, the external clock is supplied by the external clock source on the SCKx pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. When a byte is received, the device can be configured to wake-up from Sleep.

20.3.8 SLAVE SELECT SYNCHRONIZATION

The $\overline{\text{SSx}}$ pin allows a Synchronous Slave mode. The SPI must be in Slave mode with the $\overline{\text{SSx}}$ pin control enabled (SSPxCON1<3:0> = 04h). When the $\overline{\text{SSx}}$ pin is low, transmission and reception are enabled and the SDOx pin is driven. When the $\overline{\text{SSx}}$ pin goes high, the SDOx pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

Note 1: When the SPI is in Slave mode with the $\overline{\text{SSx}}$ pin control enabled (SSPxCON1<3:0> = `0100`), the SPI module will reset if the $\overline{\text{SSx}}$ pin is set to VDD.

2: If the SPI is used in Slave mode with CKE set, then the $\overline{\text{SSx}}$ 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 $\overline{\text{SSx}}$ 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 20-4: SLAVE SYNCHRONIZATION WAVEFORM

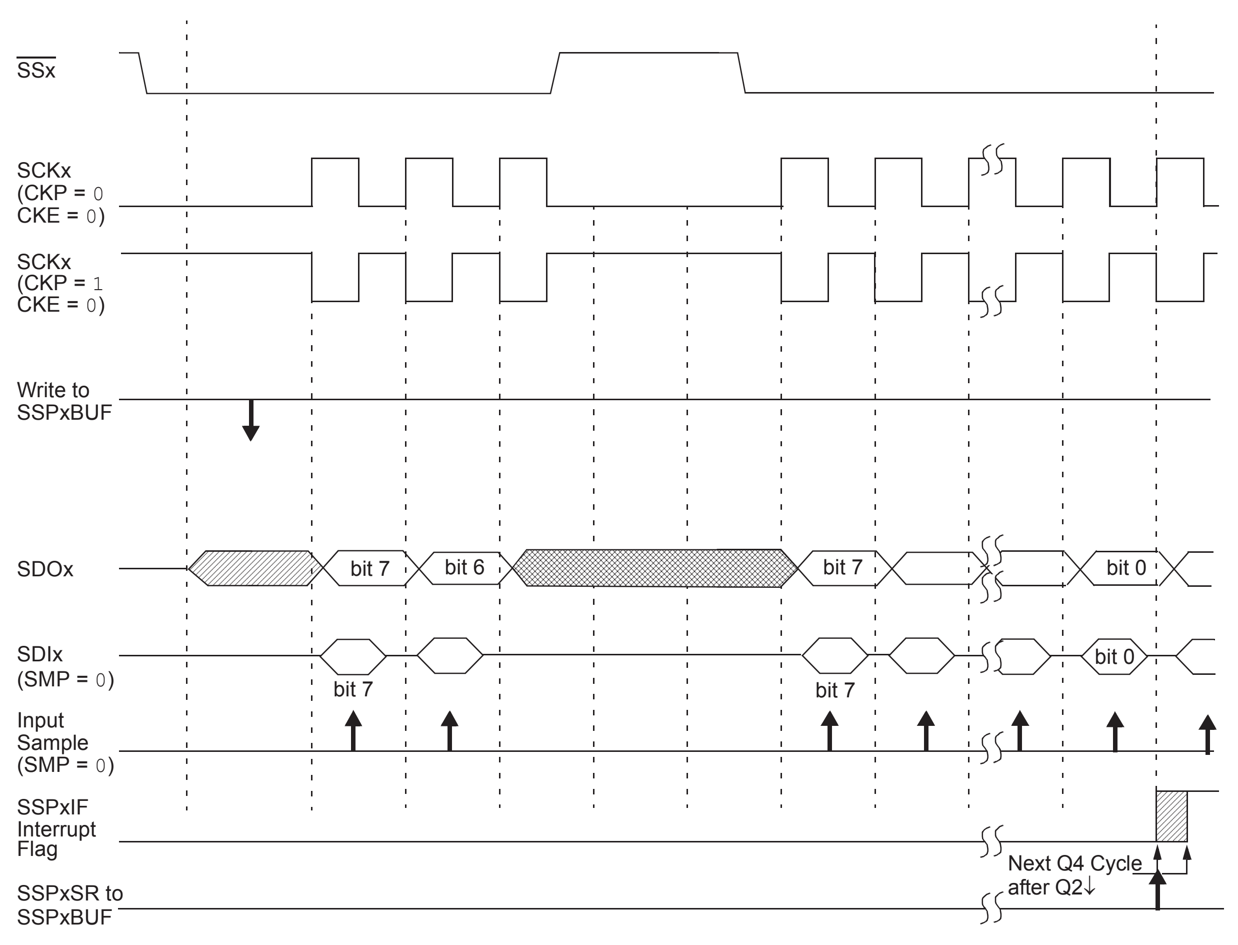


FIGURE 20-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = `0`)

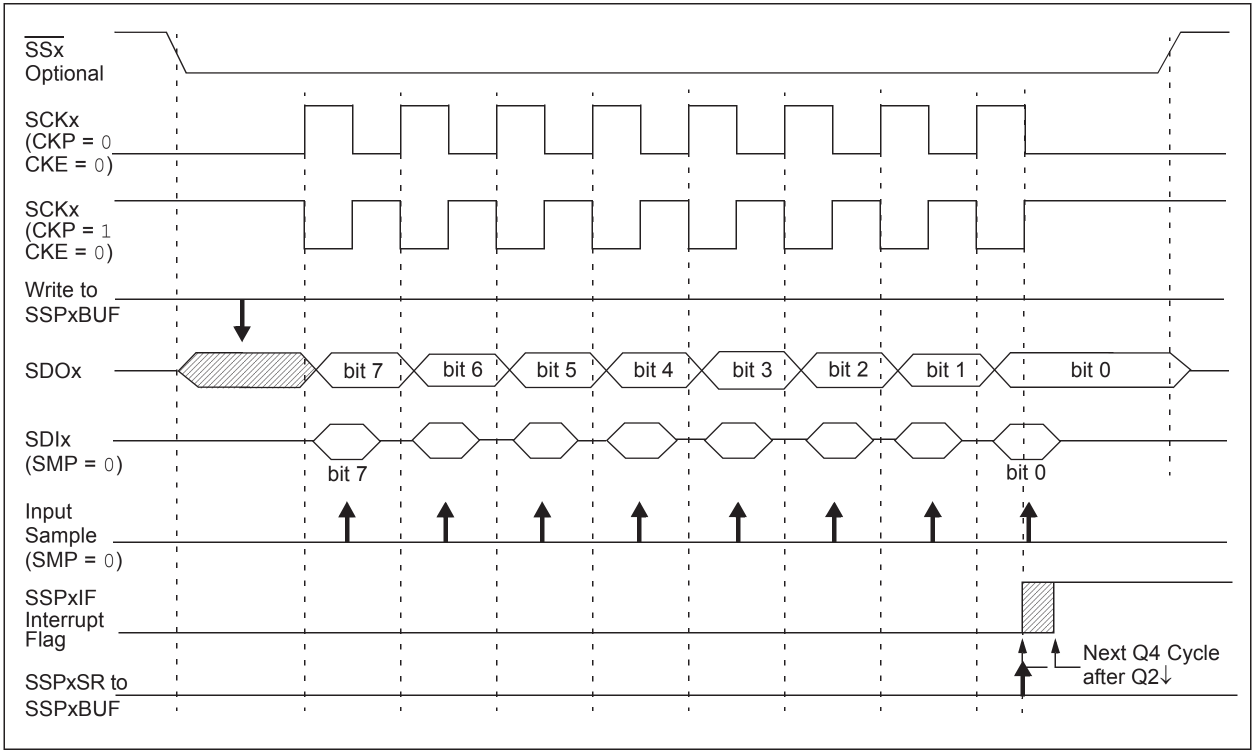
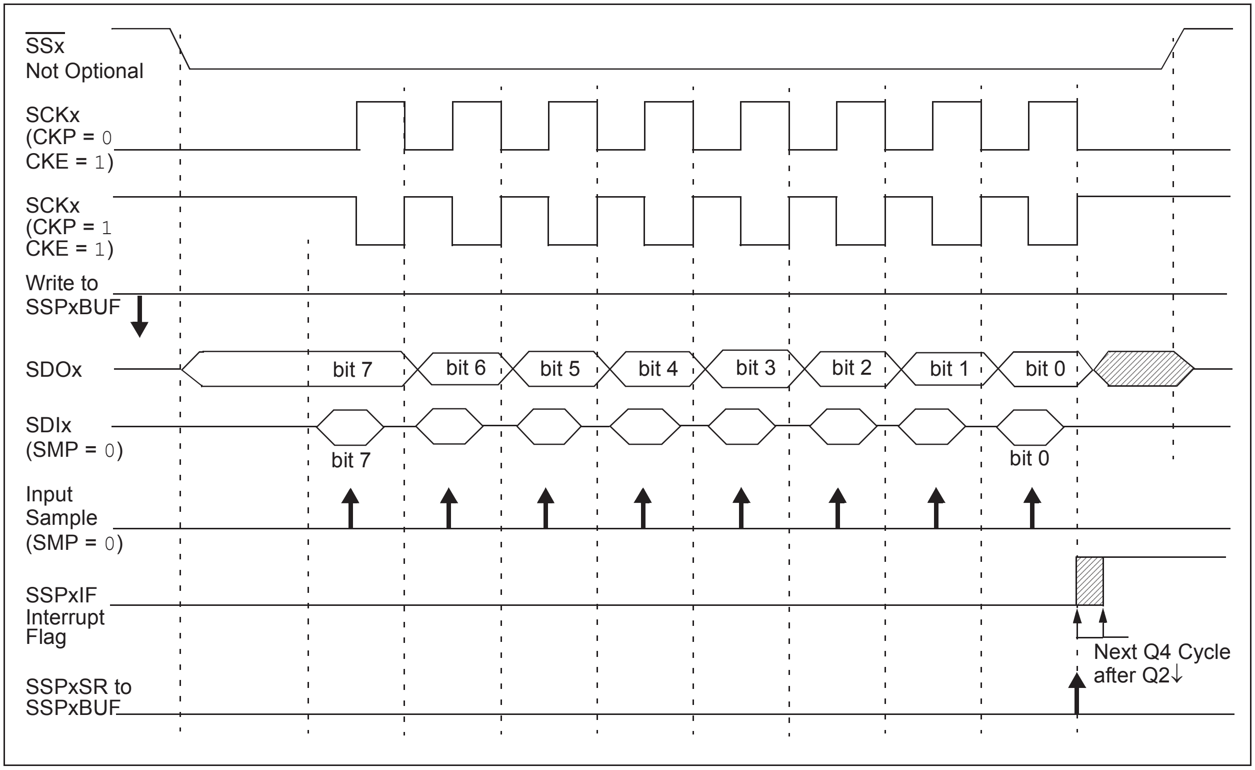

SSx Optional
SCKx (CKP = 0 CKE = 0)
SCKx (CKP = 1 CKE = 0)
Write to SSPxBUF
SDOx
bit 7
bit 6
bit 5
bit 4
bit 3
bit 2
bit 1
bit 0
SDIx (SMP = 0)
bit 7
bit 0
Input Sample (SMP = 0)
SSPxIF Interrupt Flag
SSPxSR to SSPxBUF
Next Q4 Cycle after Q2↓

FIGURE 20-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = `1`)


SSx Not Optional
SCKx (CKP = 0 CKE = 1)
SCKx (CKP = 1 CKE = 1)
Write to SSPxBUF
SDOx
bit 7
bit 6
bit 5
bit 4
bit 3
bit 2
bit 1
bit 0
SDIx (SMP = 0)
bit 7
bit 0
Input Sample (SMP = 0)
SSPxIF Interrupt Flag
SSPxSR to SSPxBUF
Next Q4 Cycle after Q2↓

20.3.9 OPERATION IN POWER-MANAGED MODES

In SPI Master mode, module clocks may be operating at a different speed than when in Full-Power mode. In the case of Sleep mode, all clocks are Halted.

In Idle modes, a clock is provided to the peripherals. That clock can be from the primary clock source, the secondary clock (Timer1 oscillator) or the INTOSC source. See **Section 3.3 "Clock Sources and Oscillator Switching"** for additional information.

In most cases, the speed that the master clocks SPI data is not important; however, this should be evaluated for each system.

If MSSP interrupts are enabled, they can wake the controller from Sleep mode, or one of the Idle modes, when the master completes sending data. If an exit from Sleep or Idle mode is not desired, MSSP interrupts should be disabled.

If the Sleep mode is selected, all module clocks are Halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in any power-managed mode and data to be shifted into the SPI Transmit/Receive Shift register. When all 8 bits have been received, the MSSPx interrupt flag bit will be set, and if enabled, will wake the device.

20.3.10 EFFECTS OF A RESET

A Reset disables the MSSP modules and terminates the current transfer.

20.3.11 BUS MODE COMPATIBILITY

Table 20-1 provides the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

TABLE 20-1: SPI BUS MODES

Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
`0, 0`	`0`	`1`
`0, 1`	`0`	`0`
`1, 0`	`1`	`1`
`1, 1`	`1`	`0`

There is also an SMP bit, which controls when the data is sampled.

20.3.12 SPI CLOCK SPEED AND MODULE INTERACTIONS

Because MSSP1 and MSSP2 are independent modules, they can operate simultaneously at different data rates. Setting the SSPM<3:0> bits of the SSPx-CON1 register determines the rate for the corresponding module.

An exception is when both modules use Timer2 as a time base in Master mode. In this instance, any changes to the Timer2 module's operation will affect both MSSP modules equally. If different bit rates are required for each module, the user should select one of the other three time base options for one of the modules.

TABLE 20-2: REGISTERS ASSOCIATED WITH SPI OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF(2)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(2)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(2)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
TRISC	TRISC7	TRISC6	—	—	—	TRISC2	TRISC1	TRISC0
TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
SSP1BUF	MSSP1 Receive Buffer/Transmit Register							
SSPxCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
SSPxSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF
SSP2BUF	MSSP2 Receive Buffer/Transmit Register							
ODCON3(1)	CTMUDS	—	—	—	—	—	SPI2OD	SPI1OD

Legend: Shaded cells are not used by the MSSPx module in SPI mode.

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

2: These bits are only available on 44-pin devices.

20.4 SPI DMA MODULE

The SPI DMA module contains control logic to allow the MSSP2 module to perform SPI direct memory access transfers. This enables the module to quickly transmit or receive large amounts of data with relatively little CPU intervention. When the SPI DMA module is used, MSSP2 can directly read and write to general purpose SRAM. When the SPI DMA module is not enabled, MSSP2 functions normally, but without DMA capability.

The SPI DMA module is composed of control logic, a Destination Receive Address Pointer, a Transmit Source Address Pointer, an interrupt manager and a Byte Count register for setting the size of each DMA transfer. The DMA module may be used with all SPI Master and Slave modes, and supports both half-duplex and full-duplex transfers.

20.4.1 I/O PIN CONSIDERATIONS

When enabled, the SPI DMA module uses the MSSP2 module. All SPI input and output signals, related to MSSP2, are routed through the Peripheral Pin Select module. The appropriate initialization procedure, as described in **Section 20.4.6 “Using the SPI DMA Module”**, will need to be followed prior to using the SPI DMA module. The output pins assigned to the SDO2 and SCK2 functions can optionally be configured as open-drain outputs, such as for level shifting operations mentioned in the same section.

20.4.2 RAM TO RAM COPY OPERATIONS

Although the SPI DMA module is primarily intended to be used for SPI communication purposes, the module can also be used to perform RAM to RAM copy operations. To do this, configure the module for Full-Duplex Master mode operation, but assign the SDO2 output and SDI2 input functions onto the same RPn pin in the PPS module. Also assign SCK2 out and SCK2 in onto the same RPn pin (a different pin than used for SDO2 and SDI2). This will allow the module to operate in Loopback mode, providing RAM copy capability.

20.4.3 IDLE AND SLEEP CONSIDERATIONS

The SPI DMA module remains fully functional when the microcontroller is in Idle mode.

During normal Sleep, the SPI DMA module is not functional and should not be used. To avoid corrupting a transfer, user firmware should be careful to make certain that pending DMA operations are complete by polling the DMAEN bit in the DMACON1 register prior to putting the microcontroller into Sleep.

In SPI Slave modes, the MSSP2 module is capable of transmitting and/or receiving one byte of data while in Sleep mode. This allows the SSP2IF flag in the PIR3 register to be used as a wake-up source. When the DMAEN bit is cleared, the SPI DMA module is effectively disabled, and the MSSP2 module functions normally, but without DMA capabilities. If the DMAEN bit is clear prior to entering Sleep, it is still possible to use the SSP2IF as a wake-up source without any data loss.

Neither MSSP2 nor the SPI DMA module will provide any functionality in Deep Sleep. Upon exiting from Deep Sleep, all of the I/O pins, MSSP2 and SPI DMA related registers will need to be fully re-initialized before the SPI DMA module can be used again.

20.4.4 REGISTERS

The SPI DMA engine is enabled and controlled by the following Special Function Registers:

- DMACON1
- TXADDRH
- RXADDRH
- DMABCH
- DMACON2
- TXADDRL
- RXADDRL
- DMABCL

20.4.4.1 DMACON1

The DMACON1 register is used to select the main operating mode of the SPI DMA module. The SSCON1 and SSCON0 bits are used to control the slave select pin.

When MSSP2 is used in SPI Master mode with the SPI DMA module, $\overline{\text{SSDMA}}$ can be controlled by the DMA module as an output pin. If MSSP2 will be used to communicate with an SPI slave device that needs the $\overline{\text{SSx}}$ pin to be toggled periodically, the SPI DMA hardware can automatically be used to deassert $\overline{\text{SSx}}$ between each byte, every two bytes or every four bytes.

Alternatively, user firmware can manually generate slave select signals with normal general purpose I/O pins, if required by the slave device(s).

When the TXINC bit is set, the TXADDR register will automatically increment after each transmitted byte. Automatic transmit address increment can be disabled by clearing the TXINC bit. If the automatic transmit address increment is disabled, each byte which is output on SDO2, will be the same (the contents of the SRAM pointed to by the TXADDR register) for the entire DMA transaction.

When the RXINC bit is set, the RXADDR register will automatically increment after each received byte. Automatic receive address increment can be disabled by clearing the RXINC bit. If RXINC is disabled in Full-Duplex or Half-Duplex Receive modes, all incoming data bytes on SDI2 will overwrite the same memory location pointed to by the RXADDR register. After the SPI DMA transaction has completed, the last received byte will reside in the memory location pointed to by the RXADDR register.

The SPI DMA module can be used for either half-duplex receive only communication, half-duplex transmit only communication or full-duplex simultaneous transmit and receive operations. All modes are available for both SPI master and SPI slave configurations. The DUPLEX0 and DUPLEX1 bits can be used to select the desired operating mode.

The behavior of the DLYINTEN bit varies greatly depending on the SPI operating mode. For example behavior for each of the modes, see Figure 20-3 through Figure 20-6.

SPI Slave mode, DLYINTEN = `1`: In this mode, an SSP2IF interrupt will be generated during a transfer if the time between successful byte transmission events is longer than the value set by the DLYCYC<3:0> bits in the DMACON2 register. This interrupt allows slave firmware to know that the master device is taking an unusually large amount of time between byte transmissions. For example, this information may be useful for implementing application defined communication protocols involving time-outs if the bus remains Idle for too long. When DLYINTEN = `1`, the DLYCYC<3:0> interrupts occur normally according to the selected setting.

SPI Slave mode, DLYINTEN = `0`: In this mode, the time-out based interrupt is disabled. No additional SSP2IF interrupt events will be generated by the SPI DMA module, other than those indicated by the INTLVL<3:0> bits in the DMACON2 register. In this mode, always set DLYCYC<3:0> = `0000`.

SPI Master mode, DLYINTEN = `0`: The DLYCYC<3:0> bits in the DMACON2 register determine the amount of additional inter-byte delay, which is added by the SPI DMA module during a transfer. The Master mode $\overline{\text{SS2}}$ output feature may be used.

SPI Master mode, DLYINTEN = `1`: The amount of hardware overhead is slightly reduced in this mode, and the minimum inter-byte delay is 8 TCY for FOSC/4, 9 TCY for FOSC/16 and 15 TCY for FOSC/64. This mode can potentially be used to obtain slightly higher effective SPI bandwidth. In this mode, the $\overline{\text{SS2}}$ control feature cannot be used, and should always be disabled (DMACON1<7:6> = `00`). Additionally, the interrupt generating hardware (used in Slave mode) remains active. To avoid extraneous SSP2IF interrupt events, set the DMACON2 delay bits, DLYCYC<3:0> = `1111`, and ensure that the SPI serial clock rate is no slower than FOSC/64.

In SPI Master modes, the DMAEN bit is used to enable the SPI DMA module and to initiate an SPI DMA transaction. After user firmware sets the DMAEN bit, the DMA hardware will begin transmitting and/or receiving data bytes according to the configuration used. In SPI Slave modes, setting the DMAEN bit will finish the initialization steps needed to prepare the SPI DMA module for communication (which still must be initiated by the master device).

To avoid possible data corruption, once the DMAEN bit is set, user firmware should not attempt to modify any of the MSSP2 or SPI DMA related registers, with the exception of the INTLVL bits in the DMACON2 register.

If user firmware wants to Halt an ongoing DMA transaction, the DMAEN bit can be manually cleared by the firmware. Clearing the DMAEN bit while a byte is currently being transmitted will not immediately Halt the byte in progress. Instead, any byte currently in progress will be completed before the MSSP2 and SPI DMA modules go back to their Idle conditions. If user firmware clears the DMAEN bit, the TXADDR, RXADDR and DMABC registers will no longer update, and the DMA module will no longer make any additional read or writes to SRAM; therefore, state information can be lost.

REGISTER 20-3: DMACON1: DMA CONTROL REGISTER 1 (ACCESS F88h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSCON1	SSCON0	TXINC	RXINC	DUPLEX1	DUPLEX0	DLYINTEN	DMAEN
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 **SSCON<1:0>:** $\overline{\text{SSDMA}}$ Output Control bits (Master modes only)

`11` = $\overline{\text{SSDMA}}$ is asserted for the duration of 4 bytes; DLYINTEN is always reset low
`01` = $\overline{\text{SSDMA}}$ is asserted for the duration of 2 bytes; DLYINTEN is always reset low
`10` = $\overline{\text{SSDMA}}$ is asserted for the duration of 1 byte; DLYINTEN is always reset low
`00` = $\overline{\text{SSDMA}}$ is not controlled by the DMA module; the DLYINTEN bit is software programmable

bit 5 **TXINC:** Transmit Address Increment Enable bit

Allows the transmit address to increment as the transfer progresses.

`1` = The transmit address is to be incremented from the initial value of TXADDR<11:0>
`0` = The transmit address is always set to the initial value of TXADDR<11:0>

bit 4 **RXINC:** Receive Address Increment Enable bit

Allows the receive address to increment as the transfer progresses.

`1` = The received address is to be incremented from the initial value of RXADDR<11:0>
`0` = The received address is always set to the initial value of RXADDR<11:0>

bit 3-2 **DUPLEX<1:0>:** Transmit/Receive Operating Mode Select bits

`10` = SPI DMA operates in Full-Duplex mode; data is simultaneously transmitted and received
`01` = DMA operates in Half-Duplex mode; data is transmitted only
`00` = DMA operates in Half-Duplex mode; data is received only

bit 1 **DLYINTEN:** Delay Interrupt Enable bit

Enables the interrupt to be invoked after the number of T_{CY} cycles specified in DLYCYC<2:0> has elapsed from the latest completed transfer.

`1` = The interrupt is enabled; SSCON<1:0> must be set to ‘`00`’
`0` = The interrupt is disabled

bit 0 **DMAEN:** DMA Operation Start/Stop bit

This bit is set by the users’ software to start the DMA operation. It is reset back to zero by the DMA engine when the DMA operation is completed or aborted.

`1` = DMA is in session
`0` = DMA is not in session

20.4.4.2 DMACON2

The DMACON2 register contains control bits for controlling interrupt generation and inter-byte delay behavior. The INTLVL<3:0> bits are used to select when an SSP2IF interrupt should be generated. The function of the DLYCYC<3:0> bits depends on the SPI operating mode (Master/Slave), as well as the DLYINTEN setting. In SPI Master mode, the DLYCYC<3:0> bits can be used to control how much time the module will Idle between bytes in a transfer. By default, the hardware requires a minimum delay of 8 TCY for FOSC/4, 9 TCY for FOSC/16 and 15 TCY for FOSC/64. An additional delay can be added with the DLYCYC bits. In SPI Slave modes, the DLYCYC<3:0> bits may optionally be used to trigger an additional time-out based interrupt.

REGISTER 20-4: DMACON2: DMA CONTROL REGISTER 2 (ACCESS F86h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DLYCYC3	DLYCYC2	DLYCYC1	DLYCYC0	INTLVL3	INTLVL2	INTLVL1	INTLVL0
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 **DLYCYC<3:0>:** Delay Cycle Selection bits

When DLYINTEN = `0`, these bits specify the additional delay (above the base overhead of the hardware) in number of TCY cycles before the SSP2BUF register is written again for the next transfer. When DLYINTEN = `1`, these bits specify the delay in number of TCY cycles from the latest completed transfer before an interrupt to the CPU is invoked. In this case, the additional delay before the SSP2BUF register is written again is 1 TCY + (base overhead of hardware).

`1111` = Delay time in number of instruction cycles is 2,048 cycles
`1110` = Delay time in number of instruction cycles is 1,024 cycles
`1101` = Delay time in number of instruction cycles is 896 cycles
`1100` = Delay time in number of instruction cycles is 768 cycles
`1011` = Delay time in number of instruction cycles is 640 cycles
`1010` = Delay time in number of instruction cycles is 512 cycles
`1001` = Delay time in number of instruction cycles is 384 cycles
`1000` = Delay time in number of instruction cycles is 256 cycles
`0111` = Delay time in number of instruction cycles is 128 cycles
`0110` = Delay time in number of instruction cycles is 64 cycles
`0101` = Delay time in number of instruction cycles is 32 cycles
`0100` = Delay time in number of instruction cycles is 16 cycles
`0011` = Delay time in number of instruction cycles is 8 cycles
`0010` = Delay time in number of instruction cycles is 4 cycles
`0001` = Delay time in number of instruction cycles is 2 cycles
`0000` = Delay time in number of instruction cycles is 1 cycle

bit 3-0 **INTLVL<3:0>:** Watermark Interrupt Enable bits

These bits specify the amount of remaining data yet to be transferred (transmitted and/or received) upon which an interrupt is generated.

`1111` = Amount of remaining data to be transferred is 576 bytes
`1110` = Amount of remaining data to be transferred is 512 bytes
`1101` = Amount of remaining data to be transferred is 448 bytes
`1100` = Amount of remaining data to be transferred is 384 bytes
`1011` = Amount of remaining data to be transferred is 320 bytes
`1010` = Amount of remaining data to be transferred is 256 bytes
`1001` = Amount of remaining data to be transferred is 192 bytes
`1000` = Amount of remaining data to be transferred is 128 bytes
`0111` = Amount of remaining data to be transferred is 67 bytes
`0110` = Amount of remaining data to be transferred is 32 bytes
`0101` = Amount of remaining data to be transferred is 16 bytes
`0100` = Amount of remaining data to be transferred is 8 bytes
`0011` = Amount of remaining data to be transferred is 4 bytes
`0010` = Amount of remaining data to be transferred is 2 bytes
`0001` = Amount of remaining data to be transferred is 1 byte
`0000` = Transfer complete

20.4.4.3 DMABCH and DMABCL

The DMABCH and DMABCL register pair forms a 10-bit Byte Count register, which is used by the SPI DMA module to send/receive up to 1,024 bytes for each DMA transaction. When the DMA module is actively running (DMAEN = `1`), the DMA Byte Count register decrements after each byte is transmitted/received. The DMA transaction will Halt and the DMAEN bit will be automatically cleared by hardware after the last byte has completed. After a DMA transaction is complete, the DMABC register will read `0x000`.

Prior to initiating a DMA transaction by setting the DMAEN bit, user firmware should load the appropriate value into the DMABCH/DMABCL registers. The DMABC is a "base zero" counter, so the actual number of bytes which will be transmitted follows in Equation 20-1.

For example, if user firmware wants to transmit 7 bytes in one transaction, DMABC should be loaded with 006h. Similarly, if user firmware wishes to transmit 1,024 bytes, DMABC should be loaded with 3FFh.

EQUATION 20-1: BYTES TRANSMITTED FOR A GIVEN DMABC

$$\text{Bytes}_{XMIT} \equiv (\text{DMABC} + 1)$$

20.4.4.4 TXADDRH and TXADDRL

The TXADDRH and TXADDRL registers pair together to form a 12-bit Transmit Source Address Pointer register. In modes that use TXADDR (Full-Duplex and Half-Duplex Transmit), the TXADDR will be incremented after each byte is transmitted. Transmitted data bytes will be taken from the memory location pointed to by the TXADDR register. The contents of the memory locations pointed to by TXADDR will not be modified by the DMA module during a transmission.

The SPI DMA module can read from, and transmit data from, all general purpose memory on the device. The SPI DMA module cannot be used to read from the Special Function Registers (SFRs) contained in Banks 14 and 15.

20.4.4.5 RXADDRH and RXADDRL

The RXADDRH and RXADDRL registers pair together to form a 12-bit Receive Destination Address Pointer. In modes that use RXADDR (Full-Duplex and Half-Duplex Receive), the RXADDR register will be incremented after each byte is received. Received data bytes will be stored at the memory location pointed to by the RXADDR register.

The SPI DMA module can write received data to all general purpose memory on the device. The SPI DMA module cannot be used to modify the Special Function Registers contained in Banks 14 and 15.

20.4.5 INTERRUPTS

The SPI DMA module alters the behavior of the SSP2IF interrupt flag. In normal non-DMA modes, the SSP2IF is set once after every single byte is transmitted/received through the MSSP2 module. When MSSP2 is used with the SPI DMA module, the SSP2IF interrupt flag will be set according to the user-selected INTLVL<3:0> value specified in the DMACON2 register. The SSP2IF interrupt condition will also be generated once the SPI DMA transaction has fully completed and the DMAEN bit has been cleared by hardware.

The SSP2IF flag becomes set once the DMA byte count value indicates that the specified INTLVL has been reached. For example, if DMACON2<3:0> = `0101` (16 bytes remaining), the SSP2IF interrupt flag will become set once DMABC reaches 00Fh. If user firmware then clears the SSP2IF interrupt flag, the flag will not be set again by the hardware until after all bytes have been fully transmitted and the DMA transaction is complete.

Note: User firmware may modify the INTLVL bits while a DMA transaction is in progress (DMAEN = `1`). If an INTLVL value is selected which is higher than the actual remaining number of bytes (indicated by DMABC + `1`), the SSP2IF interrupt flag will immediately become set.

For example, if DMABC = 00Fh (implying 16 bytes are remaining) and user firmware writes '`1111`' to INTLVL<3:0> (interrupt when 576 bytes remaining), the SSP2IF interrupt flag will immediately become set. If user firmware clears this interrupt flag, a new interrupt condition will not be generated until either: user firmware again writes INTLVL with an interrupt level higher than the actual remaining level, or the DMA transaction completes and the DMAEN bit is cleared.

Note: If the INTLVL bits are modified while a DMA transaction is in progress, care should be taken to avoid inadvertently changing the DLYCYC<3:0> value.

20.4.6 USING THE SPI DMA MODULE

The following steps would typically be taken to enable and use the SPI DMA module:

1. Configure the I/O pins, which will be used by MSSP2.
    a) Assign SCK2, SDO2, SDI2 and $\overline{\text{SS2}}$ to the RPn pins as appropriate for the SPI mode which will be used. Only functions which will be used need to be assigned to a pin.
    b) Initialize the associated LAT registers for the desired Idle SPI bus state.
    c) If Open-Drain Output mode on SDO2 and SCK2 (Master mode) is desired, set ODCON3<1>.
    d) Configure the corresponding TRIS bits for each I/O pin used.
2. Configure and enable MSSP2 for the desired SPI operating mode.
    a) Select the desired operating mode (Master or Slave, SPI Mode 0, 1, 2 and 3) and configure the module by writing to the SSP2STAT and SSP2CON1 registers.
    b) Enable MSSP2 by setting SSP2CON1<5> = `1`.
3. Configure the SPI DMA engine.
    a) Select the desired operating mode by writing the appropriate values to DMACON2 and DMACON1.
    b) Initialize the TXADDRH/TXADDRL Pointer (Full-Duplex or Half-Duplex Transmit Only mode).
    c) Initialize the RXADDRH/RXADDRL Pointer (Full-Duplex or Half-Duplex Receive Only mode).
    d) Initialize the DMABCH/DMABCL Byte Count register with the number of bytes to be transferred in the next SPI DMA operation.
    e) Set the DMAEN bit (DMACON1<0>).

       In SPI Master modes, this will initiate a DMA transaction. In SPI Slave modes, this will complete the initialization process, and the module will now be ready to begin receiving and/or transmitting data to the master device once the master starts the transaction.
4. Detect the SSP2IF interrupt condition (PIR3<7>).
    a) If the interrupt was configured to occur at the completion of the SPI DMA transaction, the DMAEN bit (DMACON1<0>) will be clear. User firmware may prepare the module for another transaction by repeating Steps 3.b through 3.e.
    b) If the interrupt was configured to occur prior to the completion of the SPI DMA transaction, the DMAEN bit may still be set, indicating the transaction is still in progress. User firmware would typically use this interrupt condition to begin preparing new data for the next DMA transaction. Firmware should not repeat Steps 3.b. through 3.e. until the DMAEN bit is cleared by the hardware, indicating the transaction is complete.

Example 20-2 provides example code demonstrating the initialization process and the steps needed to use the SPI DMA module to perform a 512-byte Full-Duplex Master mode transfer.

EXAMPLE 20-2: 512-BYTE SPI MASTER MODE `Init` AND TRANSFER

```
                                    ;For this example, let's use RP5(RB2) for SCK2,
                                    ;RP4(RB1) for SDO2, and RP3(RB0) for SDI2

                                    ;Let's use SPI master mode, CKE = 0, CKP = 0,
                                    ;without using slave select signalling.

InitSPIPins:
   movlb     0x0F                   ;Select bank 15, for access to ODCON3 register
   bcf       ODCON3, SPI2OD         ;Let's not use open drain outputs in this example

   bcf       LATB, RB2              ;Initialize our (to be) SCK2 pin low (idle).
   bcf       LATB, RB1              ;Initialize our (to be) SDO2 pin to an idle state
   bcf       TRISB, RB1             ;Make SDO2 output, and drive low
   bcf       TRISB, RB2             ;Make SCK2 output, and drive low (idle state)
   bsf       TRISB, RB0             ;SDI2 is an input, make sure it is tri-stated

                                    ;Now we should unlock the PPS registers, so we can
                                    ;assign the MSSP2 functions to our desired I/O pins.

   movlb     0x0E                   ;Select bank 14 for access to PPS registers
   bcf       INTCON, GIE            ;I/O Pin unlock sequence will not work if CPU
                                    ;services an interrupt during the sequence
   movlw     0x55                   ;Unlock sequence consists of writing 0x55
   movwf     EECON2                 ;and 0xAA to the EECON2 register.
   movlw     0xAA
   movwf     EECON2
   bcf       PPSCON, IOLOCK         ;We may now write to RPINRx and RPORx registers
   bsf       INTCON, GIE            ;May now turn back on interrupts if desired

   movlw     0x03                   ;RP3 will be SDI2
   movwf     RPINR21                ;Assign the SDI2 function to pin RP3

   movlw     0x0A                   ;Let's assign SCK2 output to pin RP4
   movwf     RPOR4                  ;RPOR4 maps output signals to RP4 pin
   movlw     0x04                   ;SCK2 also needs to be configured as an input on the
                                    same pin
   movwf     RPINR22                ;SCK2 input function taken from RP4 pin
   movlw     0x09                   ;0x09 is SDO2 output
   movwf     RPOR5                  ;Assign SDO2 output signal to the RP5 (RB2) pin
   movlb     0x0F                   ;Done with PPS registers, bank 15 has other SFRs

InitMSSP2:
   clrf      SSP2STAT               ;CKE = 0, SMP = 0 (sampled at middle of bit)
   movlw     b'00000000'            ;CKP = 0, SPI Master mode, Fosc/4
   movwf     SSP2CON1               ;MSSP2 initialized
   bsf       SSP2CON1, SSPEN        ;Enable the MSSP2 module

InitSPIDMA:
   movlw     b'00111010'            ;Full duplex, RX/TXINC enabled, no SSCON
   movwf     DMACON1                ;DLYINTEN is set, so DLYCYC3:DLYCYC0 = 1111
   movlw     b'11110000'            ;Minimum delay between bytes, interrupt
   movwf     DMACON2                ;only once when the transaction is complete
```

EXAMPLE 20-2: 512-BYTE SPI MASTER MODE `Init` AND TRANSFER (CONTINUED)

```
                                    ;Somewhere else in our project, lets assume we have
                                    ;allocated some RAM for use as SPI receive and
                                    ;transmit buffers.

;            udata     0x500
;DestBuf     res       0x200        ;Let’s reserve 0x500-0x6FF for use as our SPI
;                                   ;receive data buffer in this example
;SrcBuf      res       0x200        ;Lets reserve 0x700-0x8FF for use as our SPI
;                                   ;transmit data buffer in this example

PrepareTransfer:
   movlw     HIGH(DestBuf)          ;Get high byte of DestBuf address (0x05)
   movwf     RXADDRH                ;Load upper four bits of the RXADDR register
   movlw     LOW(DestBuf)           ;Get low byte of the DestBuf address (0x00)
   movwf     RXADDRL                ;Load lower eight bits of the RXADDR register

   movlw     HIGH(SrcBuf)           ;Get high byte of SrcBuf address (0x07)
   movwf     TXADDRH                ;Load upper four bits of the TXADDR register
   movlw     LOW(SrcBuf)            ;Get low byte of the SrcBuf address (0x00)
   movwf     TXADDRL                ;Load lower eight bits of the TXADDR register

   movlw     0x01                   ;Lets move 0x200 (512) bytes in one DMA xfer
   movwf     DMABCH                 ;Load the upper two bits of DMABC register
   movlw     0xFF                   ;Actual bytes transferred is (DMABC + 1), so
   movwf     DMABCL                 ;we load 0x01FF into DMABC to xfer 0x200 bytes

BeginXfer:
   bsf       DMACON1, DMAEN         ;The SPI DMA module will now begin transferring
                                    ;the data taken from SrcBuf, and will store
                                    ;received bytes into DestBuf.

   ;Execute whatever                ;CPU is now free to do whatever it wants to
                                    ;and the DMA operation will continue without
                                    ;intervention, until it completes.

                                    ;When the transfer is complete, the SSP2IF flag in
                                    ;the PIR3 register will become set, and the DMAEN bit
                                    ;is automatically cleared by the hardware.
                                    ;The DestBuf (0x500-0x7FF) will contain the received
                                    ;data.  To start another transfer, firmware will need
                                    ;to reinitialize RXADDR, TXADDR, DMABC and then
                                    ;set the DMAEN bit.
```

20.5 I^{2}C Mode

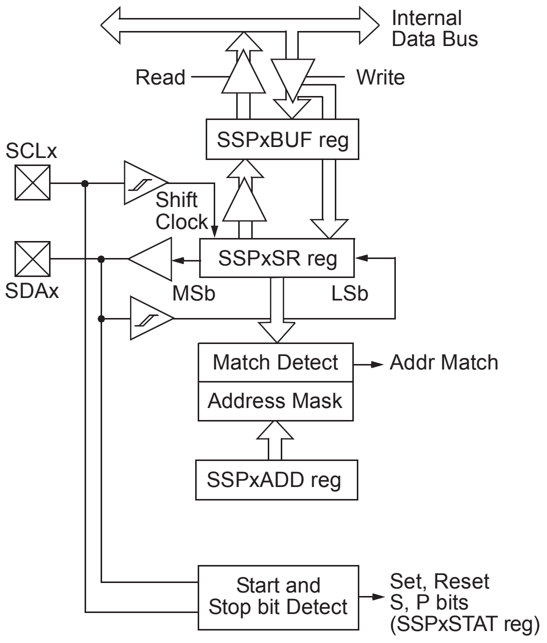
The MSSP module in I^{2}C mode fully implements all master and slave functions (including general call support), and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications and 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial Clock (SCLx) – RC3/SCK1/SCL1/RP14, RD0/PMD0/SCL2 (44-pin devices) or RB4/CCP4/PMA1/KBI0/SCL2/RP7 (28-pin devices)
- Serial Data (SDAx) – RC4/SDI1/SDA1/RP15, RD1/PMD1/SDA2 (44-pin devices) or RB5/CCP5/KBI1/SDA2/RP8 (28-pin devices)

The user must configure these pins as inputs by setting the associated TRISx bits. These pins are up to 5.5V tolerant, allowing direct use in I^{2}C buses operating at voltages higher than VDD.

FIGURE 20-7: MSSPx BLOCK DIAGRAM (I^{2}C MODE)



Note: Only port I/O names are used in this diagram for the sake of brevity. Refer to the text for a full list of multiplexed functions.

20.5.1 REGISTERS

The MSSP module has six registers for I^{2}C operation. These are:

- MSSPx Control Register 1 (SSPxCON1)
- MSSPx Control Register 2 (SSPxCON2)
- MSSPx Status Register (SSPxSTAT)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSPx Shift Register (SSPxSR) – Not directly accessible
- MSSPx Address Register (SSPxADD)
- MSSPx 7-Bit Address Mask Register (SSPxMSK)

SSPxCON1, SSPxCON2 and SSPxSTAT are the control and status registers in I^{2}C mode operation. The SSPxCON1 and SSPxCON2 registers are readable and writable. The lower six 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 I^{2}C Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPxADD act as the Baud Rate Generator (BRG) reload value.

SSPxMSK holds the slave address mask value when the module is configured for 7-Bit Address Masking mode. While it is a separate register, it shares the same SFR address as SSPxADD; it is only accessible when the SSPM<3:0> bits are specifically set to permit access. Additional details are provided in **Section 20.5.3.4 "7-Bit Address Masking Mode"**.

In receive operations, SSPxSR and SSPxBUF together, create a double-buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

REGISTER 20-5: SSPxSTAT: MSSPx STATUS REGISTER (I^2C MODE) (1, ACCESS FC7h; 2, F73h)

R/W-1	R/W-1	R-1	R-1	R-1	R-1	R-1	R-1
SMP	CKE	D/$\overline{\text{A}}$	P[1]	S[1]	R/$\overline{\text{W}}$[2,3]	UA	BF
bit 7							bit 0

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

bit 7 **SMP:** Slew Rate Control bit

In Master or Slave mode:
1 = Slew rate control is disabled for Standard Speed mode (100 kHz and 1 MHz)
0 = Slew rate control is enabled for High-Speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit

In Master or Slave mode:
1 = Enable SMBus specific inputs
0 = Disable SMBus specific inputs

bit 5 **D/$\overline{\text{A}}$:** Data/$\overline{\text{Address}}$ bit

In Master mode:
Reserved.

In Slave mode:
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit[1]

1 = Indicates that a Stop bit has been detected last
0 = Stop bit was not detected last

bit 3 **S:** Start bit[1]

1 = Indicates that a Start bit has been detected last
0 = Start bit was not detected last

bit 2 **R/$\overline{\text{W}}$:** Read/$\overline{\text{Write}}$ Information bit[2,3]

In Slave mode:
1 = Read
0 = Write

In Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress

bit 1 **UA:** Update Address bit (10-Bit Slave mode only)

1 = Indicates that the user needs to update the address in the SSPxADD register
0 = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit

In Transmit mode:
1 = SSPxBUF is full
0 = SSPxBUF is empty

In Receive mode:
1 = SSPxBUF is full (does not include the $\overline{\text{ACK}}$ and Stop bits)
0 = SSPxBUF is empty (does not include the $\overline{\text{ACK}}$ and Stop bits)

Note 1: This bit is cleared on Reset and when SSPEN is cleared.
2: This bit holds the R/$\overline{\text{W}}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{\text{ACK}}$ bit.
3: ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSPx is in Active mode.

REGISTER 20-6: SSPxCON1: MSSPx CONTROL REGISTER 1 (I^2C MODE) (1, ACCESS FC6h; 2, F73h)

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 the serial port and configures these pins as I/O port pins

bit 4 **CKP:** SCKx Release Control bit

In Slave mode:
`1` = Releases clock
`0` = Holds clock low (clock stretch); used to ensure data setup time

In Master mode:
Unused in this mode.

bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits[2]
`1111` = 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 the SSPxMSK register at the 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`’).

REGISTER 20-7: SSPxCON2: MSSPx CONTROL REGISTER 2 (I²C MASTER MODE) (1, ACCESS FC5h; 2, F71h)

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN(3)	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 (Slave mode only)(3)
`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 (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 Idle

bit 3 **RCEN:** Receive Enable bit (Master Receive mode only)(2)
`1` = Enables Receive mode for I²C
`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 I²C module is active, these bits may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).
3: This bit is not implemented in I²C Master mode.

REGISTER 20-8: SSPxCON2: MSSPx CONTROL REGISTER 2 (I^{2}C SLAVE MODE) (1, ACCESS FC5h; 2, F71h)

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT(2)	ADMSK5	ADMSK4	ADMSK3	ADMSK2	ADMSK1	SEN(1)
bit 7							bit 0

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

bit 7 **GCEN:** General Call Enable bit (Slave mode only)
`1` = Enables interrupt when a general call address (0000h) is received in the SSPxSR
`0` = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit(2)
Unused in Slave mode.

bit 5-2 **ADMSK<5:2>:** Slave Address Mask Select bits (5-bit address masking)
`1` = Masking of the corresponding bits of SSPxADD is enabled
`0` = Masking of the corresponding bits of SSPxADD is disabled

bit 1 **ADMSK1:** Slave Address Least Significant bit(s) Mask Select bit
In 7-Bit Addressing mode:
`1` = Masking of SSPxADD<1> only is enabled
`0` = Masking of SSPxADD<1> only is disabled
In 10-Bit Addressing mode:
`1` = Masking of SSPxADD<1:0> is enabled
`0` = Masking of SSPxADD<1:0> is disabled

bit 0 **SEN:** Start Condition Enable/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^{2}C 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).
2: This bit is unimplemented in I^{2}C Slave mode.

REGISTER 20-9: SSPxMSK: I^{2}C SLAVE ADDRESS MASK REGISTER (7-BIT MASKING MODE) (1, ACCESS FC8h; 2, F74h)(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 bits
`1` = Masking of the corresponding bit of SSPxADD is enabled
`0` = Masking of the corresponding bit of SSPxADD is disabled

Note 1: This register shares the same SFR address as SSPxADD and is only addressable in select MSSP operating modes. See **Section 20.5.3.4 “7-Bit Address Masking Mode”** for more details.
2: MSK0 is not used as a mask bit in 7-bit addressing.

20.5.2 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 TRISB or TRISD bits. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCLx and SDAx pins.

20.5.3 SLAVE MODE

In Slave mode, the SCLx and SDAx pins must be configured as inputs (TRISB<5:4> 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 will automatically generate the Acknowledge ($\overline{\text{ACK}}$) 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 $\overline{\text{ACK}}$ 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.

20.5.3.1 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 $\overline{\text{ACK}}$ pulse is generated.
4. The MSSPx 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. Bit, R/$\overline{\text{W}}$ (SSPxSTAT<2>), must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '`11110 A9 A8 0`', where '`A9`' and '`A8`' are the two MSbs of the address. The sequence of events for 10-bit addressing is as follows, with Steps 7 through 9 for the slave-transmitter:

1. Receive the first (high) byte of address (bits, SSPxIF, BF and UA, are set on an address match).
2. Update the SSPxADD register with the second (low) byte of the address (clears bit, UA, and releases the SCLx line).
3. Read the SSPxBUF register (clears bit, BF) and clear flag bit, SSPxIF.
4. Receive the second (low) byte of the address (bits, SSPxIF, BF and UA, are set).
5. Update the SSPxADD register with the first (high) byte of the address. If a match releases the SCLx line, this will clear bit, UA.
6. Read the SSPxBUF register (clears bit, BF) and clears flag bit, SSPxIF.
7. Receive Repeated Start condition.
8. Receive the first (high) byte of address (bits, SSPxIF and BF, are set).
9. Read the SSPxBUF register (clears bit, BF) and clear flag bit, SSPxIF.

20.5.3.2 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 SSPxBUF.

The PIC18F47J13 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 is different.

20.5.3.3 5-Bit Address Masking Mode

As the name implies, 5-Bit Address Masking mode uses an address mask of up to five bits to create a range of addresses to be Acknowledged, using Bits 5 through 1 of the incoming address. This allows the module to Acknowledge up to 31 addresses when using 7-bit addressing or 63 addresses with 10-bit addressing (see Example 20-3). 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 20-8). 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 MSbs of the address are not affected by address masking.

EXAMPLE 20-3: 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 MSbs of the address are ignored in this example, since they are not affected by masking)

ADMSK<5:1> = `00111`

Addresses Acknowledged: A0h, A1h, A2h, A3h, A4h, A5h, A6h, A7h, A8h, A9h, AAh, ABh, ACh, ADh, AEh, AFh

20.5.3.4 7-Bit Address Masking Mode

Unlike 5-Bit Address Masking mode, 7-Bit Address Masking mode uses a mask of up to eight bits (in 10-bit addressing) to define a range of addresses than can be Acknowledged, using the lowest bits of the incoming address. This allows the module to Acknowledge up to 127 different addresses with 7-bit addressing, or 255 with 10-bit addressing (see Example 20-4). This mode is the default configuration of the module and is selected when MSSPMSK is unprogrammed (‘`1`’).

The address mask for 7-Bit Address Masking mode is stored in the SSPxMSK register, instead of the SSPxCON2 register. SSPxMSK is a separate, hardware register within the module, but it is not directly addressable. Instead, it shares an address in the SFR space with the SSPxADD register. To access the SSPxMSK register, it is necessary to select MSSP mode, ‘`1001`’ (SSPCON1<3:0> = `1001`), and then read or write to the location of SSPxADD.

To use 7-Bit Address Masking mode, it is necessary to initialize SSPxMSK with a value before selecting the I^{2}C Slave Addressing mode. Thus, the required sequence of events is:

1. Select SSPxMSK Access mode (SSPxCON2<3:0> = `1001`).
2. Write the mask value to the appropriate SSPxADD register address (FC8h for MSSP1, F74h for MSSP2).
3. Set the appropriate I^{2}C Slave mode (SSPxCON2<3:0> = `0111` for 10-bit addressing, `0110` for 7-bit addressing).

Setting or clearing mask bits in SSPxMSK behaves in the opposite manner of the ADMSK bits in 5-Bit Address Masking mode. That is, clearing a bit in SSPxMSK causes the corresponding address bit to be masked; setting the bit requires a match in that position. SSPxMSK resets to all ‘`1`’s upon any Reset condition, and therefore, has no effect on the standard MSSP operation until written with a mask value.

With 7-Bit Address Masking mode, the 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 Address Masking mode, 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 MSbs of the address are not affected by address masking.

EXAMPLE 20-4: 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 MSbs are ignored in this example since they are not affected)

SSPxMSK<5:1> = `1111 0011`

Addresses Acknowledged = ACh, A8h, A4h, A0h

20.5.3.5 Reception

When the R/$\overline{\text{W}}$ bit of the address byte is clear and an address match occurs, the R/$\overline{\text{W}}$ bit of the SSPxSTAT register is cleared. The received address is loaded into the SSPxBUF register and the SDAx line is held low ($\overline{\text{ACK}}$).

When the address byte overflow condition exists, then the no Acknowledge ($\overline{\text{ACK}}$) 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 (SSPx-CON1<4>). See **Section 20.5.4 “Clock Stretching”** for more details.

20.5.3.6 Transmission

When the R/$\overline{\text{W}}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{\text{W}}$ bit of the SSPxSTAT register is set. The received address is loaded into the SSPxBUF register. The $\overline{\text{ACK}}$ pulse will be sent on the ninth bit and pin, SCLx, is held low regardless of SEN (see **Section 20.5.4 “Clock Stretching”** for more details). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data. The transmit data must be loaded into the SSPxBUF register, which also loads the SSPxSR register. Then, the SCLx pin should be enabled by setting bit, CKP (SSPxCON1<4>). The eight data bits are shifted out on the falling edge of the SCLx input. This ensures that the SDAx signal is valid during the SCLx high time (Figure 20-10).

The $\overline{\text{ACK}}$ pulse from the master-receiver is latched on the rising edge of the ninth SCLx input pulse. If the SDAx line is high (not $\overline{\text{ACK}}$), then the data transfer is complete. In this case, when the $\overline{\text{ACK}}$ is latched by the slave, the slave monitors for another occurrence of the Start bit. If the SDAx line was low ($\overline{\text{ACK}}$), the next transmit data must be loaded into the SSPxBUF register. Again, the SCLx pin must be enabled by setting bit, CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPxIF bit must be cleared in software and the SSPxSTAT register is used to determine the status of the byte. The SSPxIF bit is set on the falling edge of the ninth clock pulse.

FIGURE 20-8: I²C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)

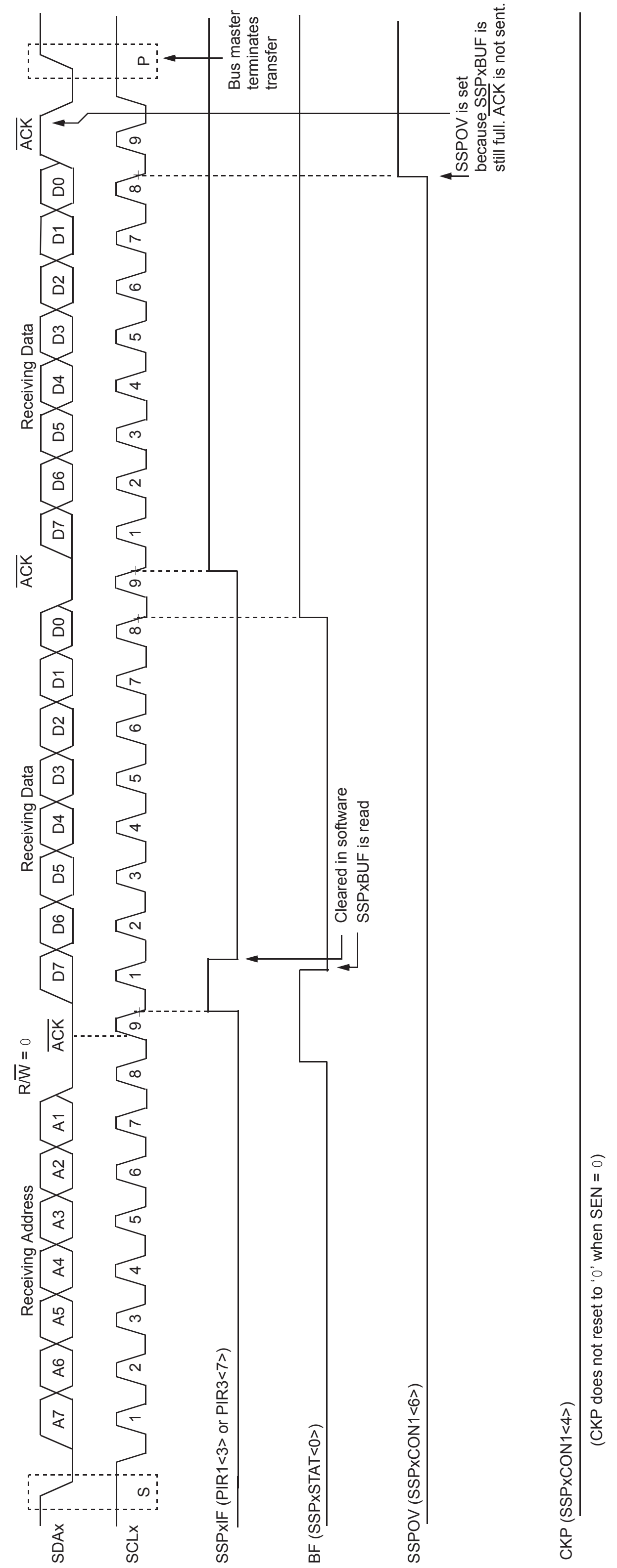
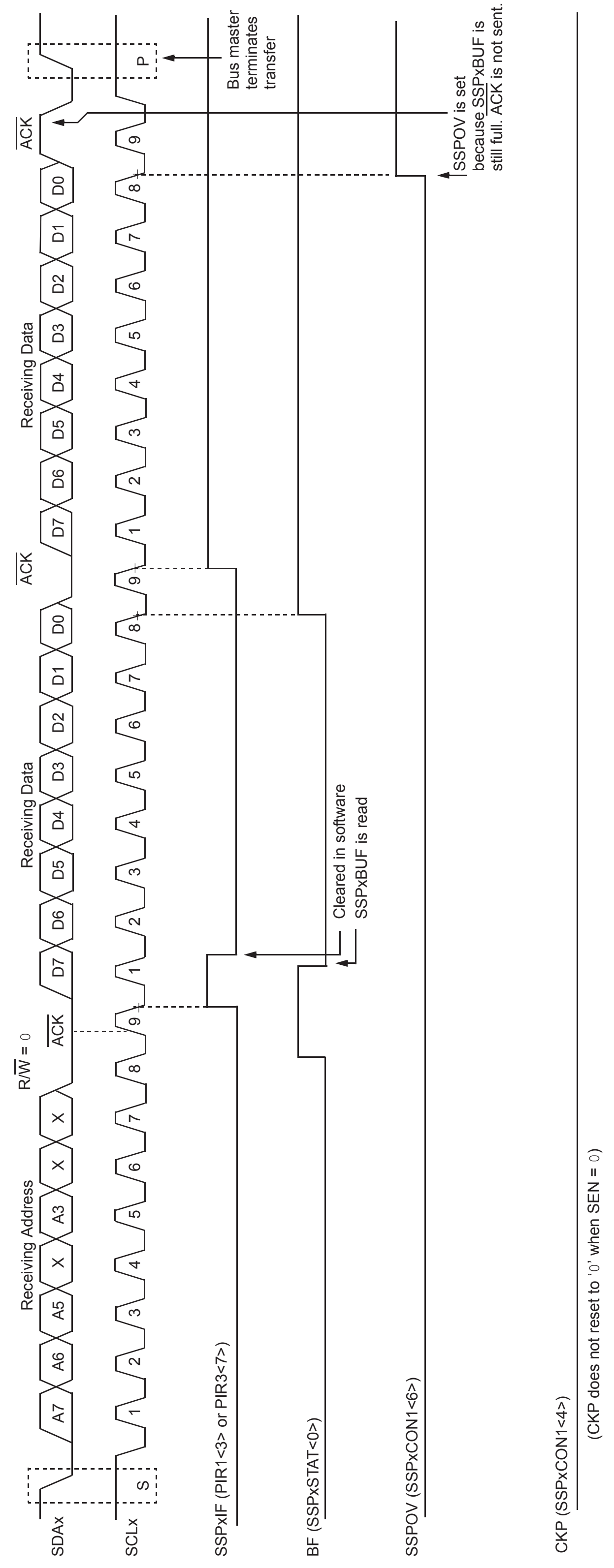

SDAx
SCLx
S
Receiving Address
A7
A6
A5
A4
A3
A2
A1
R/W = 0
ACK
1
2
3
4
5
6
7
8
9
Receiving Data
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Receiving Data
D7
D6
D5
D4
D3
D2
D1
D0
ACK
P
Bus master terminates transfer
SSPxIF (PIR1<3> or PIR3<7>)
BF (SSPxSTAT<0>)
Cleared in software
SSPxBUF is read
SSPOV (SSPxCON1<6>)
SSPOV is set because SSPxBUF is still full. ACK is not sent.
CKP (SSPxCON1<4>)
(CKP does not reset to '0' when SEN = 0)

FIGURE 20-9: I²C SLAVE MODE TIMING WITH SEN = `0` AND ADMSK<5:1> = `01011` (RECEPTION, 7-BIT ADDRESS)[1]


SDAx
SCLx
SSPxIF (PIR1<3> or PIR3<7>)
BF (SSPxSTAT<0>)
SSPOV (SSPxCON1<6>)
CKP (SSPxCON1<4>)
(CKP does not reset to '0' when SEN = 0)
Receiving Address
R/W = 0
ACK
Receiving Data
ACK
Receiving Data
ACK
S
A7
A6
A5
X
A3
X
X
D7
D6
D5
D4
D3
D2
D1
D0
1
2
3
4
5
6
7
8
9
P
Cleared in software
SSPxBUF is read
SSPOV is set because SSPxBUF is still full. ACK is not sent.
Bus master terminates transfer

Note 1: In this example, an address equal to A7.A6.A5.X.A3.X.X will be Acknowledged and cause an interrupt.

FIGURE 20-10: I^{2}C SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)

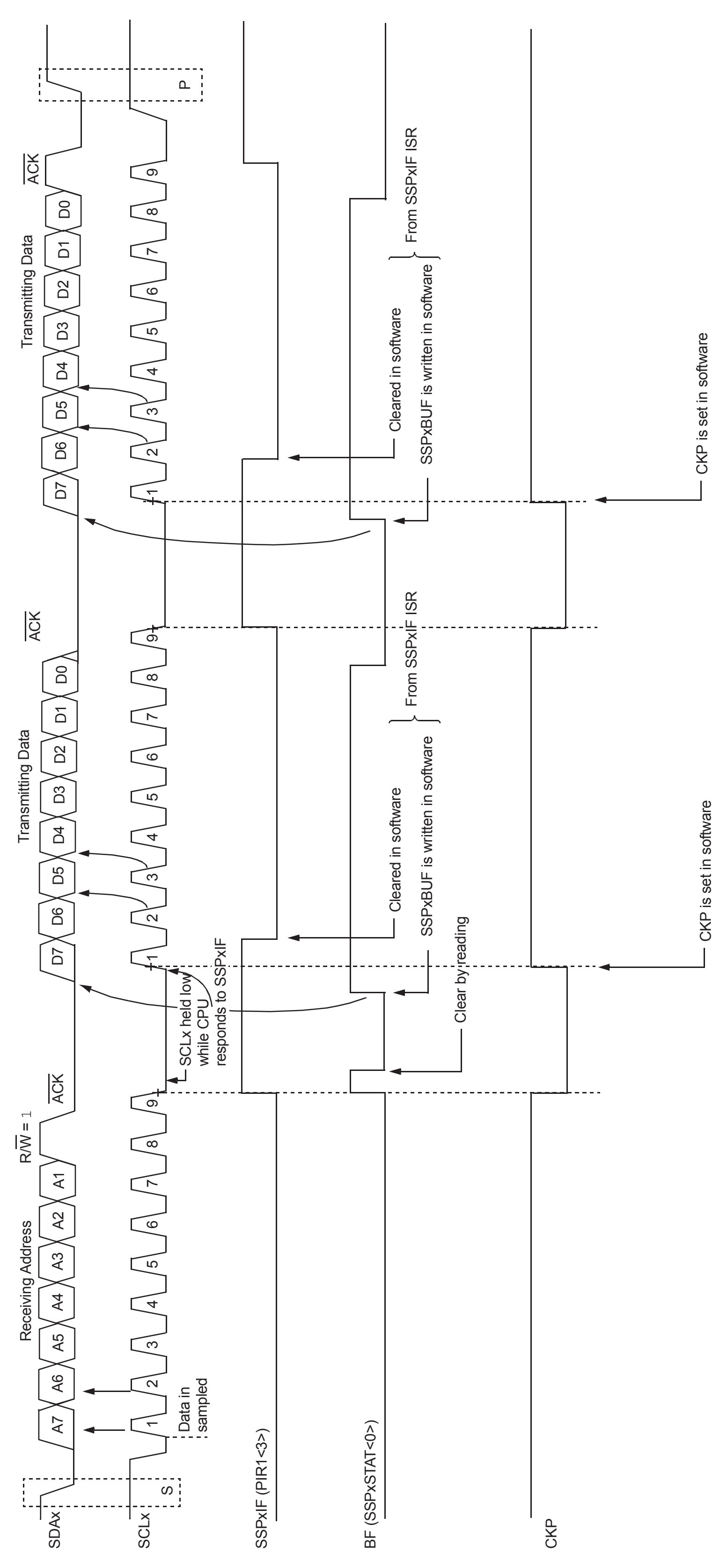
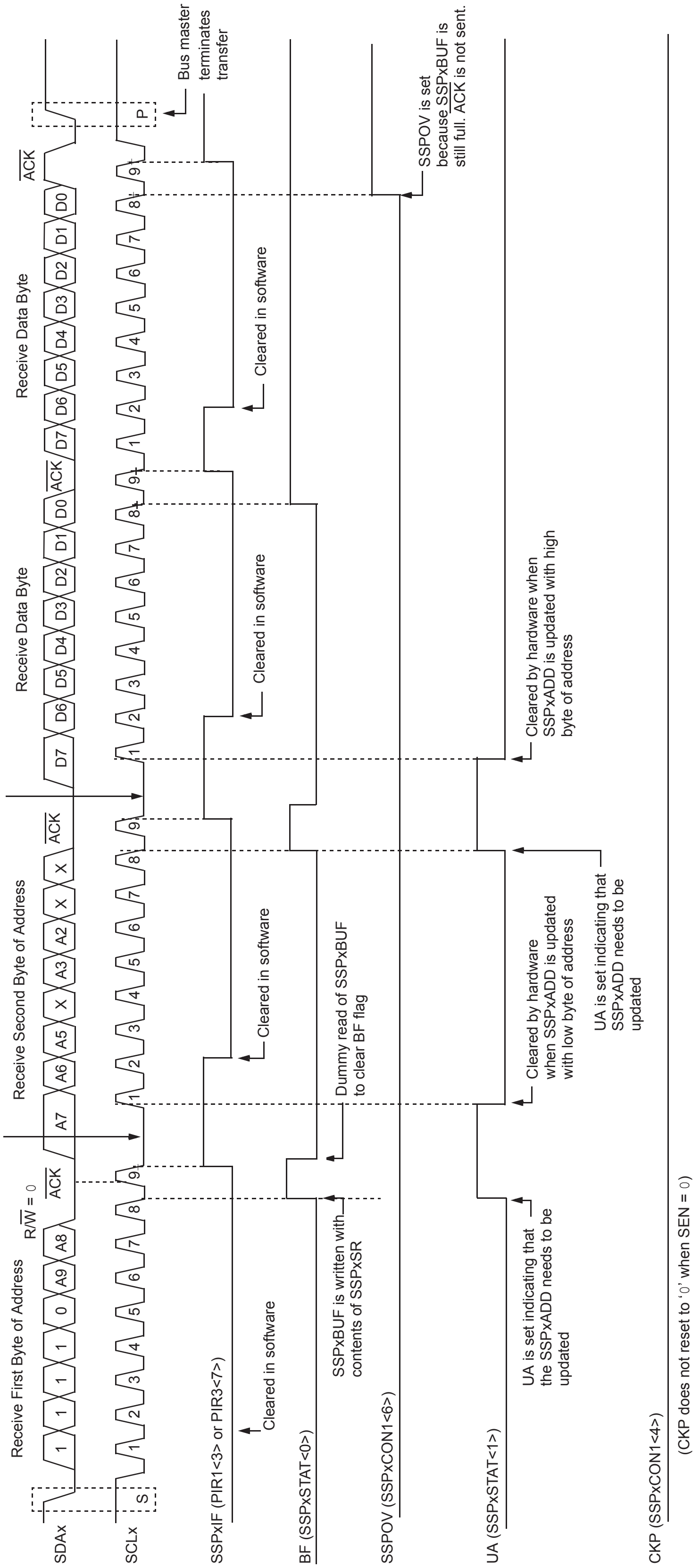

SDAx
SCLx
SSPxIF (PIR1<3>)
BF (SSPxSTAT<0>)
CKP
S
Receiving Address
A7
A6
A5
A4
A3
A2
A1
R/W = 1
ACK
1
2
3
4
5
6
7
8
9
Data in sampled
SCLx held low while CPU responds to SSPxIF
Clear by reading
Transmitting Data
D7
D6
D5
D4
D3
D2
D1
D0
ACK
1
2
3
4
5
6
7
8
9
Cleared in software
SSPxBUF is written in software
From SSPxIF ISR
CKP is set in software
Transmitting Data
D7
D6
D5
D4
D3
D2
D1
D0
ACK
1
2
3
4
5
6
7
8
9
Cleared in software
SSPxBUF is written in software
From SSPxIF ISR
CKP is set in software
P

FIGURE 20-11: I^2C SLAVE MODE TIMING WITH SEN = `0` AND ADMSK<5:1> = `01001` (RECEPTION, 10-BIT ADDRESS)[1,2]


SDAx
SCLx
S
P
Receive First Byte of Address
1 1 1 1 0 A9 A8
R/W = 0
ACK
Receive Second Byte of Address
A7 A6 A5 X A3 A2 X X
ACK
Receive Data Byte
D7 D6 D5 D4 D3 D2 D1 D0
ACK
Receive Data Byte
D7 D6 D5 D4 D3 D2 D1 D0
ACK
1 2 3 4 5 6 7 8 9
Clock is held low until update of SSPxADD has taken place
Clock is held low until update of SSPxADD has taken place
Bus master terminates transfer
SSPxIF (PIR1<3> or PIR3<7>)
Cleared in software
Cleared in software
Cleared in software
Cleared in software
BF (SSPxSTAT<0>)
SSPxBUF is written with contents of SSPxSR
Dummy read of SSPxBUF to clear BF flag
SSPOV (SSPxCON1<6>)
SSPOV is set because SSPxBUF is still full. ACK is not sent.
UA (SSPxSTAT<1>)
UA is set indicating that the SSPxADD needs to be updated
Cleared by hardware when SSPxADD is updated with low byte of address
UA is set indicating that SSPxADD needs to be updated
Cleared by hardware when SSPxADD is updated with high byte of address
CKP (SSPxCON1<4>)
(CKP does not reset to '0' when SEN = 0)

Note 1: In this example, an address equal to A9.A8.A7.A6.A5.X.A3.A2.X.X will be Acknowledged and cause an interrupt.

2: Note that the Most Significant bits of the address are not affected by the bit masking.

FIGURE 20-12: I²C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)

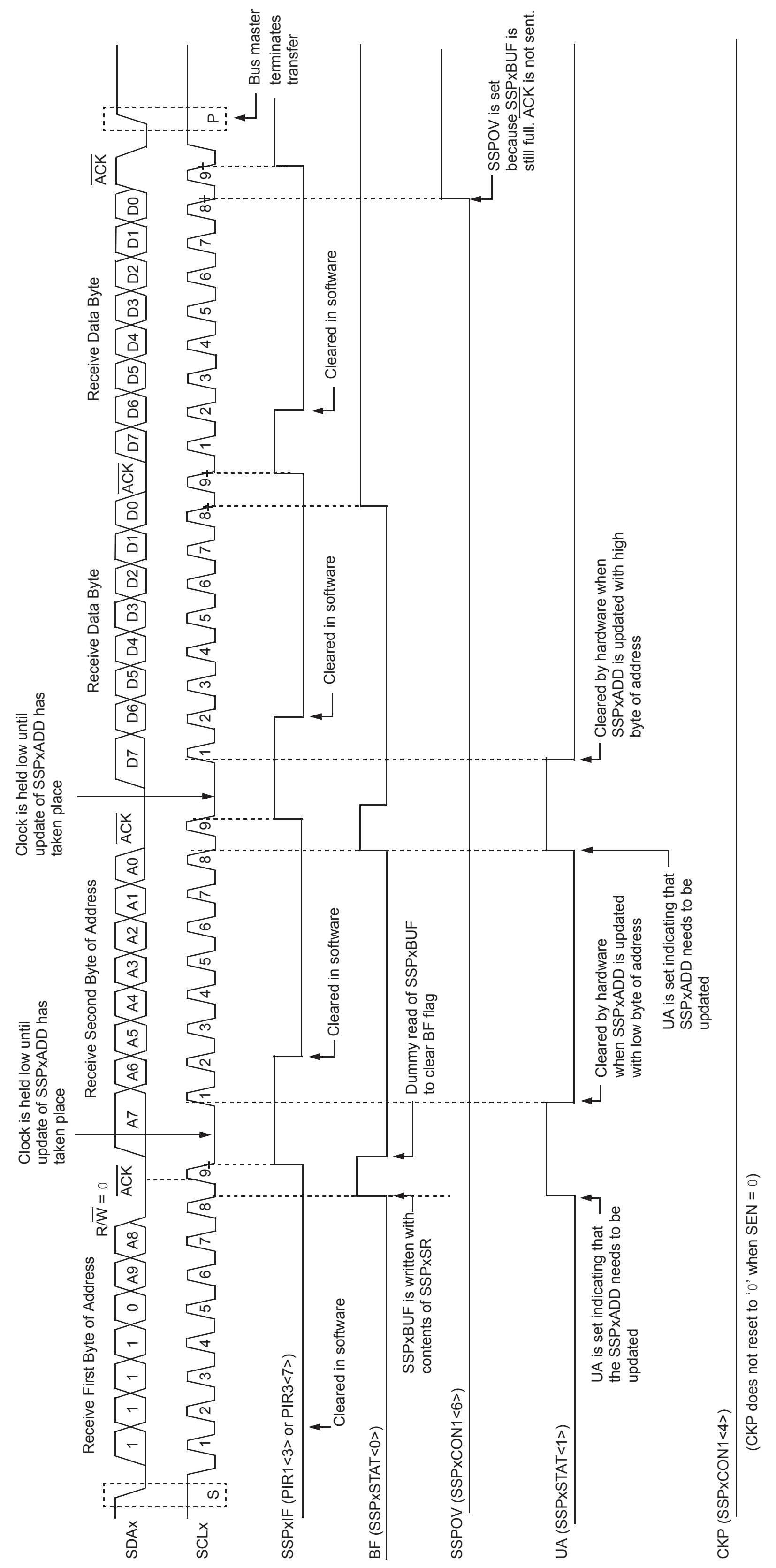
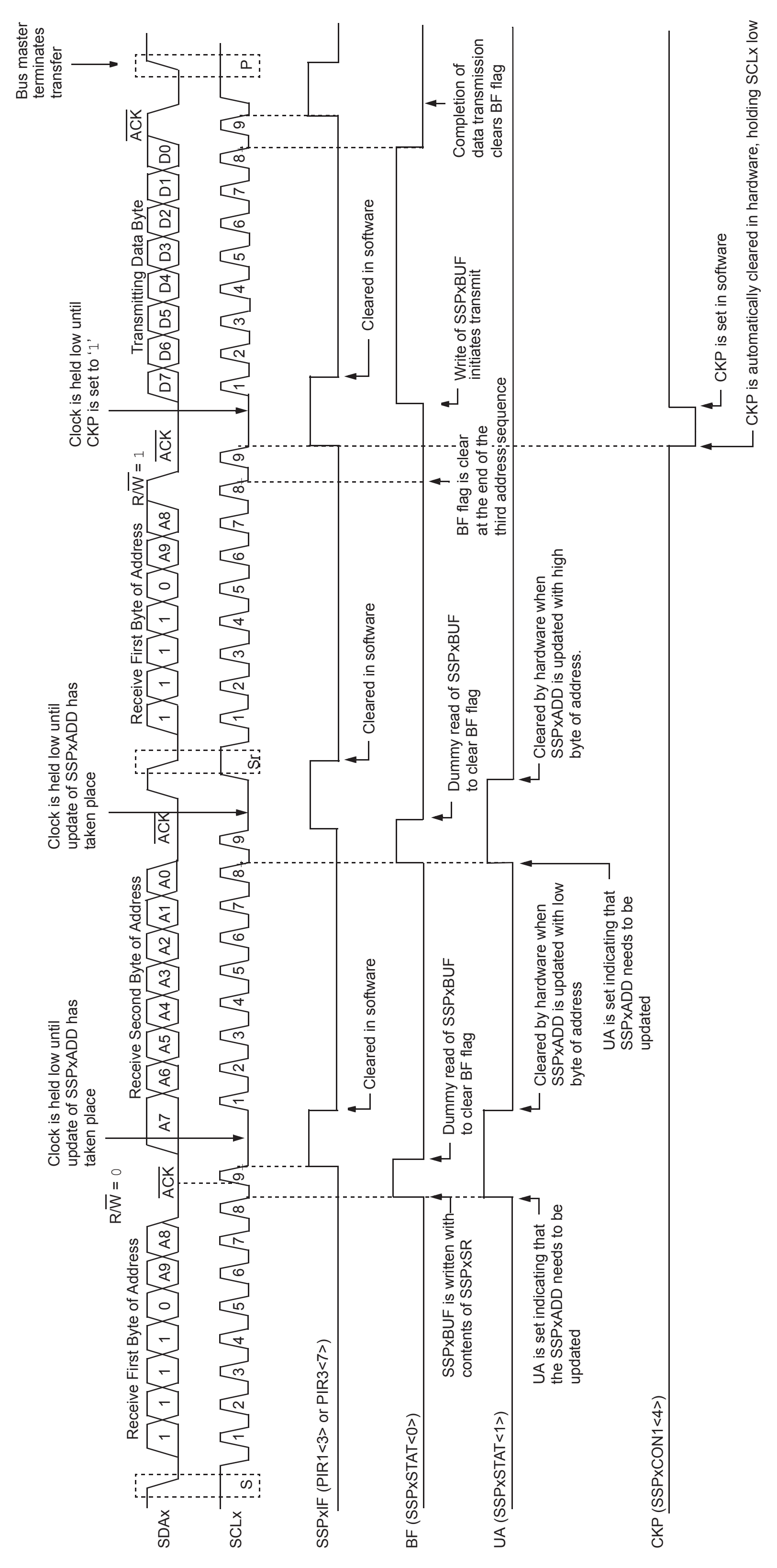

SDAx
SCLx
SSPxIF (PIR1<3> or PIR3<7>)
BF (SSPxSTAT<0>)
SSPOV (SSPxCON1<6>)
UA (SSPxSTAT<1>)
CKP (SSPxCON1<4>)
(CKP does not reset to '0' when SEN = 0)
S
Receive First Byte of Address
1
1
1
1
0
A9
A8
R/W = 0
ACK
1
2
3
4
5
6
7
8
9
Cleared in software
SSPxBUF is written with contents of SSPxSR
UA is set indicating that the SSPxADD needs to be updated
Clock is held low until update of SSPxADD has taken place
Receive Second Byte of Address
A7
A6
A5
A4
A3
A2
A1
A0
ACK
Cleared in software
Dummy read of SSPxBUF to clear BF flag
Cleared by hardware when SSPxADD is updated with low byte of address
UA is set indicating that SSPxADD needs to be updated
Clock is held low until update of SSPxADD has taken place
Receive Data Byte
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Cleared in software
Cleared by hardware when SSPxADD is updated with high byte of address
Receive Data Byte
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Cleared in software
SSPOV is set because SSPxBUF is still full. ACK is not sent.
P
Bus master terminates transfer

FIGURE 20-13: I^{2}C SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)


SDAx
SCLx
SSPxIF (PIR1<3> or PIR3<7>)
BF (SSPxSTAT<0>)
UA (SSPxSTAT<1>)
CKP (SSPxCON1<4>)
Receive First Byte of Address
1 1 1 1 0 A9 A8
R/W = 0
ACK
Receive Second Byte of Address
A7 A6 A5 A4 A3 A2 A1 A0
ACK
Receive First Byte of Address
1 1 1 1 0 A9 A8
R/W = 1
ACK
Transmitting Data Byte
D7 D6 D5 D4 D3 D2 D1 D0
ACK
S
Sr
P
1 2 3 4 5 6 7 8 9
Bus master terminates transfer
Clock is held low until update of SSPxADD has taken place
Clock is held low until update of SSPxADD has taken place
Clock is held low until CKP is set to '1'
Cleared in software
Cleared in software
Cleared in software
Cleared in software
SSPxBUF is written with contents of SSPxSR
Dummy read of SSPxBUF to clear BF flag
Dummy read of SSPxBUF to clear BF flag
BF flag is clear at the end of the third address sequence
Write of SSPxBUF initiates transmit
Completion of data transmission clears BF flag
UA is set indicating that the SSPxADD needs to be updated
Cleared by hardware when SSPxADD is updated with low byte of address
UA is set indicating that SSPxADD needs to be updated
Cleared by hardware when SSPxADD is updated with high byte of address.
CKP is set in software
CKP is automatically cleared in hardware, holding SCLx low

20.5.4 CLOCK STRETCHING

Both 7-Bit and 10-Bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPxCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCLx pin to be held low at the end of each data receive sequence.

20.5.4.1 Clock Stretching for 7-Bit Slave Receive Mode (SEN = `1`)

In 7-Bit Slave Receive mode, on the falling edge of the ninth clock at the end of the $\overline{\text{ACK}}$ sequence, if the BF bit is set, the CKP bit in the SSPxCON1 register is automatically cleared, forcing the SCLx output to be held low. The CKP bit being cleared to '`0`' will assert the SCLx line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and read the contents of the SSPxBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 20-15).

Note 1: If the user reads the contents of the SSPxBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

20.5.4.2 Clock Stretching for 10-Bit Slave Receive Mode (SEN = `1`)

In 10-Bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{\text{W}}$ bit cleared to '`0`'. The release of the clock line occurs upon updating SSPxADD. Clock stretching will occur on each data receive sequence, as described in 7-bit mode.

Note: If the user polls the UA bit and clears it by updating the SSPxADD register before the falling edge of the ninth clock occurs, and if the user has not cleared the BF bit by reading the SSPxBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

20.5.4.3 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 Interrupt Service Routine (ISR) must set the CKP bit before transmission is allowed to continue. By holding the SCLx line low, the user has time to service the ISR and load the contents of the SSPxBUF before the master device can initiate another transmit sequence (see Figure 20-10).

Note 1: If the user loads the contents of SSPxBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.

2: The CKP bit can be set in software regardless of the state of the BF bit.

20.5.4.4 Clock Stretching for 10-Bit Slave Transmit Mode

In 10-Bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-Bit Slave Receive mode. The first two addresses are followed by a third address sequence, which contains the high-order bits of the 10-bit address and the R/$\overline{\text{W}}$ bit set to '`1`'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode and clock stretching is controlled by the BF flag as in 7-Bit Slave Transmit mode (see Figure 20-13).

20.5.4.5 Clock Synchronization and CKP bit

When the CKP bit is cleared, the SCLx output is forced to '`0`'. However, clearing the CKP bit will not assert the SCLx output low until the SCLx output is already sampled low. Therefore, the CKP bit will not assert the SCLx line until an external I^2C master device has already asserted the SCLx line. The SCLx output will remain low until the CKP bit is set and all other devices on the I^2C bus have deasserted SCLx. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCLx (see Figure 20-14).

FIGURE 20-14: CLOCK SYNCHRONIZATION TIMING

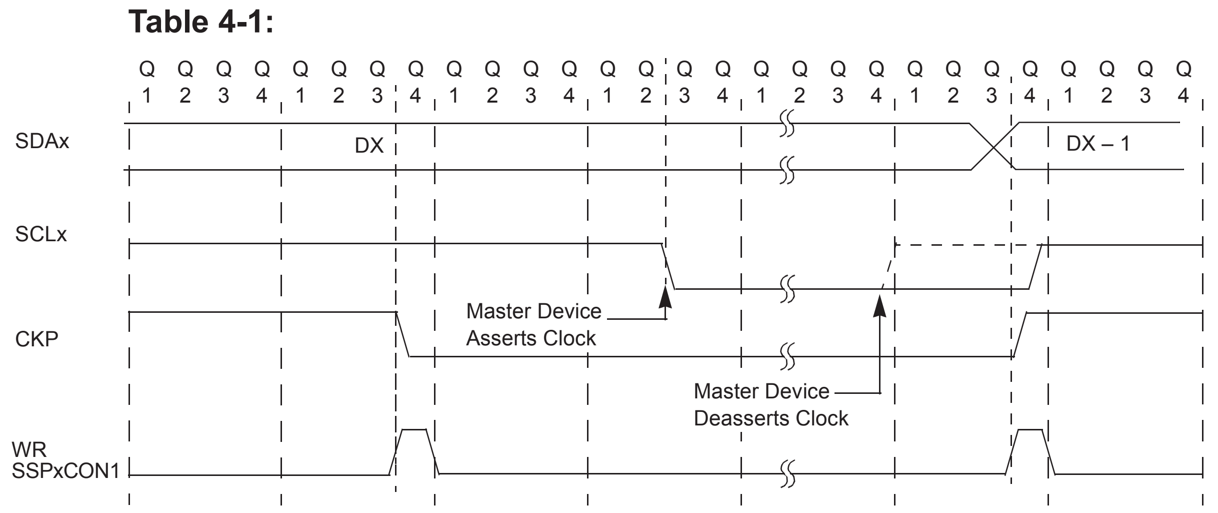


Table 4-1:
Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
SDAx
DX
DX – 1
SCLx
CKP
Master Device Asserts Clock
Master Device Deasserts Clock
WR SSPxCON1

FIGURE 20-15: I^{2}C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)

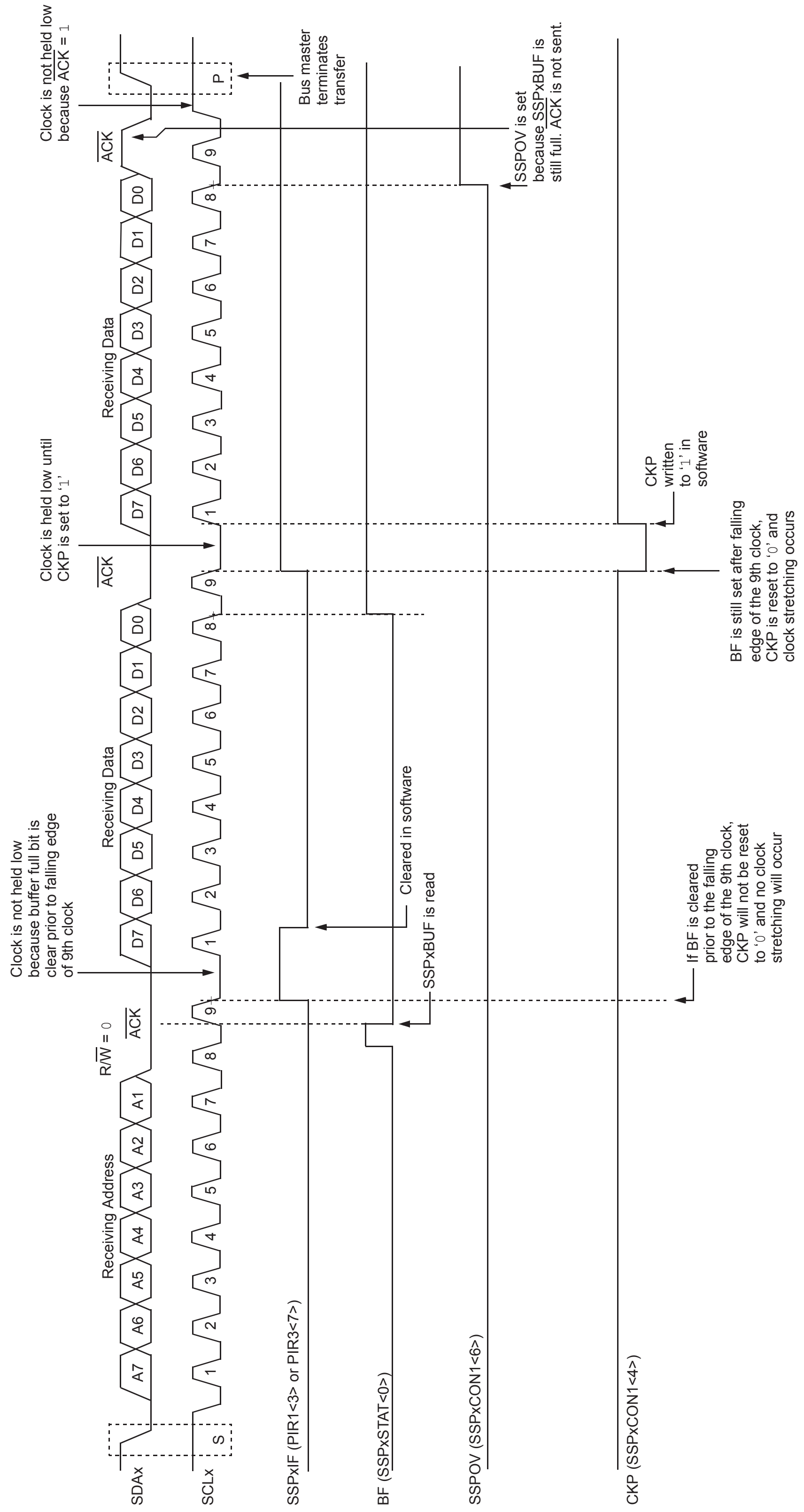
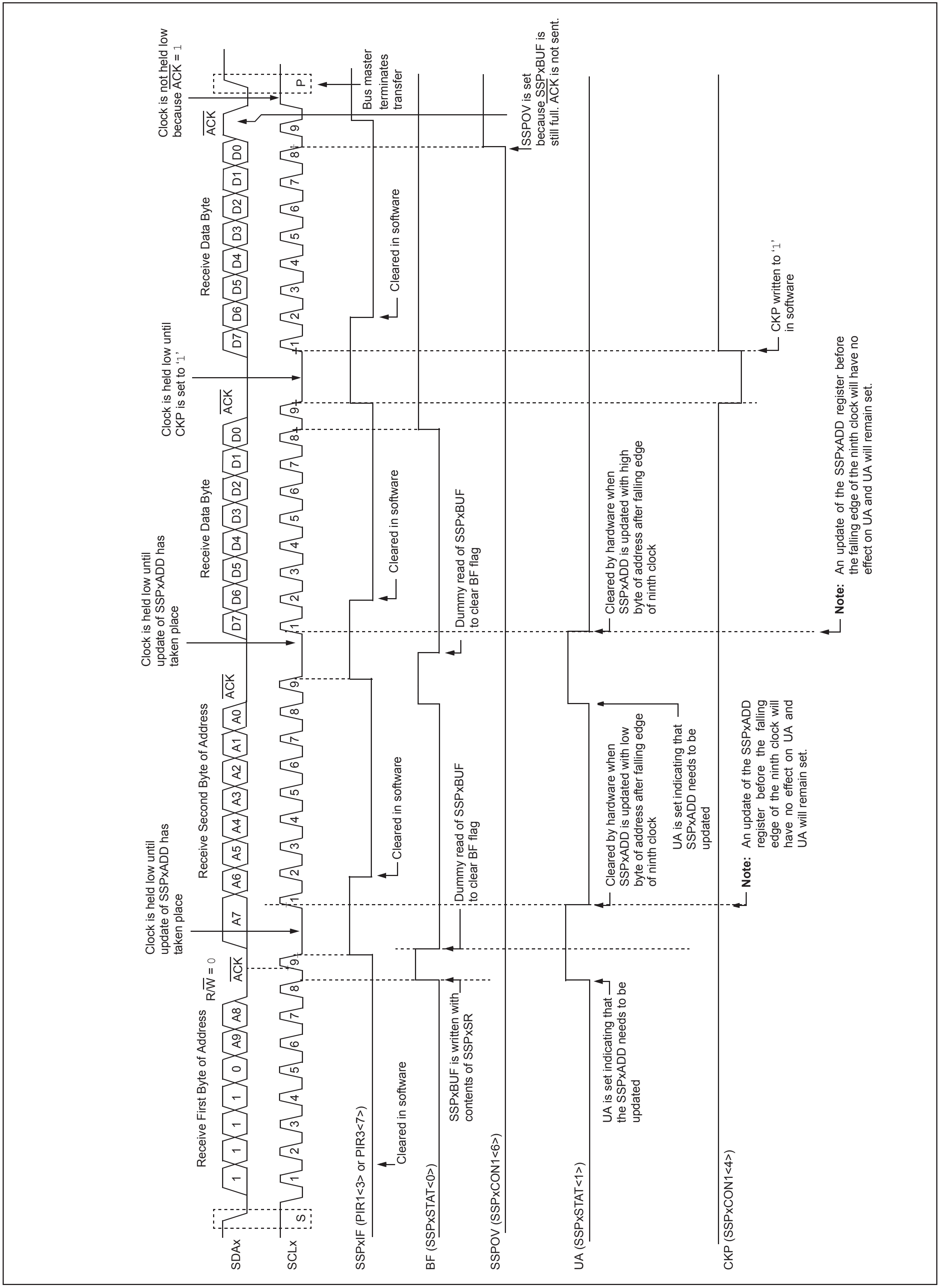

SDAx
SCLx
SSPxIF (PIR1<3> or PIR3<7>)
BF (SSPxSTAT<0>)
SSPOV (SSPxCON1<6>)
CKP (SSPxCON1<4>)
S
P
Receiving Address
A7
A6
A5
A4
A3
A2
A1
R/$\overline{W}$ = 0
$\overline{ACK}$
Receiving Data
D7
D6
D5
D4
D3
D2
D1
D0
$\overline{ACK}$
Receiving Data
D7
D6
D5
D4
D3
D2
D1
D0
$\overline{ACK}$
1
2
3
4
5
6
7
8
9
Clock is not held low because buffer full bit is clear prior to falling edge of 9th clock
Clock is held low until CKP is set to '1'
Clock is not held low because $\overline{ACK}$ = 1
SSPxBUF is read
Cleared in software
Bus master terminates transfer
SSPOV is set because SSPxBUF is still full. ACK is not sent.
If BF is cleared prior to the falling edge of the 9th clock, CKP will not be reset to '0' and no clock stretching will occur
BF is still set after falling edge of the 9th clock, CKP is reset to '0' and clock stretching occurs
CKP written to '1' in software

FIGURE 20-16: I²C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESS)


SDAx
SCLx
SSPxIF (PIR1<3> or PIR3<7>)
BF (SSPxSTAT<0>)
SSPOV (SSPxCON1<6>)
UA (SSPxSTAT<1>)
CKP (SSPxCON1<4>)
S
P
Receive First Byte of Address
1 1 1 1 0 A9 A8
R/W = 0
ACK
1 2 3 4 5 6 7 8 9
Receive Second Byte of Address
A7 A6 A5 A4 A3 A2 A1 A0
ACK
Receive Data Byte
D7 D6 D5 D4 D3 D2 D1 D0
ACK
Receive Data Byte
D7 D6 D5 D4 D3 D2 D1 D0
ACK
Clock is held low until update of SSPxADD has taken place
Clock is held low until update of SSPxADD has taken place
Clock is held low until CKP is set to '1'
Clock is not held low because ACK = 1
Bus master terminates transfer
Cleared in software
Cleared in software
Cleared in software
Cleared in software
SSPxBUF is written with contents of SSPxSR
Dummy read of SSPxBUF to clear BF flag
Dummy read of SSPxBUF to clear BF flag
SSPOV is set because SSPxBUF is still full. ACK is not sent.
UA is set indicating that the SSPxADD needs to be updated
Cleared by hardware when SSPxADD is updated with low byte of address after falling edge of ninth clock
UA is set indicating that SSPxADD needs to be updated
Cleared by hardware when SSPxADD is updated with high byte of address after falling edge of ninth clock
Note: An update of the SSPxADD register before the falling edge of the ninth clock will have no effect on UA and UA will remain set.
Note: An update of the SSPxADD register before the falling edge of the ninth clock will have no effect on UA and UA will remain set.
CKP written to '1' in software

20.5.5 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`.

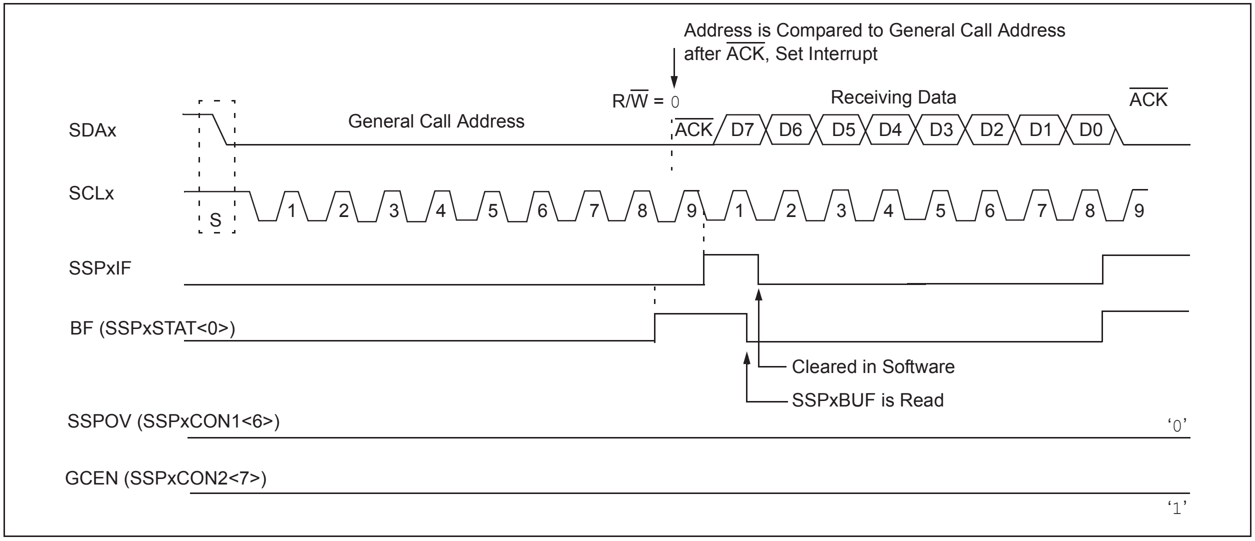
The general call address is recognized when the General Call Enable bit, GCEN, is enabled (SSPxCON2<7> set). Following a Start bit detect, 8 bits are shifted into the SSPxSR and the address is compared against the SSPxADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPxSR is transferred to the SSPxBUF, the BF flag bit is set (eighth bit), and on the falling edge of the ninth bit ($\overline{ACK}$ bit), the SSPxIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPxBUF. The value can be used to determine if the address was device-specific or a general call address.

In 10-bit mode, the SSPxADD is required to be updated for the second half of the address to match and the UA bit is set (SSPxSTAT<1>). If the general call address is sampled when the GCEN bit is set, while the slave is configured in 10-Bit Addressing mode, then the second half of the address is not necessary. The UA bit will not be set and the slave will begin receiving data after the Acknowledge (Figure 20-17).

FIGURE 20-17: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7-BIT OR 10-BIT ADDRESSING MODE)



20.5.6 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 TRISx bits are set.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Start (S) and Stop (P) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^2C bus may be taken when the Stop bit is set, or the bus is Idle, with both the Start and Stop 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.

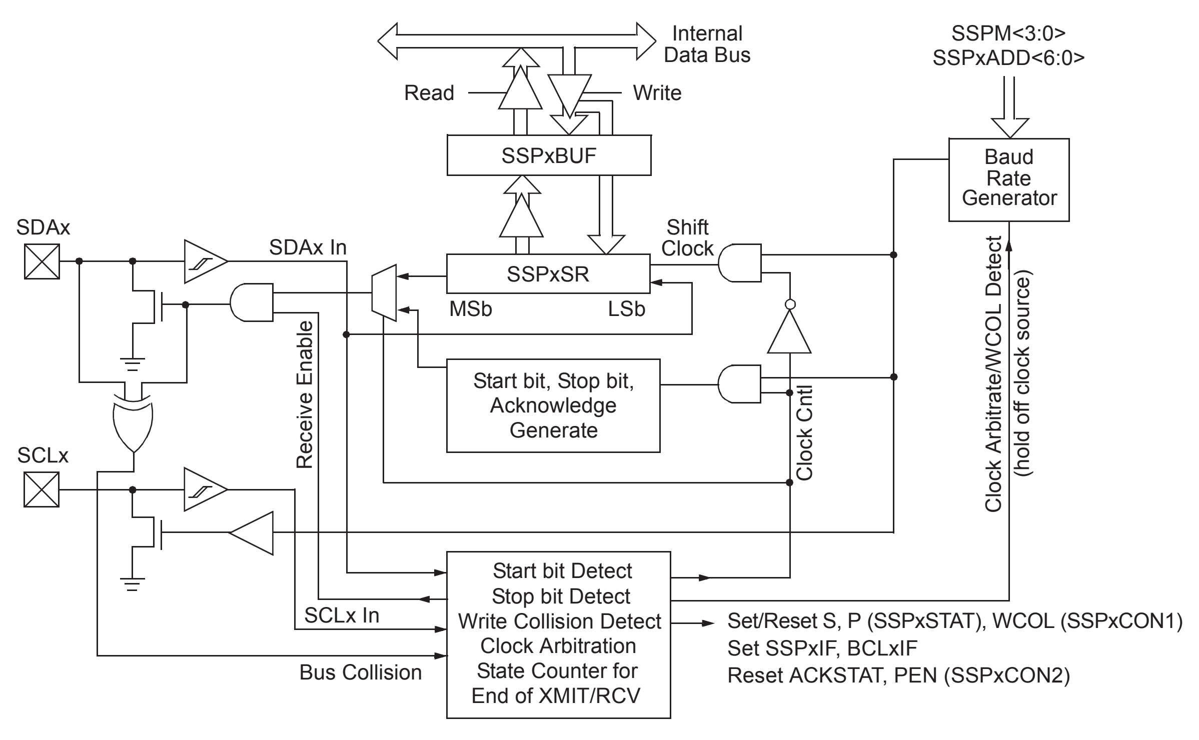
1. Assert a Start condition on SDAx and SCLx.
2. Assert a Repeated Start condition on SDAx and SCLx.
3. Write to the SSPxBUF register initiating transmission of data/address.
4. Configure the I^2C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDAx and SCLx.

Note: The MSSP module, when configured in I²C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur.

The following events will cause the MSSP Interrupt Flag bit, SSPxIF, to be set (and MSSP interrupt, if enabled):

- Start condition
- Stop condition
- Data transfer byte transmitted/received
- Acknowledge transmitted
- Repeated Start

FIGURE 20-18: MSSPx BLOCK DIAGRAM (I²C MASTER MODE)


Internal
Data Bus
SSPM<3:0>
SSPxADD<6:0>
Read
Write
SSPxBUF
Baud
Rate
Generator
SDAx
SDAx In
Shift
Clock
SSPxSR
MSb
LSb
Receive Enable
Start bit, Stop bit,
Acknowledge
Generate
Clock Cntl
Clock Arbitrate/WCOL Detect
(hold off clock source)
SCLx
SCLx In
Bus Collision
Start bit Detect
Stop bit Detect
Write Collision Detect
Clock Arbitration
State Counter for
End of XMIT/RCV
Set/Reset S, P (SSPxSTAT), WCOL (SSPxCON1)
Set SSPxIF, BCLxIF
Reset ACKSTAT, PEN (SSPxCON2)

20.5.6.1 I^{2}C Master Mode Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I^{2}C bus will not be released.

In Master Transmitter mode, serial data is output through SDAx while SCLx outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/$\overline{\text{Write}}$ (R/$\overline{\text{W}}$) bit. In this case, the R/$\overline{\text{W}}$ bit will be logic `0`. Serial data is transmitted, 8 bits at a time. After each byte is transmitted, an Acknowledge bit is received. S and P conditions are output to indicate the beginning and 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/$\overline{\text{W}}$ bit. In this case, the R/$\overline{\text{W}}$ 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. S and P conditions indicate the beginning and end of transmission.

The BRG used for the SPI mode operation is used to set the SCLx clock frequency for either 100 kHz, 400 kHz or 1 MHz I^{2}C operation. See **Section 20.5.7 "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 for the required start time before any other operation takes place.
3. The user loads the SSPxBUF with the slave address to transmit.
4. The address is shifted out of 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 8 bits of data.
8. Data is shifted out of 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. The interrupt is generated once the Stop condition is complete.

20.5.7 BAUD RATE

In I^{2}C Master mode, the BRG reload value is placed in the lower seven bits of the SSPxADD register (Figure 20-19). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (TCY) on the Q2 and Q4 clocks. In I^{2}C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by $\overline{\text{ACK}}$), the internal clock will automatically stop counting and the SCLx pin will remain in its last state.

Table 20-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD. The SSPADD BRG value of 0x00 is not supported.

20.5.7.1 Baud Rate and Module Interdependence

Because MSSP1 and MSSP2 are independent, they can operate simultaneously in I^{2}C Master mode at different baud rates. This is done by using different BRG reload values for each module.

Because this mode derives its basic clock source from the system clock, any changes to the clock will affect both modules in the same proportion. It may be possible to change one or both baud rates back to a previous value by changing the BRG reload value.

FIGURE 20-19: BAUD RATE GENERATOR BLOCK DIAGRAM

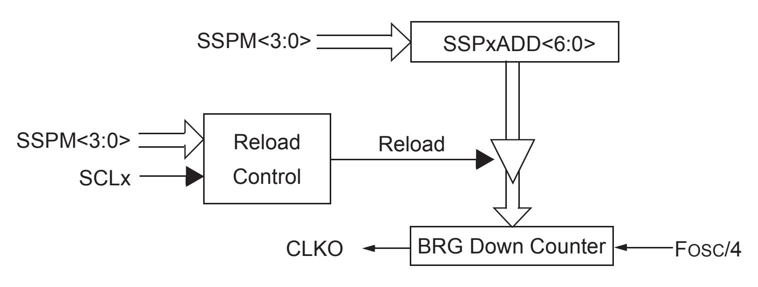


TABLE 20-3: I^{2}C CLOCK RATE w/BRG

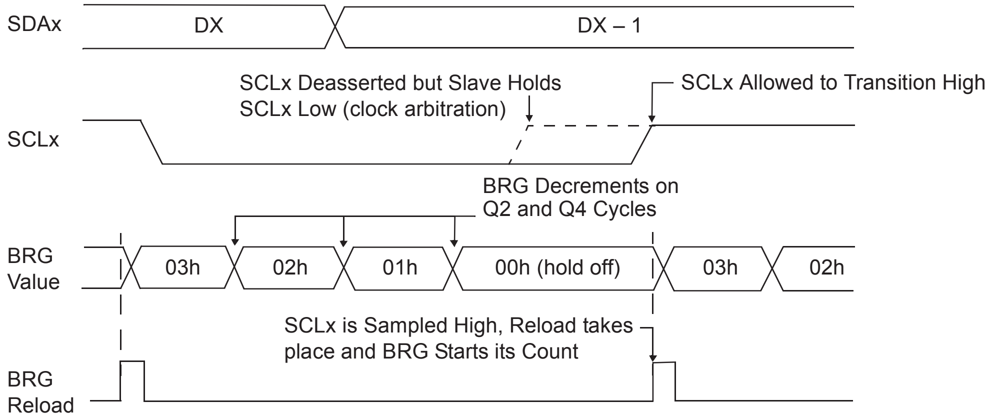
FOSC	FCY	FCY * 2	BRG Value	FSCL (2 Rollovers of BRG)
40 MHz	10 MHz	20 MHz	18h	400 kHz
40 MHz	10 MHz	20 MHz	1Fh	312.5 kHz
40 MHz	10 MHz	20 MHz	63h	100 kHz
16 MHz	4 MHz	8 MHz	09h	400 kHz
16 MHz	4 MHz	8 MHz	0Ch	308 kHz
16 MHz	4 MHz	8 MHz	27h	100 kHz
4 MHz	1 MHz	2 MHz	02h	333 kHz
4 MHz	1 MHz	2 MHz	09h	100 kHz
16 MHz	4 MHz	8 MHz	03h	1 MHz(1)

Note 1: The I^{2}C interface does not conform to the 400 kHz I^{2}C specification (which applies to rates greater than 100 kHz) in all details, but may be used with care where higher rates are required by the application.

20.5.7.2 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 BRG is suspended from counting until the SCLx pin is actually sampled high. When the SCLx pin is sampled high, the BRG is reloaded with the contents of SSPxADD<6:0> and begins counting. This ensures that the SCLx high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 20-20).

FIGURE 20-20: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION


SDAx
DX
DX – 1
SCLx Deasserted but Slave Holds
SCLx Low (clock arbitration)
SCLx Allowed to Transition High
SCLx
BRG Decrements on
Q2 and Q4 Cycles
BRG
Value
03h
02h
01h
00h (hold off)
03h
02h
SCLx is Sampled High, Reload takes
place and BRG Starts its Count
BRG
Reload

20.5.8 I^{2}C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Enable bit, SEN (SSPxCON2<0>). If the SDAx and SCLx pins are sampled high, the BRG 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 Start bit (SSPxSTAT<3>) to be set. Following this, the BRG is reloaded with the contents of SSPxADD<6:0> and resumes its count. When the BRG times out (TBRG), the SEN bit (SSPxCON2<0>) will be automatically cleared by hardware. The BRG is suspended, leaving the SDAx line held low and the Start condition is complete.

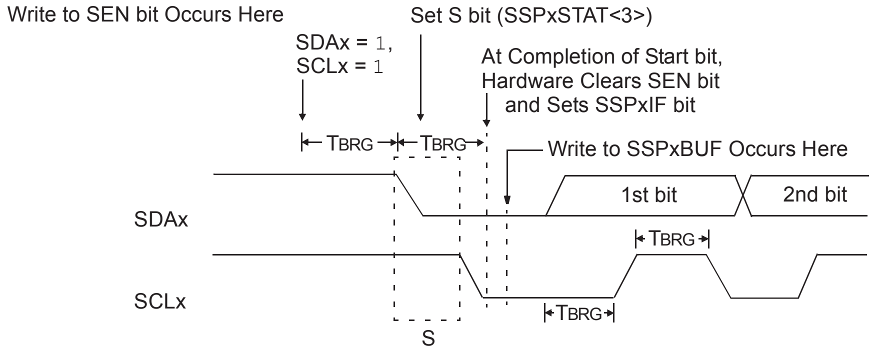
Note:	If, at the beginning of the Start condition, the SDAx and SCLx pins are already sampled low, or if during the Start condition, the SCLx line is sampled low before the SDAx line is driven low, a bus collision occurs. The Bus Collision Interrupt Flag, BCLxIF, is set, the Start condition is aborted and the I^{2}C module is reset into its Idle state.

20.5.8.1 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 does not occur).

Note:	Because queueing of events is not allowed, writing to the lower five bits of SSPxCON2 is disabled until the Start condition is complete.

FIGURE 20-21: FIRST START BIT TIMING


Write to SEN bit Occurs Here
Set S bit (SSPxSTAT<3>)
SDAx = 1,
SCLx = 1
At Completion of Start bit,
Hardware Clears SEN bit
and Sets SSPxIF bit
TBRG
TBRG
Write to SSPxBUF Occurs Here
1st bit
2nd bit
SDAx
TBRG
SCLx
TBRG
S

20.5.9 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 BRG is loaded with the contents of SSPxADD<5:0> and begins counting. The SDAx pin is released (brought high) for one BRG count (TBRG). When the BRG times out, and if SDAx is sampled high, the SCLx pin will be deasserted (brought high). When SCLx is sampled high, the BRG 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 BRG 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 Start bit (SSPxSTAT<3>) will be set. The SSPxIF bit will not be set until the BRG 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`’.

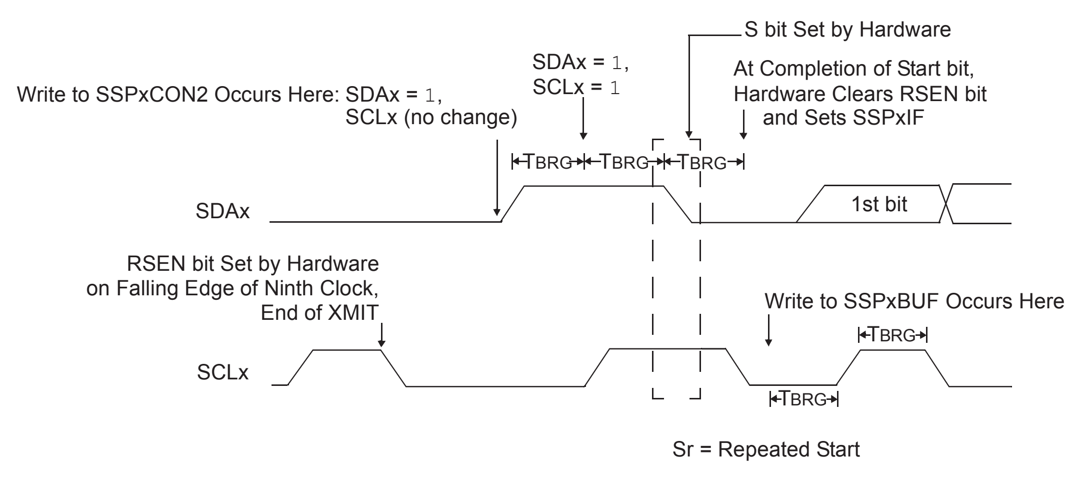
Immediately following the SSPxIF bit getting set, the user may write the SSPxBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional 8 bits of address (10-bit mode) or 8 bits of data (7-bit mode).

20.5.9.1 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 does not occur).

Note: Because queueing of events is not allowed, writing of the lower five bits of SSPxCON2 is disabled until the Repeated Start condition is complete.

FIGURE 20-22: REPEATED START CONDITION WAVEFORM


S bit Set by Hardware
SDAx = 1,
SCLx = 1
At Completion of Start bit,
Hardware Clears RSEN bit
and Sets SSPxIF
Write to SSPxCON2 Occurs Here: SDAx = 1,
SCLx (no change)
TBRG
TBRG
TBRG
SDAx
1st bit
RSEN bit Set by Hardware
on Falling Edge of Ninth Clock,
End of XMIT
Write to SSPxBUF Occurs Here
TBRG
SCLx
TBRG
Sr = Repeated Start

20.5.10 I^{2}C 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 BRG 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 BRG 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 $\overline{\text{ACK}}$ bit during the ninth bit time if an address match occurred or if data was received properly. The status of $\overline{\text{ACK}}$ 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 (BRG) is suspended until the next data byte is loaded into the SSPxBUF, leaving SCLx low and SDAx unchanged (Figure 20-23).

After the write to the SSPxBUF, each bit of the address will be shifted out on the falling edge of SCLx until all seven address bits and the R/$\overline{\text{W}}$ 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 $\overline{\text{ACK}}$ 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 BRG is turned off until another write to the SSPxBUF takes place, holding SCLx low and allowing SDAx to float.

20.5.10.1 BF Status Flag

In Transmit mode, the BF bit (SSPxSTAT<0>) is set when the CPU writes to SSPxBUF and is cleared when all eight bits are shifted out.

20.5.10.2 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 does not occur) after 2 TCY after the SSPxBUF write. If SSPxBUF is rewritten within 2 TCY, the WCOL bit is set and SSPxBUF is updated. This may result in a corrupted transfer.

The user should verify that the WCOL bit is clear after each write to SSPxBUF to ensure the transfer is correct. In all cases, WCOL must be cleared in software.

20.5.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPxCON2<6>) is cleared when the slave has sent an Acknowledge ($\overline{\text{ACK}}$ = `0`) and is set when the slave does not Acknowledge ($\overline{\text{ACK}}$ = `1`). A slave sends an Acknowledge when it has recognized its address (including a general call) or when the slave has properly received its data.

20.5.11 I^{2}C 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 BRG 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 BRG is suspended from counting, holding SCLx low. The MSSP is now in an Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable bit, ACKEN (SSPxCON2<4>).

20.5.11.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPxBUF from SSPxSR. It is cleared when the SSPxBUF register is read.

20.5.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPxSR and the BF flag bit is already set from a previous reception.

20.5.11.3 WCOL Status Flag

If users write the SSPxBUF when a receive is already in progress (i.e., SSPxSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

FIGURE 20-23: I²C MASTER MODE WAVEFORM (TRANSMISSION, 7-BIT OR 10-BIT ADDRESS)

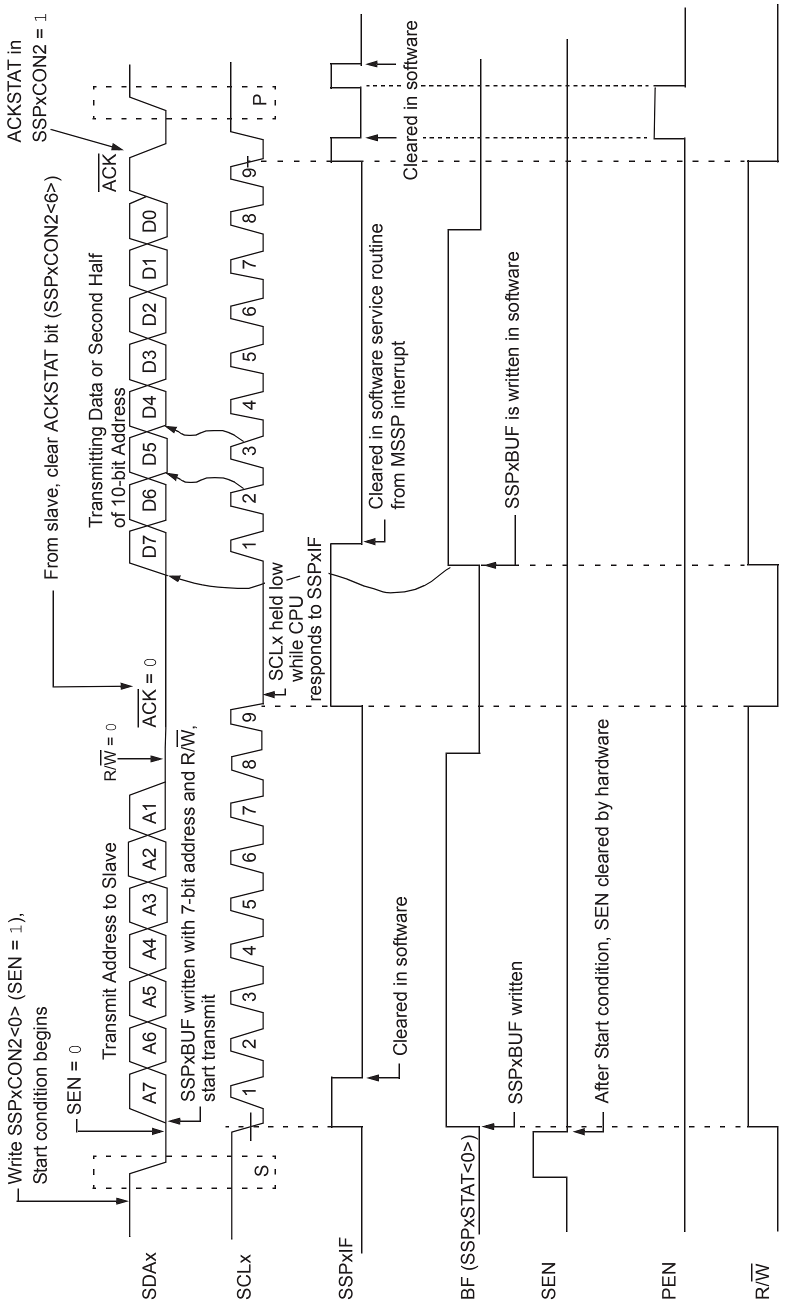
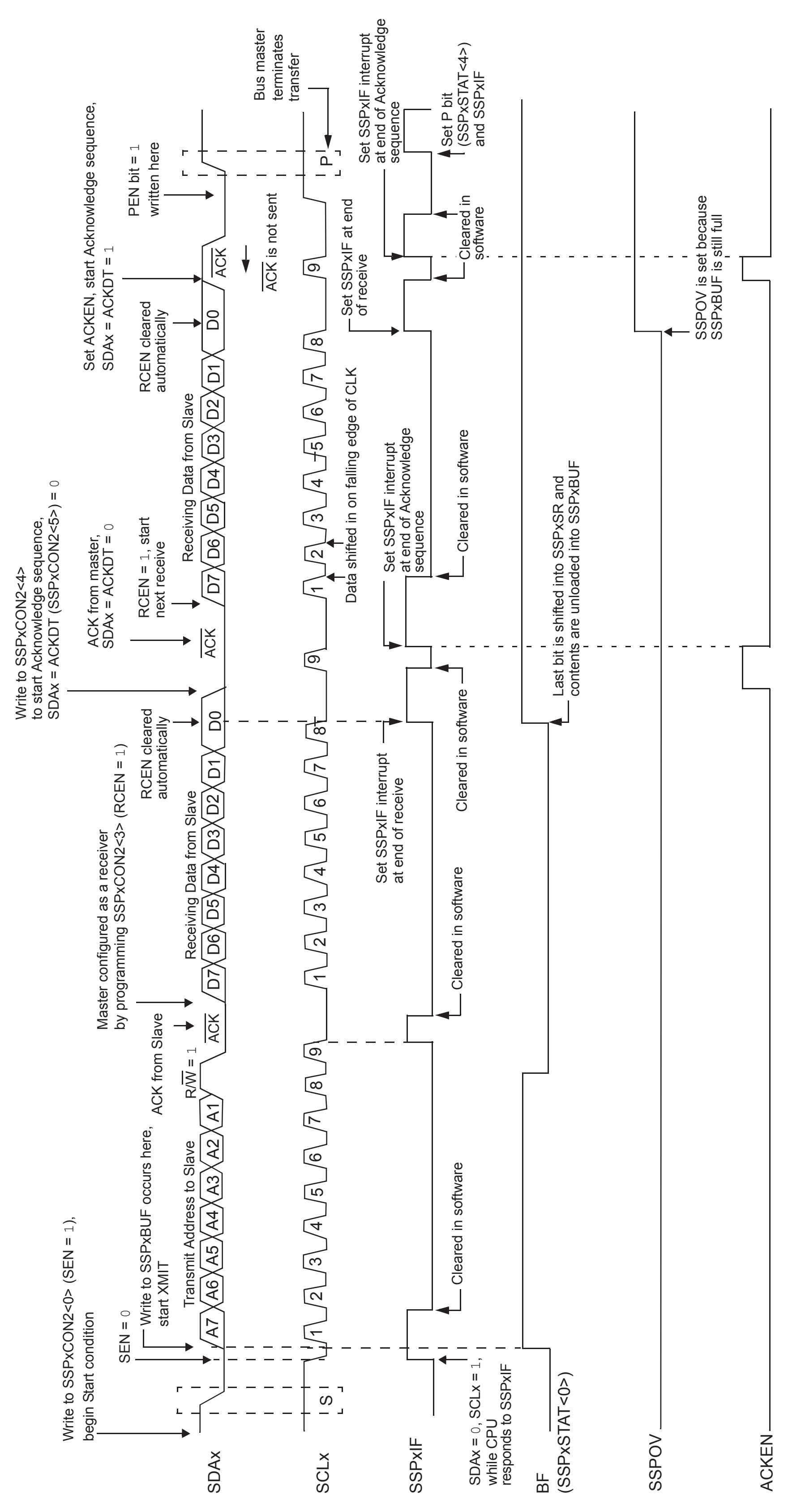

SDAx
SCLx
SSPxIF
BF (SSPxSTAT<0>)
SEN
PEN
R/W̄
Write SSPxCON2<0> (SEN = 1),
Start condition begins
SEN = 0
Transmit Address to Slave
A7
A6
A5
A4
A3
A2
A1
R/W̄ = 0
ĀCK̄ = 0
SSPxBUF written with 7-bit address and R/W̄,
start transmit
S
1
2
3
4
5
6
7
8
9
Cleared in software
SSPxBUF written
After Start condition, SEN cleared by hardware
From slave, clear ACKSTAT bit (SSPxCON2<6>)
Transmitting Data or Second Half
of 10-bit Address
D7
D6
D5
D4
D3
D2
D1
D0
ĀCK̄
ACKSTAT in
SSPxCON2 = 1
P
SCLx held low
while CPU
responds to SSPxIF
Cleared in software service routine
from MSSP interrupt
SSPxBUF is written in software
Cleared in software

FIGURE 20-24: I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)


SDAx
SCLx
SSPxIF
SDAx = 0, SCLx = 1, while CPU responds to SSPxIF
BF (SSPxSTAT<0>)
SSPOV
ACKEN
Write to SSPxCON2<0> (SEN = 1), begin Start condition
SEN = 0
Write to SSPxBUF occurs here, start XMIT
Transmit Address to Slave
ACK from Slave
R/W̄ = 1
Master configured as a receiver by programming SSPxCON2<3> (RCEN = 1)
Receiving Data from Slave
RCEN cleared automatically
Write to SSPxCON2<4> to start Acknowledge sequence, SDAx = ACKDT (SSPxCON2<5>) = 0
ACK from master, SDAx = ACKDT = 0
RCEN = 1, start next receive
Set ACKEN, start Acknowledge sequence, SDAx = ACKDT = 1
PEN bit = 1 written here
Bus master terminates transfer
ACK is not sent
S
P
A7 A6 A5 A4 A3 A2 A1
ACK
D7 D6 D5 D4 D3 D2 D1 D0
1 2 3 4 5 6 7 8 9
Cleared in software
Set SSPxIF interrupt at end of receive
Set SSPxIF interrupt at end of Acknowledge sequence
Data shifted in on falling edge of CLK
Set SSPxIF at end of receive
Set SSPxIF interrupt at end of Acknowledge sequence
Set P bit (SSPxSTAT<4>) and SSPxIF
Last bit is shifted into SSPxSR and contents are unloaded into SSPxBUF
SSPOV is set because SSPxBUF is still full

20.5.12 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 BRG 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 BRG counts for TBRG; the SCLx pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the BRG is turned off and the MSSP module then goes into an inactive state (Figure 20-25).

20.5.12.1 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 does not occur).

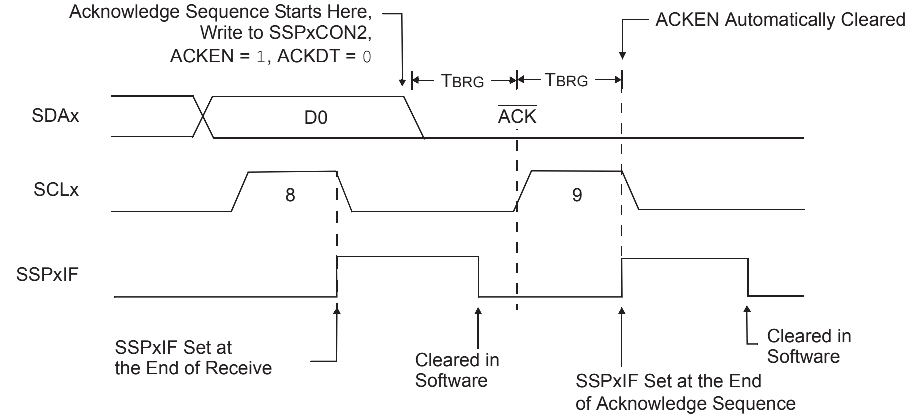
20.5.13 STOP CONDITION TIMING

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 BRG is reloaded and counts down to 0. When the BRG times out, the SCLx pin will be brought high and one Baud Rate Generator rollover count (TBRG) later, the SDAx pin will be deasserted. When the SDAx pin is sampled high while SCLx is high, the Stop bit (SSPxSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPxIF bit is set (Figure 20-26).

20.5.13.1 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 does not occur).

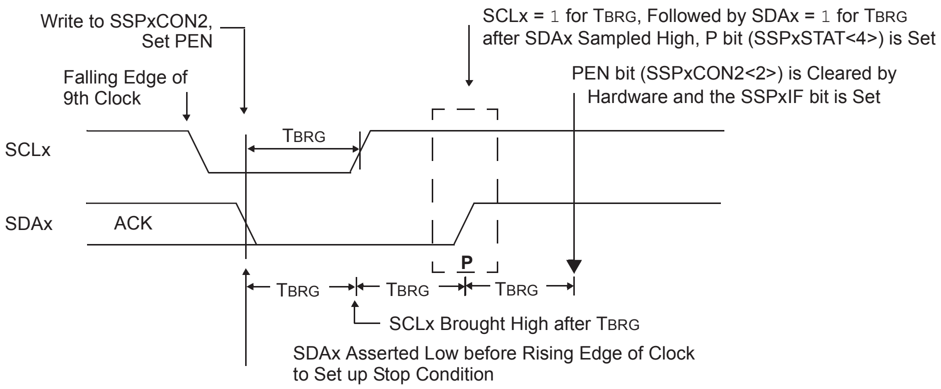
FIGURE 20-25: ACKNOWLEDGE SEQUENCE WAVEFORM



Acknowledge Sequence Starts Here,
Write to SSPxCON2,
ACKEN = 1, ACKDT = 0
ACKEN Automatically Cleared
TBRG
TBRG
SDAx
D0
ACK
SCLx
8
9
SSPxIF
SSPxIF Set at the End of Receive
Cleared in Software
SSPxIF Set at the End of Acknowledge Sequence
Cleared in Software

Note: TBRG = one Baud Rate Generator period.

FIGURE 20-26: STOP CONDITION RECEIVE OR TRANSMIT MODE



Write to SSPxCON2, Set PEN
Falling Edge of 9th Clock
SCLx = 1 for TBRG, Followed by SDAx = 1 for TBRG after SDAx Sampled High, P bit (SSPxSTAT<4>) is Set
PEN bit (SSPxCON2<2>) is Cleared by Hardware and the SSPxIF bit is Set
SCLx
TBRG
SDAx
ACK
P
TBRG
TBRG
TBRG
SCLx Brought High after TBRG
SDAx Asserted Low before Rising Edge of Clock to Set up Stop Condition

Note: TBRG = one Baud Rate Generator period.

20.5.14 SLEEP OPERATION

While in Sleep mode, the I^{2}C module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

20.5.15 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

20.5.16 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 Start and Stop bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^{2}C bus may be taken when the P bit (SSPxSTAT<4>) is set, or the bus is Idle, with both the Start and Stop bits clear. When the bus is busy, enabling the MSSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDAx line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed in hardware with the result placed in the BCLxIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

20.5.17 MULTI -MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDAx pin, arbitration takes place when the master outputs a '`1`' on SDAx, by letting SDAx float high, and another master asserts a '`0`'. When the SCLx pin floats high, data should be stable. If the expected data on SDAx is a '`1`' and the data sampled on the SDAx pin = `0`, then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF, and reset the I^{2}C port to its Idle state (Figure 20-27).

If a transmit was in progress when the bus collision occurred, the transmission is Halted, the BF flag is cleared, the SDAx and SCLx lines are deasserted and the SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine, and if the I^{2}C bus is free, the user can resume communication by asserting a Start condition.

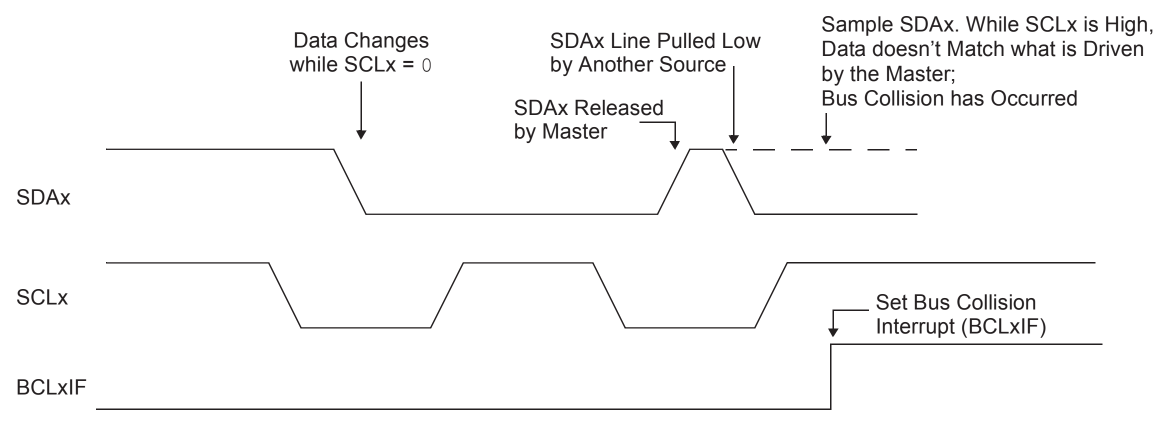
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDAx and SCLx lines are deasserted, and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine (ISR), and if the I^{2}C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDAx and SCLx pins. If a Stop condition occurs, the SSPxIF bit will be set.

A write to the SSPxBUF will start the transmission of data at the first data bit regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I^{2}C bus can be taken when the Stop bit is set in the SSPxSTAT register, or the bus is Idle and the Start and Stop bits are cleared.

FIGURE 20-27: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE


Data Changes while SCLx = 0
SDAx Line Pulled Low by Another Source
Sample SDAx. While SCLx is High, Data doesn't Match what is Driven by the Master; Bus Collision has Occurred
SDAx Released by Master
SDAx
SCLx
Set Bus Collision Interrupt (BCLxIF)
BCLxIF

20.5.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDAx or SCLx is sampled low at the beginning of the Start condition (Figure 20-28).

b) SCLx is sampled low before SDAx is asserted low (Figure 20-29).

During a Start condition, both the SDAx and the SCLx pins are monitored.

If the SDAx pin is already low, or the SCLx pin is already low, then all of the following occur:

- The Start condition is aborted
- The BCLxIF flag is set
- The MSSP module is reset to its inactive state (Figure 20-28)

The Start condition begins with the SDAx and SCLx pins deasserted. When the SDAx pin is sampled high, the BRG is loaded from SSPxADD<6:0> and counts down to 0. If the SCLx pin is sampled low while SDAx is high, a bus collision occurs because it is assumed that another master is attempting to drive a data ‘1’ during the Start condition.

If the SDAx pin is sampled low during this count, the BRG is reset and the SDAx line is asserted early (Figure 20-30). If, however, a ‘1’ is sampled on the SDAx pin, the SDAx pin is asserted low at the end of the BRG count. The BRG 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 20-28: BUS COLLISION DURING START CONDITION (SDAx ONLY)

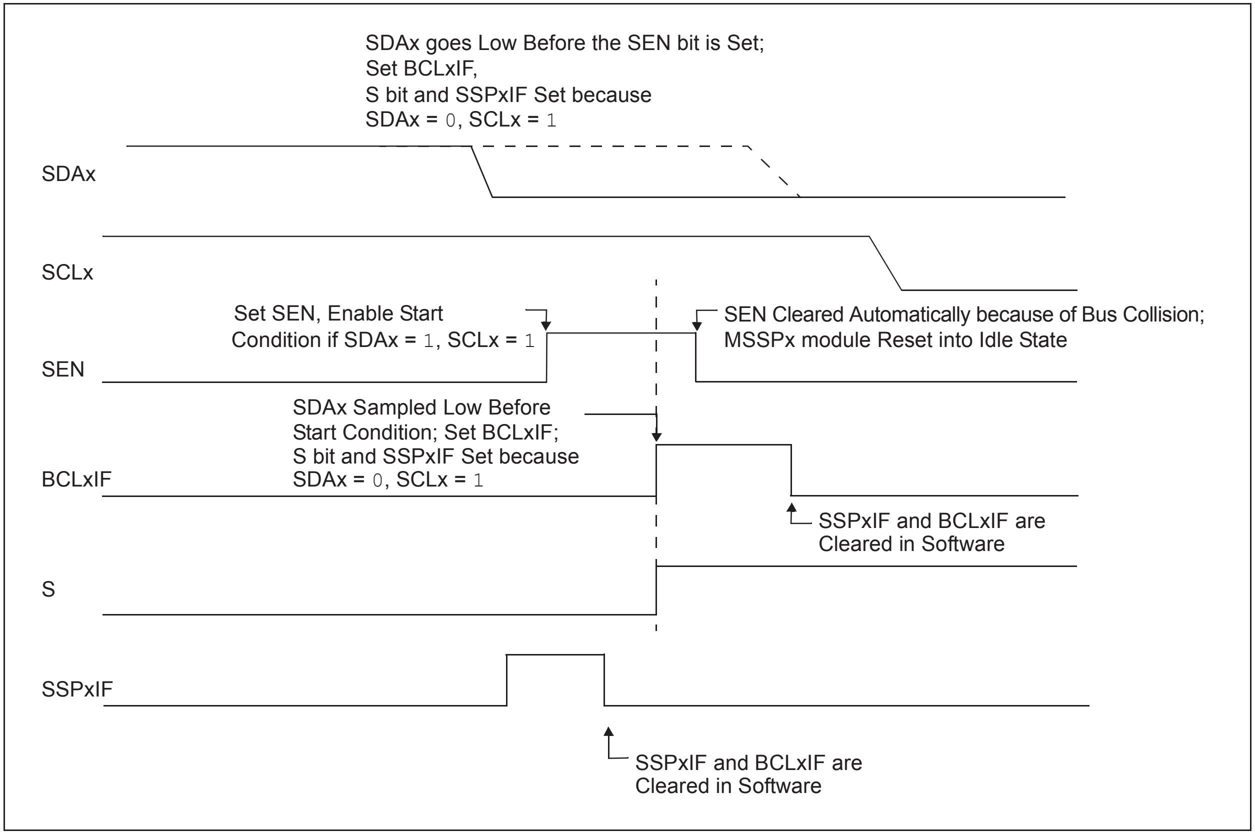

SDAx goes Low Before the SEN bit is Set;
Set BCLxIF,
S bit and SSPxIF Set because
SDAx = 0, SCLx = 1
SDAx
SCLx
Set SEN, Enable Start
Condition if SDAx = 1, SCLx = 1
SEN Cleared Automatically because of Bus Collision;
MSSPx module Reset into Idle State
SEN
SDAx Sampled Low Before
Start Condition; Set BCLxIF;
S bit and SSPxIF Set because
SDAx = 0, SCLx = 1
BCLxIF
SSPxIF and BCLxIF are
Cleared in Software
S
SSPxIF
SSPxIF and BCLxIF are
Cleared in Software

FIGURE 20-29: BUS COLLISION DURING START CONDITION (SCLx = 0)

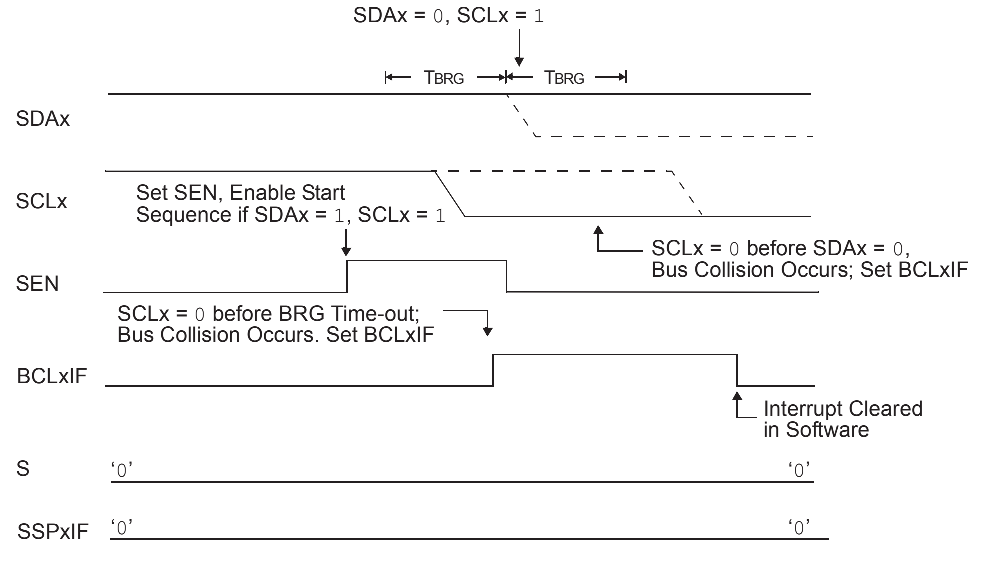
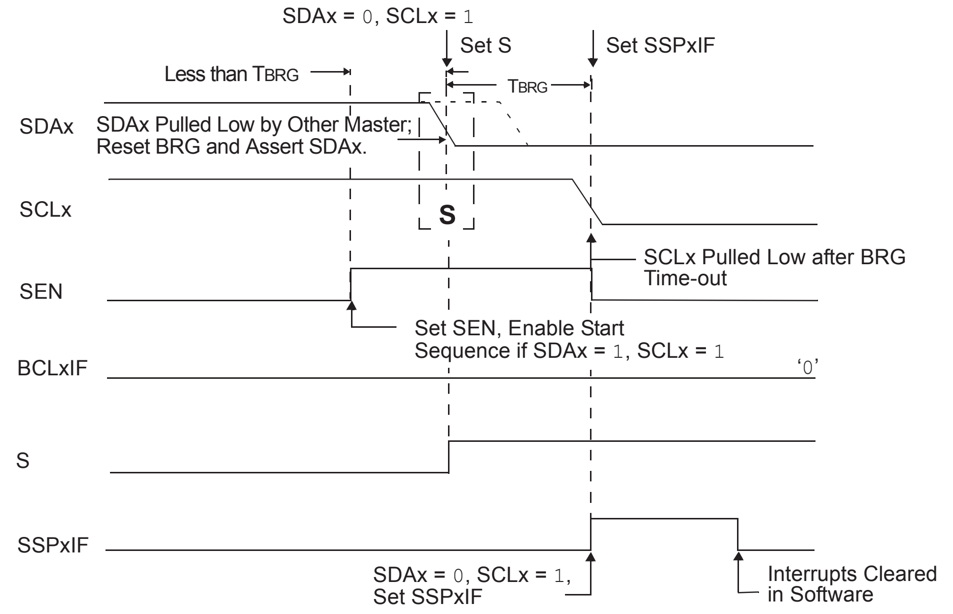

SDAx = 0, SCLx = 1
TBRG
TBRG
SDAx
SCLx
Set SEN, Enable Start Sequence if SDAx = 1, SCLx = 1
SCLx = 0 before SDAx = 0, Bus Collision Occurs; Set BCLxIF
SEN
SCLx = 0 before BRG Time-out; Bus Collision Occurs. Set BCLxIF
BCLxIF
Interrupt Cleared in Software
S
'0'
'0'
SSPxIF
'0'
'0'

FIGURE 20-30: BRG RESET DUE TO SDAx ARBITRATION DURING START CONDITION


SDAx = 0, SCLx = 1
Set S
Set SSPxIF
Less than TBRG
TBRG
SDAx
SDAx Pulled Low by Other Master; Reset BRG and Assert SDAx.
SCLx
S
SCLx Pulled Low after BRG Time-out
SEN
Set SEN, Enable Start Sequence if SDAx = 1, SCLx = 1
BCLxIF
'0'
S
SSPxIF
SDAx = 0, SCLx = 1, Set SSPxIF
Interrupts Cleared in Software

20.5.17.2 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'; see Figure 20-31). If SDAx is sampled high, the BRG is reloaded and begins counting. If SDAx goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDAx at exactly the same time.

If SCLx goes from high-to-low before the BRG times out and SDAx has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition (see Figure 20-32).

If, at the end of the BRG time-out, both SCLx and SDAx are still high, the SDAx pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCLx pin, the SCLx pin is driven low and the Repeated Start condition is complete.

FIGURE 20-31: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

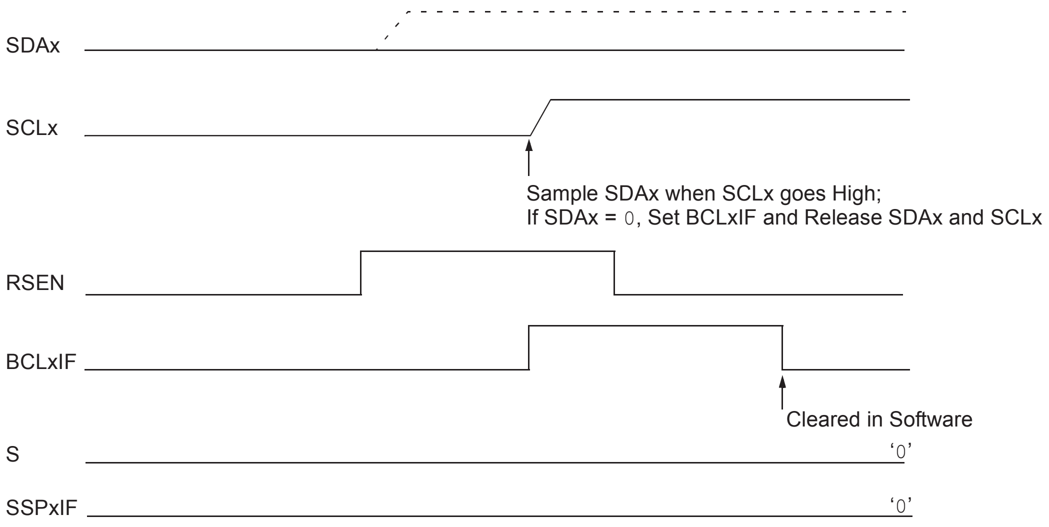
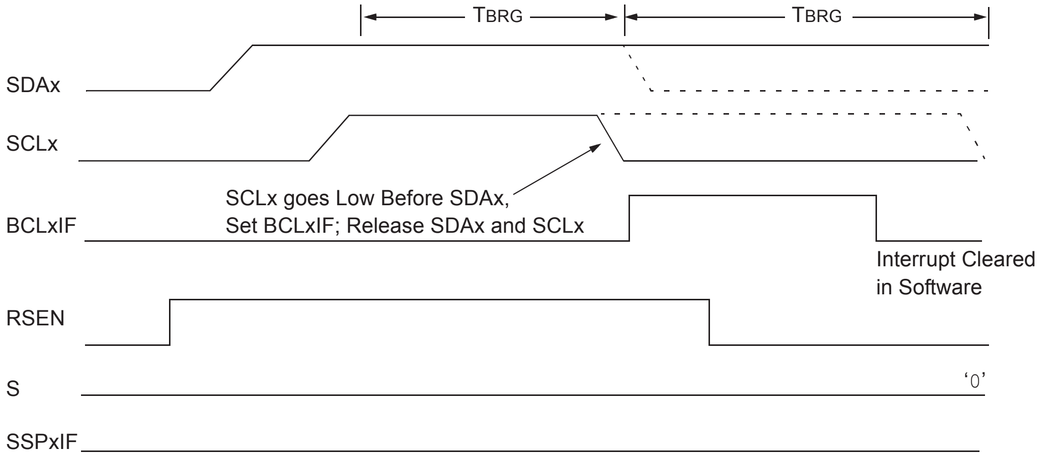

SDAx
SCLx
Sample SDAx when SCLx goes High;
If SDAx = 0, Set BCLxIF and Release SDAx and SCLx
RSEN
BCLxIF
Cleared in Software
S
'0'
SSPxIF
'0'

FIGURE 20-32: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)


TBRG
TBRG
SDAx
SCLx
SCLx goes Low Before SDAx,
Set BCLxIF; Release SDAx and SCLx
BCLxIF
Interrupt Cleared
in Software
RSEN
S
'0'
SSPxIF

20.5.17.3 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 BRG is loaded with SSPxADD<6:0> and counts down to 0. After the BRG times out, SDAx is sampled. If SDAx is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 20-33). If the SCLx pin is sampled low before SDAx is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 20-34).

FIGURE 20-33: BUS COLLISION DURING A STOP CONDITION (CASE 1)

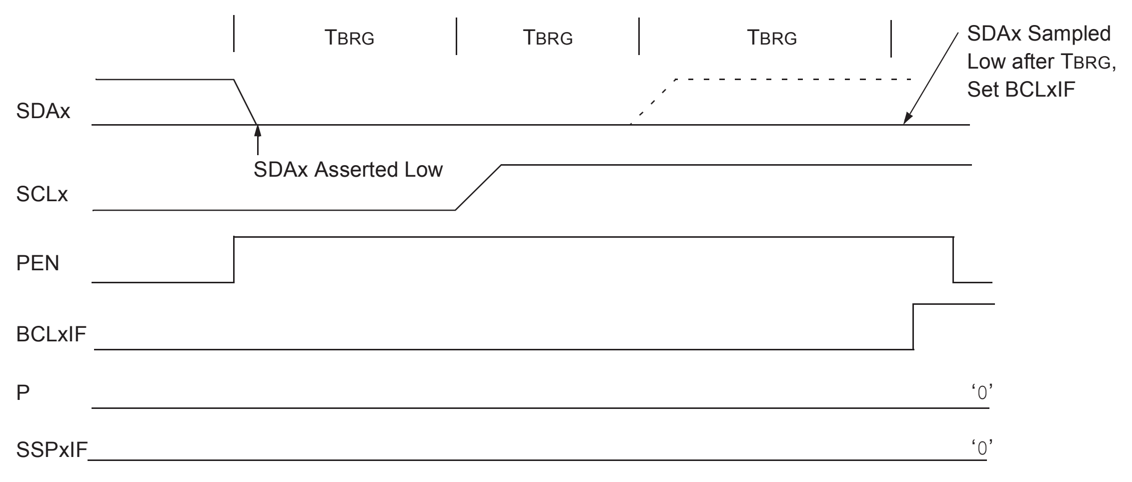

TBRG
TBRG
TBRG
SDAx Sampled Low after TBRG, Set BCLxIF
SDAx
SDAx Asserted Low
SCLx
PEN
BCLxIF
P
'0'
SSPxIF
'0'

FIGURE 20-34: BUS COLLISION DURING A STOP CONDITION (CASE 2)

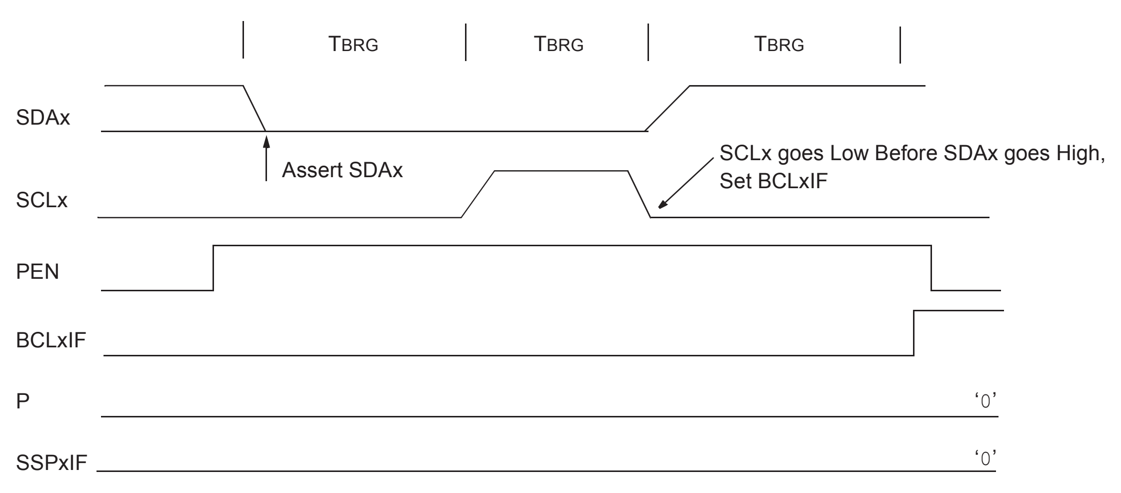

TBRG
TBRG
TBRG
SDAx
Assert SDAx
SCLx goes Low Before SDAx goes High, Set BCLxIF
SCLx
PEN
BCLxIF
P
'0'
SSPxIF
'0'

TABLE 20-4: REGISTERS ASSOCIATED WITH I^{2}C 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	PMPIF(3)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(3)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(3)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR2	OSCFIF	CM2IF	CM1IF	—	BCL1IF	HLVDIF	TMR3IF	CCP2IF
PIE2	OSCFIE	CM2IE	CM1IE	—	BCL1IE	HLVDIE	TMR3IE	CCP2IE
IPR2	OSCFIP	CM2IP	CM1IP	—	BCL1IP	HLVDIP	TMR3IP	CCP2IP
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
TRISD(3)	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
SSP1BUF	MSSP1 Receive Buffer/Transmit Register							
SSP1ADD	MSSP1 Address Register (I^{2}C Slave mode), MSSP1 Baud Rate Reload Register (I^{2}C Master mode)							
SSPxMSK(1)	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0
SSPxCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
SSPxCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
	GCEN	ACKSTAT	ADMSK5(2)	ADMSK4(2)	ADMSK3(2)	ADMSK2(2)	ADMSK1(2)	SEN
SSPxSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
SSP2BUF	MSSP2 Receive Buffer/Transmit Register							
SSP2ADD	MSSP2 Address Register (I^{2}C Slave mode), MSSP2 Baud Rate Reload Register (I^{2}C Master mode)							

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used by the MSSPx module in I^{2}C mode.

Note 1: SSPxMSK shares the same address in SFR space as SSPxADD, but is only accessible in certain I^{2}C Slave mode operations in 7-Bit Masking mode. See **Section 20.5.3.4 “7-Bit Address Masking Mode”** for more details.

2: Alternate bit definitions for use in I^{2}C Slave mode operations only.

3: These bits are only available on 44-pin devices.

21.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is one of two serial I/O modules. (Generically, the EUSART is also known as a Serial Communications Interface or SCI.) The EUSART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs and so on.

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 PIC18F47J13 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/CCP9/PMA5/TX1/CK1/RP17 and RC7/CCP10/PMA4/RX1/DT1/RP18), and remapped (RPn1/TX2/CK2 and RPn2/RX2/DT2), respectively. In order to configure these pins as an EUSART:

- For EUSART1:
  - SPEN bit (RCSTA1<7>) must be set (= `1`)
  - TRISC<7> bit must be set (= `1`)
  - TRISC<6> bit must be cleared (= `0`) for Asynchronous and Synchronous Master modes
  - TRISC<6> bit must be set (= `1`) for Synchronous Slave mode
- For EUSART2:
  - SPEN bit (RCSTA2<7>) must be set (= `1`)
  - TRIS bit for RPn2/RX2/DT2 = `1`
  - TRIS bit for RPn1/TX2/CK2 = `0` for Asynchronous and Synchronous Master modes
  - TRISC<6> bit must be set (= `1`) for Synchronous Slave mode

Note: The EUSART control will automatically reconfigure the pin from input to output as needed.

The TXx/CKx I/O pins have an optional open-drain output capability. By default, when this pin is used by the EUSART as an output, it will function as a standard push-pull CMOS output. The TXx/CKx I/O pins' open-drain, output feature can be enabled by setting the corresponding UxOD bit in the ODCON2 register. For more details, see **Section 20.3.3 "Open-Drain Output Option"**.

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 covered in detail in Register 21-1, Register 21-2 and Register 21-3, respectively.

Note: Throughout this section, references to register and bit names that may be associated with a specific EUSART module are referred to generically by the use of 'x' in place of the specific module number. Thus, "RCSTAx" might refer to the Receive Status register for either EUSART1 or EUSART2.

REGISTER 21-1: TXSTAx: TRANSMIT STATUS AND CONTROL REGISTER (1, ACCESS FADh; 2, FA8h)

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 and the TXx/CKx pin is configured as an output
`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 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 21-2: RCSTAx: RECEIVE STATUS AND CONTROL REGISTER (1, ACCESS FACh; 2, FC9h)

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 enabled
`0` = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-Bit Receive Enable bit

`1` = Selects 9-bit reception
`0` = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit

Asynchronous mode:
Don’t care.

Synchronous mode – Master:
`1` = Enables single receive
`0` = Disables single receive
This bit is cleared after reception is complete.

Synchronous mode – Slave:
Don’t care.

bit 4 **CREN:** Continuous Receive Enable bit

Asynchronous mode:
`1` = Enables receiver
`0` = Disables receiver

Synchronous mode:
`1` = Enables continuous receive until enable bit, CREN, is cleared (CREN overrides SREN)
`0` = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit

Asynchronous mode 9-Bit (RX9 = `1`):
`1` = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
`0` = Disables address detection, all bytes are received and 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.

REGISTER 21-3: BAUDCONx: BAUD RATE CONTROL REGISTER (ACCESS F7Eh, F7Ch)

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

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

bit 7 **ABDOVF:** Auto-Baud Acquisition Rollover Status bit

1 = A BRG rollover has occurred during Auto-Baud Rate Detect mode (must be cleared in software)
0 = No BRG rollover has occurred

bit 6 **RCIDL:** Receive Operation Idle Status bit

1 = Receive operation is Idle
0 = Receive operation is active

bit 5 **RXDTP:** Data/Receive Polarity Select bit

Asynchronous mode:
1 = Receive data (RXx) is inverted (active-low)
0 = Receive data (RXx) is not inverted (active-high)

Synchronous mode:
1 = Data (DTx) is inverted (active-low)
0 = Data (DTx) is not inverted (active-high)

bit 4 **TXCKP:** Synchronous Clock Polarity Select bit

Asynchronous mode:
1 = Idle state for transmit (TXx) is a low level
0 = Idle state for transmit (TXx) is a high level

Synchronous mode:
1 = Idle state for clock (CKx) is a high level
0 = Idle state for clock (CKx) is a low level

bit 3 **BRG16:** 16-Bit Baud Rate Register Enable bit

1 = 16-bit Baud Rate Generator – SPBRGHx and SPBRGx
0 = 8-bit Baud Rate Generator – SPBRGx only (Compatible mode), SPBRGHx value is ignored

bit 2 **Unimplemented:** Read as ‘0’

bit 1 **WUE:** Wake-up Enable bit

Asynchronous mode:
1 = EUSART will continue to sample the RXx pin – interrupt is generated on the falling edge; bit is cleared in hardware on the following rising edge
0 = RXx pin is not monitored or the rising edge detected

Synchronous mode:
Unused in this mode.

bit 0 **ABDEN:** Auto-Baud Detect Enable bit

Asynchronous mode:
1 = Enable baud rate measurement on the next character; requires reception of a Sync field (55h); cleared in hardware upon completion
0 = Baud rate measurement disabled or completed

Synchronous mode:
Unused in this mode.

21.1 Baud Rate Generator (BRG)

The BRG is a dedicated, 8-bit or 16-bit generator that supports both the Asynchronous and Synchronous modes of the 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, the BRGH (TXSTAx<2>) and BRG16 (BAUDCONx<3>) bits also control the baud rate. In Synchronous mode, BRGH is ignored.

Table 21-1 provides 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 21-1. From this, the error in baud rate can be determined. An example calculation is provided in Example 21-1. Typical baud rates and error values for the various Asynchronous modes are provided in Table 21-2. It may be advantageous to use the high baud rate (BRGH = `1`) or the 16-bit BRG to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGHx:SPBRGx registers causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

When operated in the Synchronous mode, SPBRGH:SPBRG values of 0000h and 0001h are not supported. In the Asynchronous mode, all BRG values may be used.

21.1.1 OPERATION IN POWER-MANAGED MODES

The device clock is used to generate the desired baud rate. When one of the power-managed modes is entered, the new clock source may be operating at a different frequency. This may require an adjustment to the value in the SPBRGx register pair.

21.1.2 SAMPLING

The data on the RXx pin (either RC7/CCP10/PMA4/RX1/DT1/RP18 or RPn/RX2/DT2) is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RXx pin.

TABLE 21-1: BAUD RATE FORMULAS
TABLE 21-4:

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

EXAMPLE 21-1: CALCULATING BAUD RATE ERROR

```
For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, and
8-bit BRG:
Desired Baud Rate = Fosc/(64 ([SPBRGHx:SPBRGx] + 1))
Solving for SPBRGHx:SPBRGx:
    X   =   ((Fosc/Desired Baud Rate)/64) – 1
        =   ((16000000/9600)/64) – 1
        =   [25.042] = 25
Calculated Baud Rate=16000000/(64 (25 + 1))
        =   9615
Error   =   (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
        =   (9615 – 9600)/9600 = 0.16%
```

TABLE 21-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
BAUDCONx	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGHx	EUSARTx Baud Rate Generator High Byte							
SPBRGx	EUSARTx Baud Rate Generator Low Byte							

Legend: — = unimplemented, read as ‘`0`’. Shaded cells are not used by the BRG.

TABLE 21-3: BAUD RATES FOR ASYNCHRONOUS MODES

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.16	207	0.300	-0.16	103	0.300	-0.16	51
1.2	1.202	0.16	51	1.201	-0.16	25	1.201	-0.16	12
2.4	2.404	0.16	25	2.403	-0.16	12	—	—	—
9.6	8.929	-6.99	6	—	—	—	—	—	—
19.2	20.833	8.51	2	—	—	—	—	—	—
57.6	62.500	8.51	0	—	—	—	—	—	—
115.2	62.500	-45.75	0	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2.403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9615.	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

TABLE 21-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082	0.300	-0.04	1665
1.2	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520	1.201	-0.16	415
2.4	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259	2.403	-0.16	207
9.6	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.04	832	0.300	-0.16	415	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	33332	0.300	0.00	16665	0.300	0.00	8332	0.300	-0.01	6665
1.2	1.200	0.00	8332	1.200	0.02	4165	1.200	0.02	2082	1.200	-0.04	1665
2.4	2.400	0.02	4165	2.400	0.02	2082	2.402	0.06	1040	2.400	-0.04	832
9.6	9.606	0.06	1040	9.596	-0.03	520	9.615	0.16	259	9.615	-0.16	207
19.2	19.193	-0.03	520	19.231	0.16	259	19.231	0.16	129	19.230	-0.16	103
57.6	57.803	0.35	172	57.471	-0.22	86	58.140	0.94	42	57.142	0.79	34
115.2	114.943	-0.22	86	116.279	0.94	42	113.636	-1.36	21	117.647	-2.12	16

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.01	3332	0.300	-0.04	1665	0.300	-0.04	832
1.2	1.200	0.04	832	1.201	-0.16	415	1.201	-0.16	207
2.4	2.404	0.16	415	2.403	-0.16	207	2.403	-0.16	103
9.6	9.615	0.16	103	9.615	-0.16	51	9.615	-0.16	25
19.2	19.231	0.16	51	19.230	-0.16	25	19.230	-0.16	12
57.6	58.824	2.12	16	55.555	3.55	8	—	—	—
115.2	111.111	-3.55	8	—	—	—	—	—	—

21.1.3 AUTO-BAUD RATE DETECT

The Enhanced USART 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 21-1) begins whenever a Start bit is received and the ABDEN bit is set. The calculation is self-averaging.

In the Auto-Baud Rate Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RXx signal, the RXx signal is timing the BRG. In ABD mode, the internal BRG 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 ABD 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 high bit time in order to minimize any effects caused by asymmetry of the incoming signal. After a Start bit, the SPBRGx begins counting up, using the preselected clock source on the first rising edge of RXx. After eight bits on the RXx pin, or the fifth rising edge, an accumulated value totalling the proper BRG period is left in the SPBRGHx:SPBRGx register pair. Once the 5th edge is seen (this should correspond to the Stop bit), the ABDEN bit is automatically cleared.

If a rollover of the BRG occurs (an overflow from FFFFh to 0000h), the event is trapped by the ABDOVF status bit (BAUDCONx<7>). It is set in hardware by BRG rollovers and can be set or cleared by the user in software. ABD mode remains active after rollover events and the ABDEN bit remains set (Figure 21-2).

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. Note that the BRG clock can 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 21-4 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, it is recommended to set BRG16 bit if auto-baud feature is used.

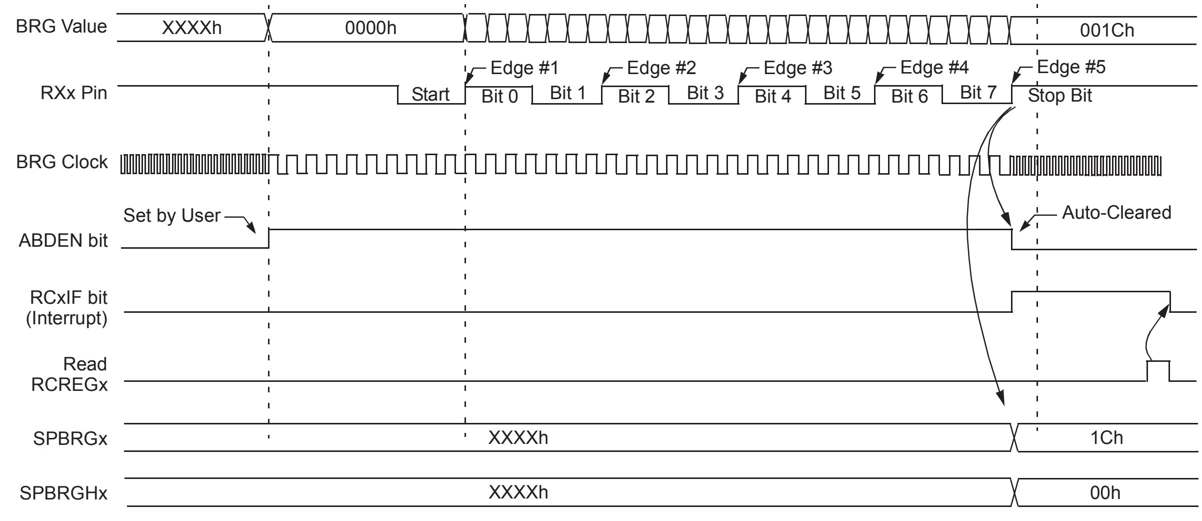
TABLE 21-4: BRG COUNTER CLOCK RATES

BRG16	BRGH	BRG Counter Clock
0	0	FOSC/512
0	1	FOSC/128
1	0	FOSC/128
1	1	FOSC/32

21.1.3.1 ABD and EUSART Transmission

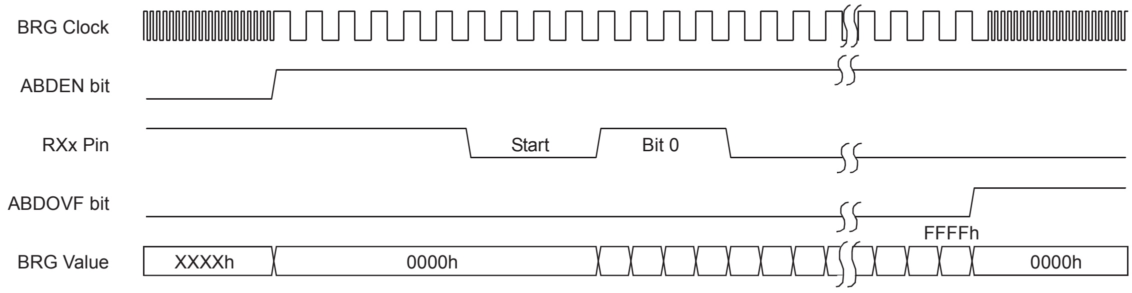
Since the BRG clock is reversed during ABD acquisition, the EUSART transmitter cannot be used during ABD. This means that whenever the ABDEN bit is set, TXREGx cannot be written to. Users should also ensure that ABDEN does not become set during a transmit sequence. Failing to do this may result in unpredictable EUSART operation.

FIGURE 21-1: AUTOMATIC BAUD RATE CALCULATION


BRG Value
XXXXh
0000h
001Ch
RXx Pin
Edge #1
Edge #2
Edge #3
Edge #4
Edge #5
Start
Bit 0
Bit 1
Bit 2
Bit 3
Bit 4
Bit 5
Bit 6
Bit 7
Stop Bit
BRG Clock
Set by User
Auto-Cleared
ABDEN bit
RCxIF bit (Interrupt)
Read RCREGx
SPBRGx
XXXXh
1Ch
SPBRGHx
XXXXh
00h

Note: The ABD sequence requires the EUSART module to be configured in Asynchronous mode and WUE = 0.

FIGURE 21-2: BRG OVERFLOW SEQUENCE


BRG Clock
ABDEN bit
RXx Pin
Start
Bit 0
ABDOVF bit
FFFFh
BRG Value
XXXXh
0000h
0000h

21.2 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 BRG 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 BRG 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 ninth 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

21.2.1 EUSART ASYNCHRONOUS TRANSMITTER

Figure 21-3 displays the EUSART transmitter block diagram.

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.

FIGURE 21-3: EUSART TRANSMIT BLOCK DIAGRAM

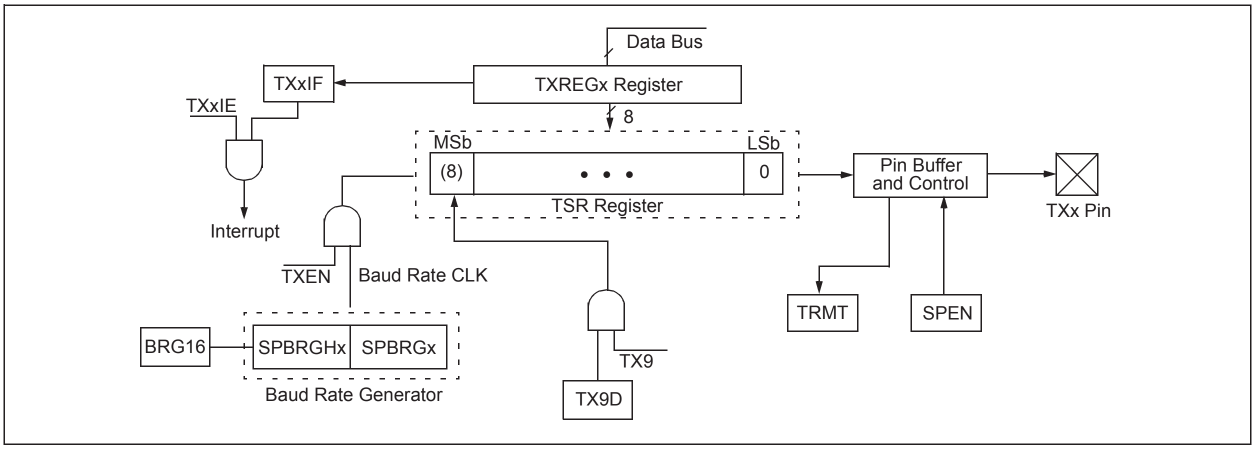


FIGURE 21-4: ASYNCHRONOUS TRANSMISSION

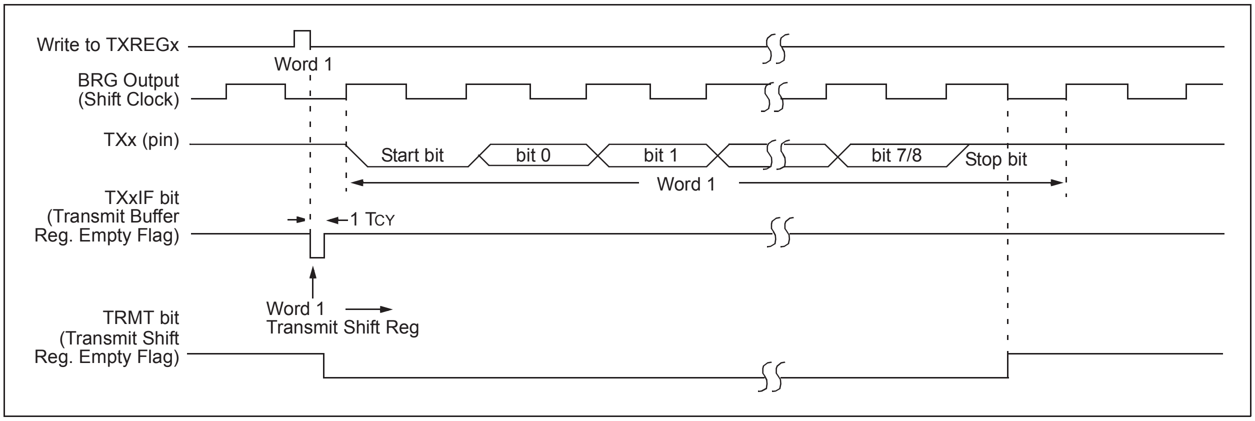
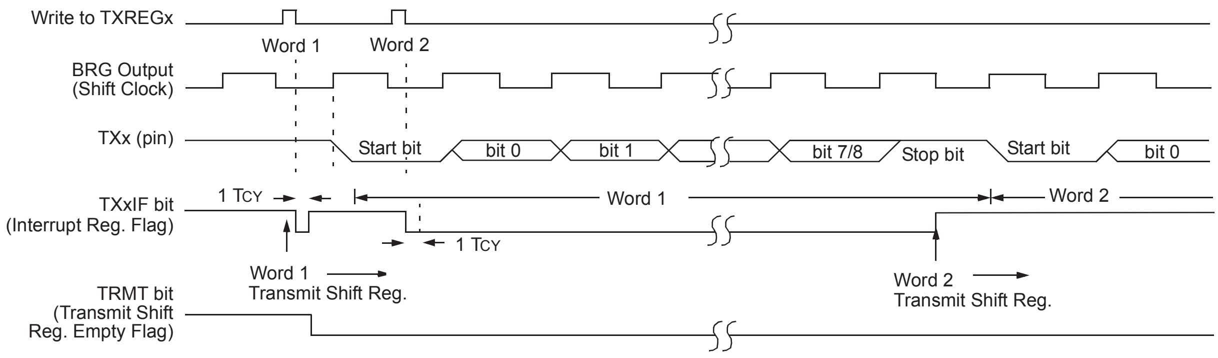

Write to TXREGx
Word 1
BRG Output (Shift Clock)
TXx (pin)
Start bit
bit 0
bit 1
bit 7/8
Stop bit
Word 1
TXxIF bit (Transmit Buffer Reg. Empty Flag)
1 TCY
Word 1 Transmit Shift Reg
TRMT bit (Transmit Shift Reg. Empty Flag)

FIGURE 21-5: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)


Write to TXREGx
Word 1
Word 2
BRG Output (Shift Clock)
TXx (pin)
Start bit
bit 0
bit 1
bit 7/8
Stop bit
Start bit
bit 0
TXxIF bit (Interrupt Reg. Flag)
1 TCY
Word 1
Word 2
1 TCY
Word 1 Transmit Shift Reg.
Word 2 Transmit Shift Reg.
TRMT bit (Transmit Shift Reg. Empty Flag)

Note: This timing diagram shows two consecutive transmissions.

TABLE 21-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF(1)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(1)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(1)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREGx	EUSARTx Transmit Register							
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCONx	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGHx	EUSARTx Baud Rate Generator Register High Byte							
SPBRGx	EUSARTx Baud Rate Generator Register Low Byte							
ODCON2	—	—	—	—	CCP10OD	CCP9OD	U2OD	U1OD

Legend: – = unimplemented locations read as ‘0’. Shaded cells are not used for asynchronous transmission.

Note 1: These bits are only available on 44-pin devices.

21.2.2 EUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is displayed in Figure 21-6. The data is received on the RXx pin and drives the data recovery block. The data recovery block is actually a high-speed shifter, operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

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 ninth bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREGx register.
9. If any error occurred, clear the error by clearing enable bit, CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

21.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCxIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCxIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCxIE and GIE bits are set.
8. Read the RCSTAx register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREGx to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 21-6: EUSARTx RECEIVE BLOCK DIAGRAM

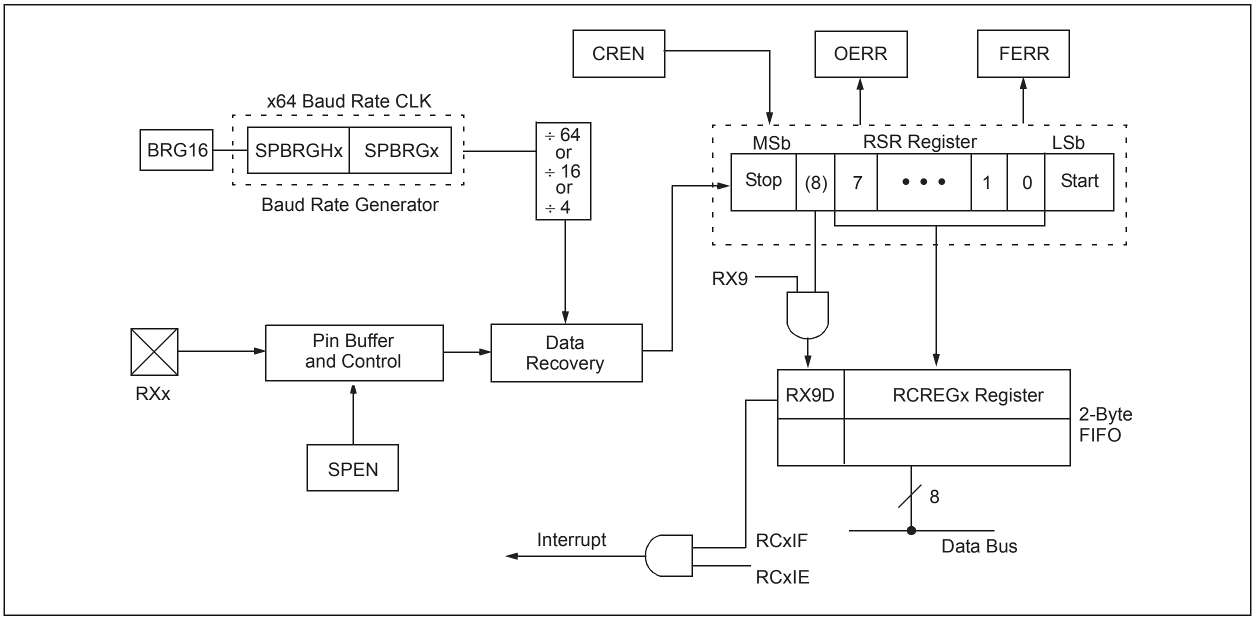
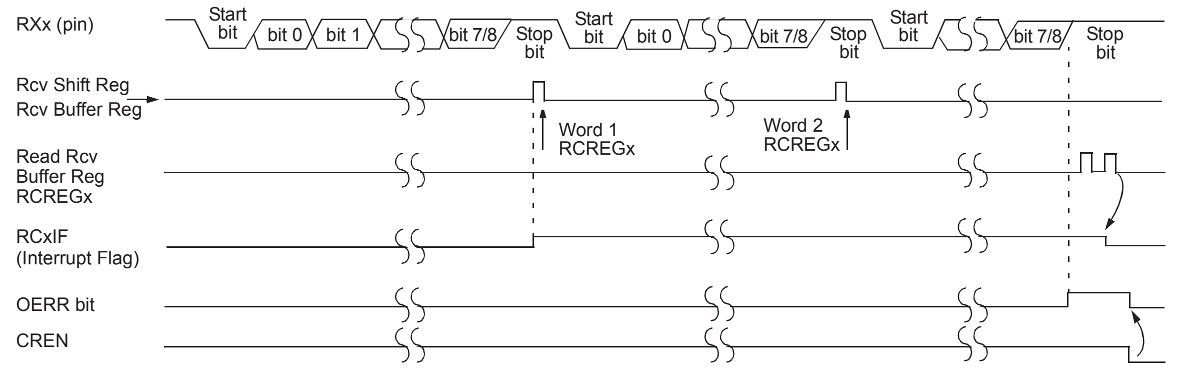


FIGURE 21-7: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RXx input. The RCREGx (Receive Buffer) is read after the third word causing the OERR (Overrun) bit to be set.

TABLE 21-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF(1)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(1)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(1)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREGx	EUSARTx Receive Register							
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCONx	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGHx	EUSARTx Baud Rate Generator High Byte							
SPBRGx	EUSARTx Baud Rate Generator Low Byte							

Legend: — = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

Note 1: These bits are only available on 44-pin devices.

21.2.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the BRG 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 21-8) and asynchronously if the device is in Sleep mode (Figure 21-9). The interrupt condition is cleared by reading the RCREGx register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RXx line following the wake-up event. At this point, the EUSART module is in Idle mode and returns to normal operation. This signals to the user that the Sync Break event is over.

21.2.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RXx/DTx, information with any state changes before the Stop bit may signal a false End-Of-Character (EOC) and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all '`0`'s. This can be 00h (8 bits) for standard RS-232 devices or 000h (12 bits) for 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.

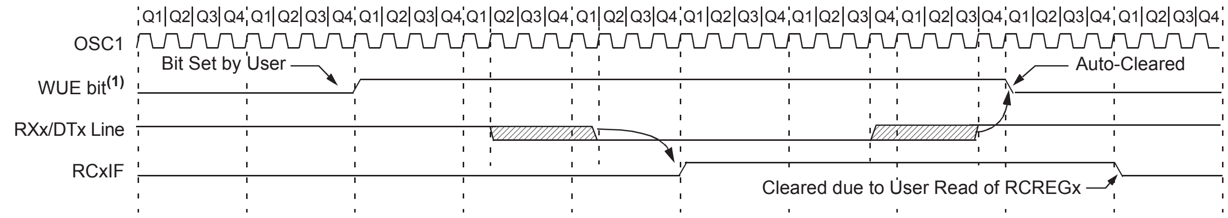
21.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCxIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the 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.

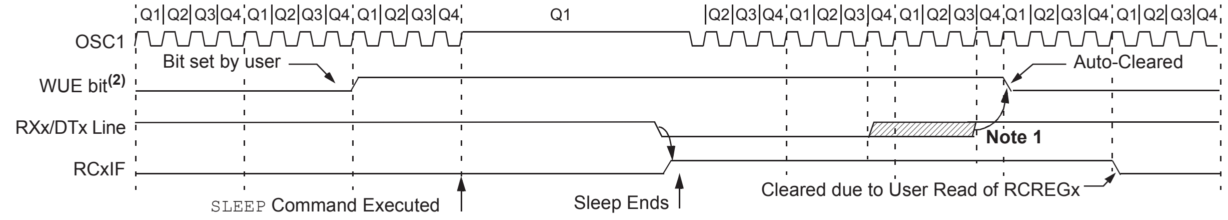
To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering Sleep mode.

FIGURE 21-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION


Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
OSC1
Bit Set by User
WUE bit(1)
Auto-Cleared
RXx/DTx Line
RCxIF
Cleared due to User Read of RCREGx

Note 1: The EUSART remains in Idle while the WUE bit is set.

FIGURE 21-9: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP


Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
OSC1
Bit set by user
WUE bit(2)
Auto-Cleared
RXx/DTx Line
Note 1
RCxIF
SLEEP Command Executed
Sleep Ends
Cleared due to User Read of RCREGx

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.

21.2.5 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>) are set while the Transmit Shift Register is loaded with data.

Note that the value of data written to TXREGx will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN/J2602 specification).

Note that the data value written to the TXREGx for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 21-10 for the timing of the Break character sequence.

21.2.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a Break, followed by an Auto-Baud Sync byte. This sequence is typical of a LIN/J2602 bus master.

1. Configure the 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.

21.2.6 RECEIVING A BREAK CHARACTER

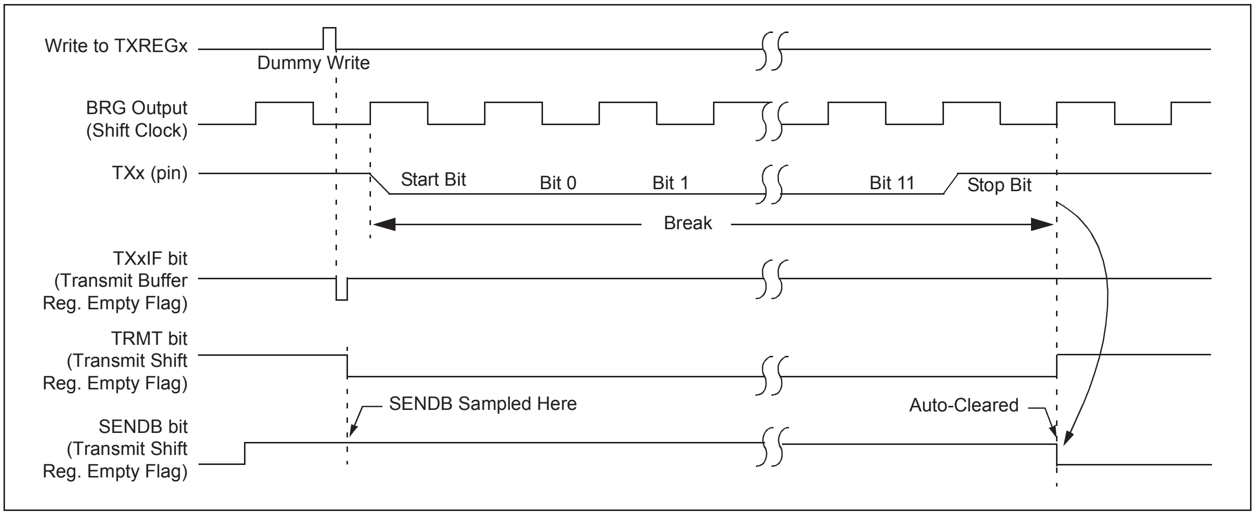
The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 of the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 21.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the 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 21-10: SEND BREAK CHARACTER SEQUENCE



21.3 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.

21.3.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in Figure 21-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The shift register obtains its data from the Read/Write Transmit Buffer register, TXREGx. The TXREGx register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREGx (if available).

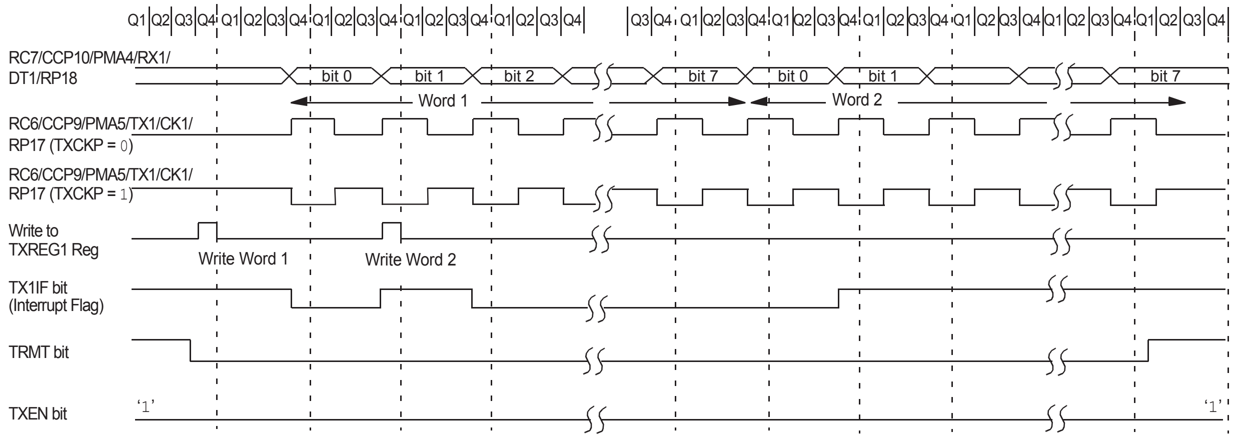
Once the TXREGx register transfers the data to the TSR register (occurs in one TCY), the TXREGx is empty and the TXxIF flag bit is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxIE. TXxIF is set regardless of the state of enable bit, TXxIE; it cannot be cleared in software. It will reset only when new data is loaded into the TXREGx register.

While flag bit, TXxIF, indicates the status of the TXREGx register, another bit, TRMT (TXSTAx<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user must poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

To set up a Synchronous Master Transmission:

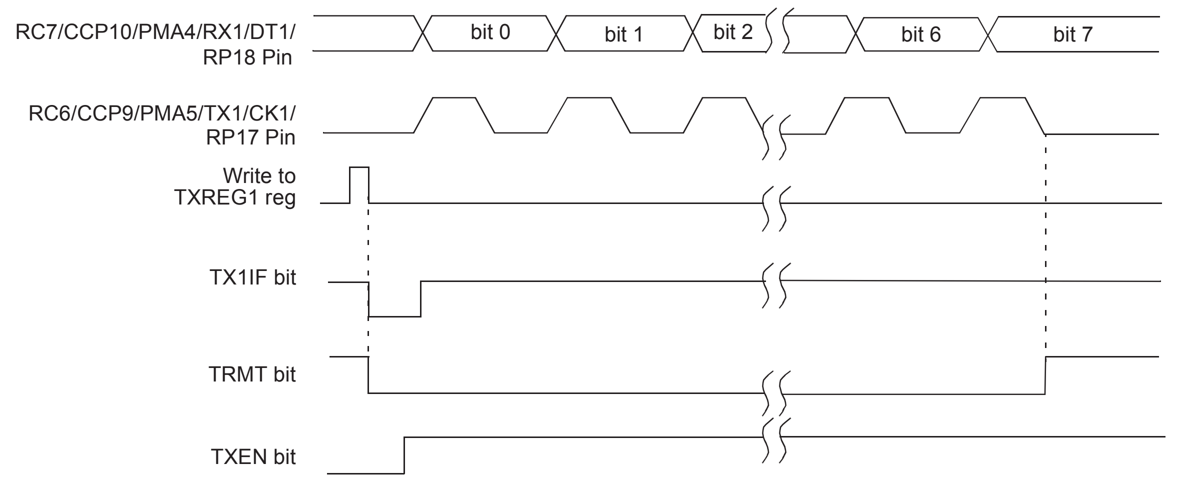
1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the required baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit, TXxIE.
4. If 9-bit transmission is required, set bit, TX9.
5. Enable the transmission by setting bit, TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Start transmission by loading data to the TXREGx register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 21-11: SYNCHRONOUS TRANSMISSION


Q1 Q2 Q3 Q4
RC7/CCP10/PMA4/RX1/DT1/RP18
bit 0
bit 1
bit 2
bit 7
bit 0
bit 1
bit 7
Word 1
Word 2
RC6/CCP9/PMA5/TX1/CK1/RP17 (TXCKP = 0)
RC6/CCP9/PMA5/TX1/CK1/RP17 (TXCKP = 1)
Write to TXREG1 Reg
Write Word 1
Write Word 2
TX1IF bit (Interrupt Flag)
TRMT bit
TXEN bit
'1'
'1'

Note: Sync Master mode, SPBRGx = `0`; continuous transmission of two 8-bit words. This example is equally applicable to EUSART2 (RPn1/TX2/CK2 and RPn2/RX2/DT2).

FIGURE 21-12: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)



Note: This example is equally applicable to EUSART2 (RPn1/TX2/CK2 and RPn2/RX2/DT2).

TABLE 21-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF[(1)]	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE[(1)]	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP[(1)]	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREGx	EUSARTx Transmit Register							
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCONx	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGHx	EUSARTx Baud Rate Generator High Byte							
SPBRGx	EUSARTx Baud Rate Generator Low Byte							
ODCON2	—	—	—	—	CCP10OD	CCP9OD	U2OD	U1OD

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous master transmission.

Note 1: These pins are only available on 44-pin devices.

21.3.2 EUSART SYNCHRONOUS MASTER RECEPTION

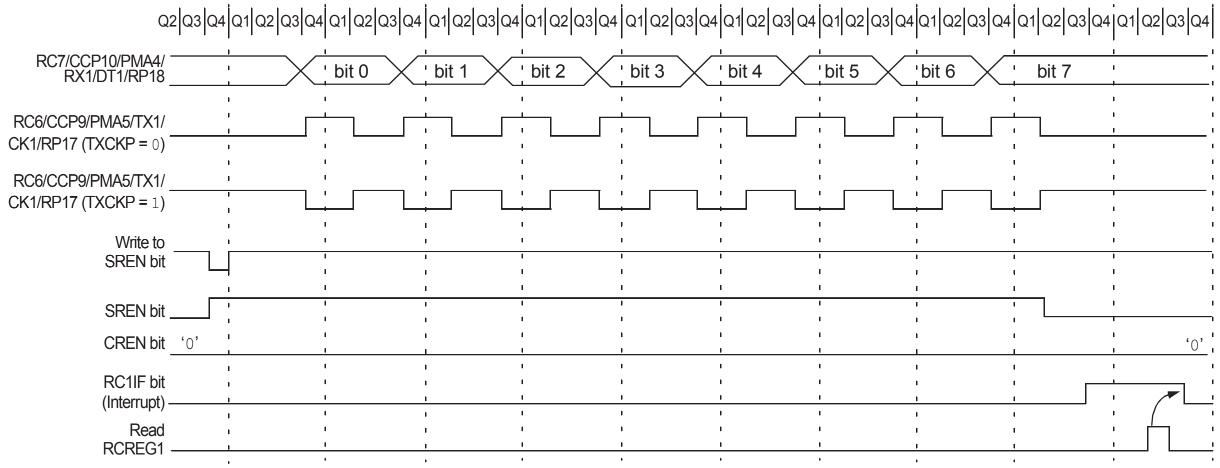
Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTAx<5>), or the Continuous Receive Enable bit, CREN (RCSTAx<4>). Data is sampled on the RXx pin on the falling edge of the clock.

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 ninth bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREGx register.
10. If any error occurred, clear the error by clearing bit, CREN.
11. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 21-13: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)


Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
RC7/CCP10/PMA4/RX1/DT1/RP18
bit 0
bit 1
bit 2
bit 3
bit 4
bit 5
bit 6
bit 7
RC6/CCP9/PMA5/TX1/CK1/RP17 (TXCKP = 0)
RC6/CCP9/PMA5/TX1/CK1/RP17 (TXCKP = 1)
Write to SREN bit
SREN bit
CREN bit
'0'
'0'
RC1IF bit (Interrupt)
Read RCREG1

Note: Timing diagram demonstrates Sync Master mode with bit, SREN = `1`, and bit, BRGH = `0`. This example is equally applicable to EUSART2 (RPn1/TX2/CK2 and RPn2/RX2/DT2).

TABLE 21-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF(1)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(1)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(1)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREGx	EUSARTx Receive Register							
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCONx	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGHx	EUSARTx Baud Rate Generator High Byte							
SPBRGx	EUSARTx Baud Rate Generator Low Byte							
ODCON2	—	—	—	—	CCP10OD	CCP9OD	U2OD	U1OD

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous master reception.

Note 1: These pins are only available on 44-pin devices.

21.4 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.

21.4.1 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 21-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF[1]	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE[1]	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP[1]	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
TXREGx	EUSARTx Transmit Register							
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCONx	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGHx	EUSARTx Baud Rate Generator High Byte							
SPBRGx	EUSARTx Baud Rate Generator Low Byte							

Legend: — = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

Note 1: These pins are only available on 44-pin devices.

21.4.2 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 ninth bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREGx register.
8. If any error occurred, clear the error by clearing bit, CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 21-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF(1)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(1)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(1)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
RCREGx	EUSARTx Receive Register							
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
BAUDCONx	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
SPBRGHx	EUSARTx Baud Rate Generator High Byte							
SPBRGx	EUSARTx Baud Rate Generator Low Byte							

Legend: — = unimplemented, read as ‘0’. Shaded cells are not used for synchronous slave reception.

Note 1: These pins are only available on 44-pin devices.

22.0 10/12-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module in the PIC18F47J13 family of devices has 10 inputs for the 28-pin devices and 13 inputs for the 44-pin devices. This module allows conversion of an analog input signal to a corresponding 10- or 12-bit digital number.

The module has these registers:

- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Port Configuration Register 0 (ANCON0)
- A/D Port Configuration Register 1 (ANCON1)
- A/D Result Registers (ADRESH and ADRESL)
- A/D Trigger Register (ADCTRIG)
- Configuration Register 3 High (ADCSEL, CONFIG3H<1>)

The ADCON0 register, shown in Register 22-1, controls the operation of the A/D module.

The ADCON1 register, shown in Register 22-2, configures the A/D clock source, programmed acquisition time and justification. The ANCON0 and ANCON1 registers, in Register 22-1 and Register 22-2, configure the functions of the port pins.

The ADCSEL Configuration bit (CONFIG3H<1>) sets the module for 10- or 12-bit conversions. The 10-Bit Conversion mode is useful for applications that favor conversion speed over conversion resolution.

REGISTER 22-1: ADCON0: A/D CONTROL REGISTER 0 (ACCESS FC2h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
VCFG1	VCFG0	CHS3(2)	CHS2(2)	CHS1(2)	CHS0(2)	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 **VCFG1:** Voltage Reference Configuration bit (VREF- source)

`1` = VREF- (AN2)
`0` = AVSS(4)

bit 6 **VCFG0:** Voltage Reference Configuration bit (VREF+ source)

`1` = VREF+ (AN3)
`0` = AVDD(4)

bit 5-2 **CHS<3:0>:** Analog Channel Select bits(2)

`0000` = Channel 00 (AN0)
`0001` = Channel 01 (AN1)
`0010` = Channel 02 (AN2)
`0011` = Channel 03 (AN3)
`0100` = Channel 04 (AN4)
`0101` = Channel 05 (AN5)(1)
`0110` = Channel 06 (AN6)(1)
`0111` = Channel 07 (AN7)(1)
`1000` = Channel 08 (AN8)
`1001` = Channel 09 (AN9)
`1010` = Channel 10 (AN10)
`1011` = Channel 11 (AN11)
`1100` = Channel 12 (AN12)
`1101` = (Reserved)
`1110` = VDDCORE
`1111` = VBG Absolute Reference (~1.2V)(3)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit

When ADON = `1`:
`1` = A/D conversion is in progress
`0` = A/D is Idle

bit 0 **ADON:** A/D On bit

`1` = A/D Converter module is enabled
`0` = A/D Converter module is disabled

Note 1: These channels are not implemented on 28-pin devices.
2: Performing a conversion on unimplemented channels will return random values.
3: For best accuracy, the band gap reference circuit should be enabled (ANCON1<7> = `1`) at least 10 ms before performing a conversion on this channel.
4: On 44-pin, QFN devices, AVDD and AVSS reference sources are intended to be externally connected to VDD and VSS levels. Other package types tie AVDD and AVSS to VDD and VSS internally.

REGISTER 22-2: ADCON1: A/D CONTROL REGISTER 1 (ACCESS FC1h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCAL	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

Legend:

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

bit 7 **ADFM:** A/D Result Format Select bit

`1` = Right justified
`0` = Left justified

bit 6 **ADCAL:** A/D Calibration bit

`1` = Calibration is performed on the next A/D conversion
`0` = Normal A/D Converter operation

bit 5-3 **ACQT<2:0>:** A/D Acquisition Time Select bits

`111` = 20 TAD
`110` = 16 TAD
`101` = 12 TAD
`100` = 8 TAD
`011` = 6 TAD
`010` = 4 TAD
`001` = 2 TAD
`000` = 0 TAD

bit 2-0 **ADCS<2:0>:** A/D Conversion Clock Select bits

`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.

REGISTER 22-3: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

U-1	U-1	U-1	U-1	R/WO-1	U-0	R/WO-1	R/WO-1
—	—	—	—	MSSPMSK	—	ADCSEL	IOL1WAY
bit 7							bit 0

Legend:

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

bit 7-4 **Unimplemented:** Program the corresponding Flash Configuration bit to ‘`1`’

bit 3 **MSSPMSK:** MSSP 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 **ADCSEL:** A/D Converter Mode bit

`1` = 10-Bit Conversion mode is enabled
`0` = 12-Bit Conversion mode is enabled

bit 0 **IOL1WAY:** IOLOCK One Way Set Enable bit

`1` = IOLOCK bit (PPSCON<0>) can be set once, provided the unlock sequence has been completed. Once set, the Peripheral Pin Select registers cannot be written to a second time.
`0` = IOLOCK bit (PPSCON<0>) can be set and cleared as needed, provided the unlock sequence has been completed

The ANCON0 and ANCON1 registers are used to configure the operation of the I/O pin associated with each analog channel. Setting any one of the PCFG bits configures the corresponding pin to operate as a digital only I/O. Clearing a bit configures the pin to operate as an analog input for either the A/D Converter or the comparator module. All digital peripherals are disabled and digital inputs read as '`0`'. As a rule, I/O pins that are multiplexed with analog inputs default to analog operation on device Resets.

In order to correctly perform A/D conversions on the VBG band gap reference (ADCON0<5:2> = `1111`), the reference circuit must be powered on first. The VBGEN bit in the ANCON1 register allows the firmware to manually request that the band gap reference circuit should be enabled. For best accuracy, firmware should allow a settling time of at least 10 ms prior to performing the first acquisition on this channel after enabling the band gap reference.

The reference circuit may already have been turned on if some other hardware module (such as the on-chip voltage regulator, comparators or HLVD) has already requested it. In this case, the initial turn-on settling time may have already elapsed and firmware does not need to wait as long before measuring VBG. Once the acquisition is complete, firmware may clear the VBGEN bit, which will save a small amount of power if no other modules are still requesting the VBG reference.

REGISTER 22-4: ANCON0: A/D PORT CONFIGURATION REGISTER 0 (BANKED F48h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7[(1)]	PCFG6[(1)]	PCFG5[(1)]	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

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

bit 7-0 **PCFG<7:0>:** Analog Port Configuration bits (AN7-AN0)
`1` = Pin configured as a digital port
`0` = Pin configured as an analog channel – digital input disabled and reads '`0`'

Note 1: These bits are only available only on 44-pin devices.

REGISTER 22-5: ANCON1: A/D PORT CONFIGURATION REGISTER 1 (BANKED F49h)

R/W-0	R	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
VBGEN	r	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 7							bit 0

Legend:	r = Reserved 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 **VBGEN:** 1.2V Band Gap Reference Enable bit
`1` = 1.2V band gap reference is powered on
`0` = 1.2V band gap reference is turned off to save power (if no other modules are requesting it)

bit 6 **Reserved:** Always maintain as '`0`' for lowest power consumption

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

bit 4-0 **PCFG<12:8>:** Analog Port Configuration bits (AN12-AN8)
`1` = Pin configured as a digital port
`0` = Pin configured as an analog channel – digital input disabled and reads '`0`'

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (AVDD and AVSS), or the voltage level on the RA3/AN3/C1INB/VREF+ and RA2/AN2/C2INB/C1IND/C3INB/VREF-/CVREF pins.

The A/D Converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

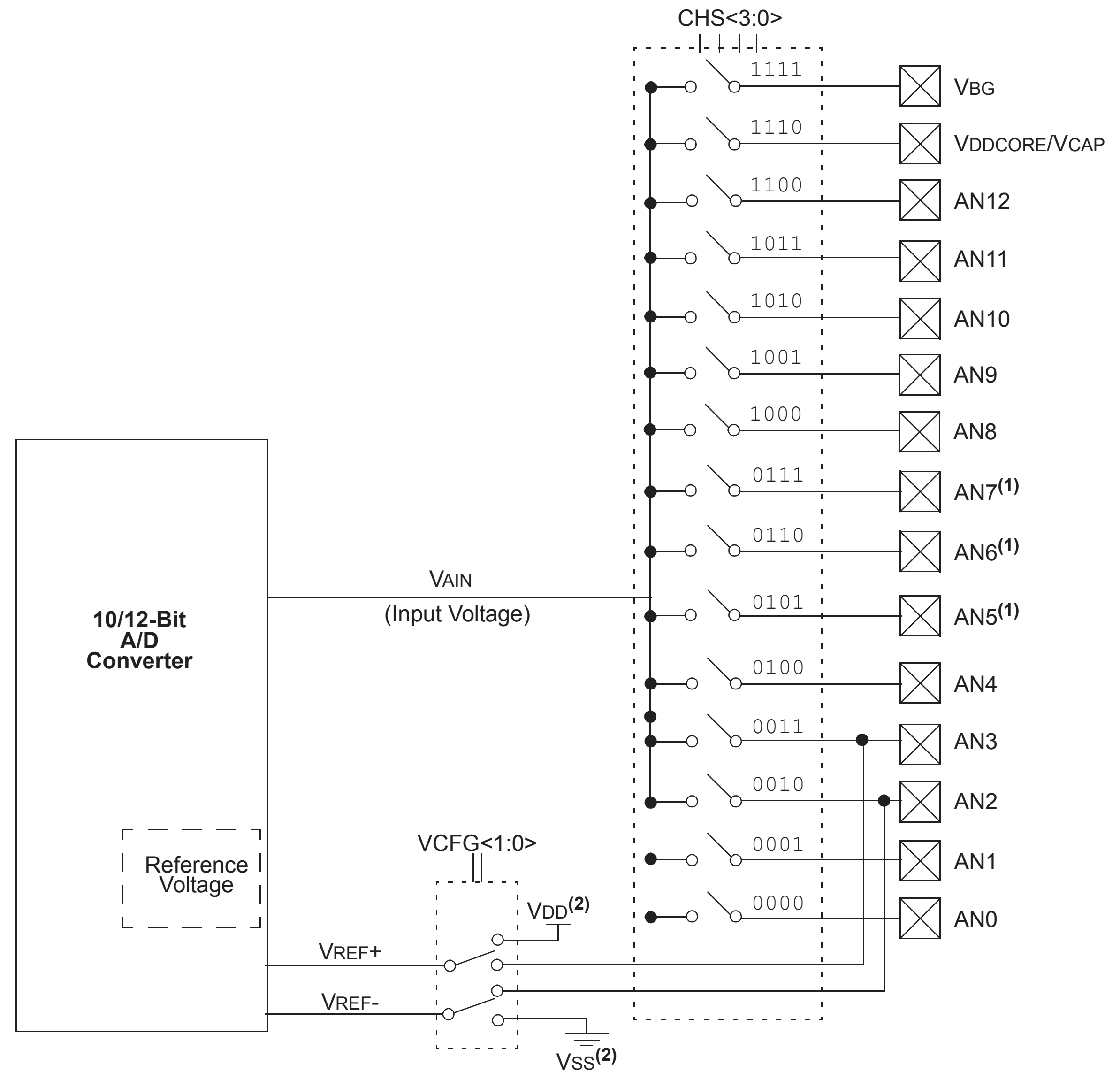
The output of the sample and hold is the input into the converter, which generates the result via successive approximation.

Each port pin associated with the A/D Converter can be configured as an analog input or as a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH:ADRESL register pair, the GO/DONE bit (ADCON0<1>) is cleared and the A/D Interrupt Flag bit, ADIF, is set.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion in progress is aborted. The value in the ADRESH:ADRESL register pair is not modified for a Power-on Reset (POR). These registers will contain unknown data after a POR.

Figure 22-1 provides the block diagram of the A/D module.

FIGURE 22-1: A/D BLOCK DIAGRAM


CHS<3:0>
1111
1110
1100
1011
1010
1001
1000
0111
0110
0101
0100
0011
0010
0001
0000
VBG
VDDCORE/VCAP
AN12
AN11
AN10
AN9
AN8
AN7(1)
AN6(1)
AN5(1)
AN4
AN3
AN2
AN1
AN0
VAIN
(Input Voltage)
10/12-Bit A/D Converter
Reference Voltage
VCFG<1:0>
VDD(2)
VREF+
VREF-
VSS(2)

Note 1: Channels, AN5, AN6 and AN7, are not available on 28-pin devices.
2: I/O pins have diode protection to VDD and VSS.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as inputs. To determine acquisition time, see **Section 22.1 “A/D Acquisition Requirements”**. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/$\overline{\text{DONE}}$ bit and the actual start of the conversion.

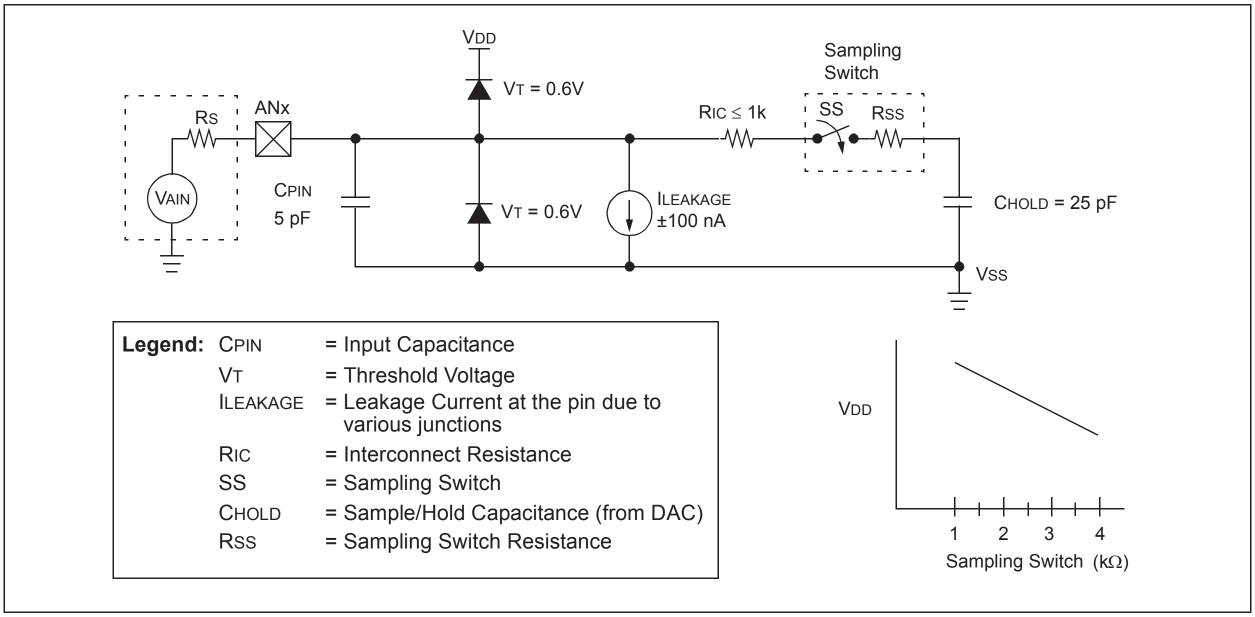
The following steps should be followed to do an A/D conversion:

1. Configure the A/D module:
   - Configure the required ADC pins as analog pins using ANCON0, ANCON1
   - Set voltage reference using ADCON0
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON1)
   - Select A/D conversion clock (ADCON1)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
   - Set GO/$\overline{\text{DONE}}$ bit (ADCON0<1>)
5. Wait for A/D conversion to complete, by either:
   - Polling for the GO/$\overline{\text{DONE}}$ bit to be cleared

   OR

   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit, ADIF, if required.
7. For the next conversion, go to Step 1 or Step 2, as required. The A/D conversion time per bit is defined as T$_{AD}$. A minimum Wait of 2 T$_{AD}$ is required before the next acquisition starts.

FIGURE 22-2: ANALOG INPUT MODEL



Legend:
C$_{PIN}$ = Input Capacitance
V$_T$ = Threshold Voltage
I$_{LEAKAGE}$ = Leakage Current at the pin due to various junctions
R$_{IC}$ = Interconnect Resistance
SS = Sampling Switch
C$_{HOLD}$ = Sample/Hold Capacitance (from DAC)
R$_{SS}$ = Sampling Switch Resistance

22.1 A/D Acquisition Requirements

For the A/D Converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is illustrated in Figure 22-2. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor, CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD). The source impedance affects the offset voltage at the analog input (due to pin leakage current). **The maximum recommended impedance for analog sources is 2.5 kΩ for 10-bit conversions and 1 kΩ for 12-bit conversions**. After the analog input channel is selected (changed), the channel must be sampled for at least the minimum acquisition time before starting a conversion.

> **Note:** When the conversion is started, the holding capacitor is disconnected from the input pin.

To calculate the minimum acquisition time, Equation 22-1 may be used. This equation assumes that 1/2 LSb error is used (1,024 steps for the 10-bit A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Equation 22-3 provides the calculation of the minimum required acquisition time, TACQ. This calculation is based on the following application system assumptions:

CHOLD = 25 pF
Rs = 2.5 kΩ
Conversion Error ≤ 1/2 LSb
VDD = 3V → Rss = 2 kΩ
Temperature = 85°C (system max.)

EQUATION 22-1: ACQUISITION TIME

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_{C} + T_{COFF} \end{aligned}$$

EQUATION 22-2: 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 22-3: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$$\begin{aligned} T_{ACQ} &= T_{AMP} + T_C + T_{COFF} \\ T_{AMP} &= 0.2\ \mu s \\ T_{COFF} &= (\text{Temp} - 25^\circ C)(0.02\ \mu s/^\circ C) \\ &\quad (85^\circ C - 25^\circ C)(0.02\ \mu s/^\circ C) \\ &\quad 1.2\ \mu s \end{aligned}$$

Temperature coefficient is only required for temperatures > 25°C. Below 25°C, TCOFF = 0 µs.

$$\begin{aligned} T_C &= -(C_{HOLD})(R_{IC} + R_{SS} + R_S) \ln(1/2048)\ \mu s \\ &\quad -(25\ pF)\ (1\ k\Omega + 2\ k\Omega + 2.5\ k\Omega) \ln(0.0004883)\ \mu s \\ &\quad 1.05\ \mu s \\ T_{ACQ} &= 0.2\ \mu s + 1.05\ \mu s + 1.2\ \mu s \\ &\quad 2.45\ \mu s \end{aligned}$$

22.2 Selecting and Configuring Automatic Acquisition Time

The ADCON1 register allows the user to select an acquisition time that occurs each time the GO/$\overline{\text{DONE}}$ bit is set.

When the GO/$\overline{\text{DONE}}$ 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/$\overline{\text{DONE}}$ bit. This occurs when the ACQT<2:0> bits (ADCON1<5:3>) remain in their Reset state (‘`000`’) and is compatible with devices that do not offer programmable acquisition times.

If desired, the ACQT bits can be set to select a programmable acquisition time for the A/D module. When the GO/$\overline{\text{DONE}}$ 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/$\overline{\text{DONE}}$ bit.

In either case, when the conversion is completed, the GO/$\overline{\text{DONE}}$ bit is cleared, the ADIF flag is set and the A/D begins sampling the currently selected channel again. If an acquisition time is programmed, there is nothing to indicate if the acquisition time has ended or if the conversion has begun.

22.3 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 11 TAD per 10-bit conversion and 13 TAD per 12-bit conversion. The source of the A/D conversion clock is software selectable.

There are seven possible options for TAD:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Internal RC Oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be as short as possible but greater than the minimum TAD (see parameter 130 in Table 30-32 for more information).

Table 22-1 provides the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

TABLE 22-1: TAD vs. DEVICE OPERATING FREQUENCIES

AD Clock Source (TAD)		Maximum Device Frequency (MHz)$^{(3)}$	
Operation	**ADCS<2:0>**	**10-Bit Mode**	**12-Bit Mode**
2 TOSC	`000`	**2.86**	**2.5**
4 TOSC	`100`	**5.71**	**5**
8 TOSC	`001`	**11.43**	**10**
16 TOSC	`101`	**22.86**	**20**
32 TOSC	`010`	**45.71**	**40**
64 TOSC	`110`	48	48
RC$^{(2)}$	`011`	1$^{(1)}$	1$^{(1)}$

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.

3: Maximum conversion speeds are achieved at the bolded device frequencies.

22.4 Configuring Analog Port Pins

The ANCON0, ANCON1 and TRISA registers control the operation of the A/D port pins. The port pins needed as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS<3:0> bits and the TRIS bits.

Note 1: When reading the PORT register, all pins configured as analog input channels will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will be accurately converted.

2: Analog levels on any pin defined as a digital input may cause the digital input buffer to consume current out of the device’s specification limits.

22.5 A/D Conversions

Figure 22-3 displays 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 22-4 displays the operation of the A/D Converter after the GO/$\overline{\text{DONE}}$ bit has been set. The ACQT<2:0> bits are set to '`010`' and are selecting a 4 TAD acquisition time before the conversion starts.

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.

22.6 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 Overflow
- RTCC Alarm

To start an A/D conversion:

- The A/D module must be enabled (ADON = `1`)
- The appropriate analog input channel selected
- The minimum acquisition period is set in 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/$\overline{\text{DONE}}$ 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 Timer3 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.

REGISTER 22-6: ADCTRIG: A/D TRIGGER REGISTER (BANKED EB8h)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	TRIGSEL1	TRIGSEL0
bit 7							bit 0

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

bit 7-2	**Unimplemented:** Read as '`0`'
bit 1-0	**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

FIGURE 22-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)

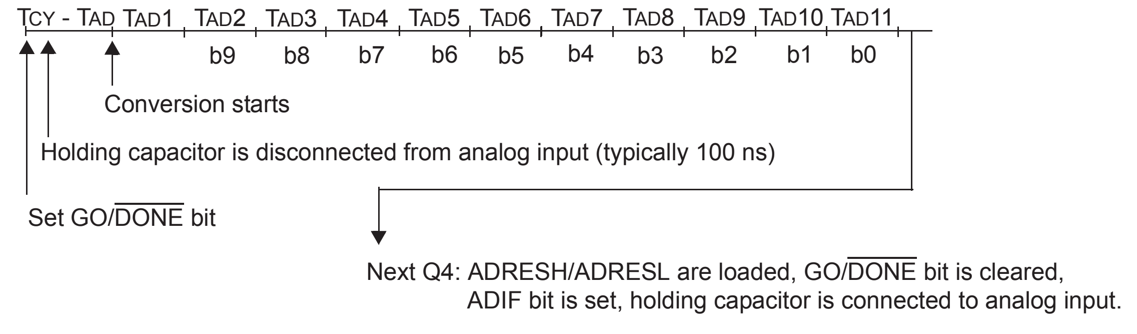
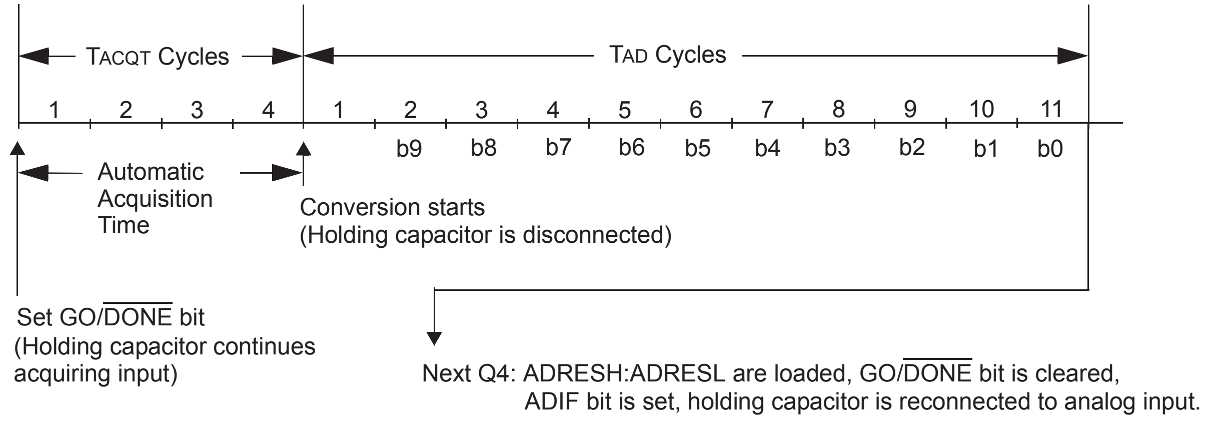

TCY - TAD
TAD1
TAD2
TAD3
TAD4
TAD5
TAD6
TAD7
TAD8
TAD9
TAD10
TAD11
b9
b8
b7
b6
b5
b4
b3
b2
b1
b0
Conversion starts
Holding capacitor is disconnected from analog input (typically 100 ns)
Set GO/$\overline{\text{DONE}}$ bit
Next Q4: ADRESH/ADRESL are loaded, GO/$\overline{\text{DONE}}$ bit is cleared,
ADIF bit is set, holding capacitor is connected to analog input.

FIGURE 22-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)


TACQT Cycles
TAD Cycles
1
2
3
4
1
2
3
4
5
6
7
8
9
10
11
b9
b8
b7
b6
b5
b4
b3
b2
b1
b0
Automatic
Acquisition
Time
Conversion starts
(Holding capacitor is disconnected)
Set GO/$\overline{\text{DONE}}$ bit
(Holding capacitor continues
acquiring input)
Next Q4: ADRESH:ADRESL are loaded, GO/$\overline{\text{DONE}}$ bit is cleared,
ADIF bit is set, holding capacitor is reconnected to analog input.

22.7 A/D Converter Calibration

The A/D Converter in the PIC18F47J13 family of devices includes a self-calibration feature, which compensates for any offset generated within the module. The calibration process is automated and is initiated by setting the ADCAL bit (ADCON1<6>). The next time the GO/$\overline{\text{DONE}}$ bit is set, the module will perform an offset calibration and store the result internally. Thus, subsequent offsets will be compensated.

Example 22-1 provides an example of a calibration routine.

The calibration process assumes that the device is in a relatively steady-state operating condition. If A/D calibration is used, it should be performed after each device Reset or if there are other major changes in operating conditions.

22.8 Operation in Power-Managed Modes

The selection of the automatic acquisition time and A/D conversion clock is determined in part by the clock source and frequency while in a power-managed mode.

If the A/D is expected to operate while the device is in a power-managed mode, the ACQT<2:0> and ADCS<2:0> bits in ADCON1 should be updated in accordance with the power-managed mode clock that will be used. After the power-managed mode is entered (either of the power-managed Run modes), an A/D acquisition or conversion may be started. Once an acquisition or conversion is started, the device should continue to be clocked by the same power-managed mode clock source until the conversion has been completed. If desired, the device may be placed into the corresponding power-managed Idle mode during the conversion.

If the power-managed mode clock frequency is less than 1 MHz, the A/D RC clock source should be selected.

Operation in Sleep mode requires the A/D RC clock to be selected. If bits, ACQT<2:0>, are set to '`000`' and a conversion is started, the conversion will be delayed one instruction cycle to allow execution of the `SLEEP` instruction and entry to Sleep mode. The IDLEN and SCS bits in the OSCCON register must have already been cleared prior to starting the conversion.

EXAMPLE 22-1: SAMPLE A/D CALIBRATION ROUTINE

```
        BCF     ANCON0,PCFG0        ;Make Channel 0 analog
        BSF     ADCON0,ADON         ;Enable A/D module
        BSF     ADCON1,ADCAL        ;Enable Calibration
        BSF     ADCON0,GO           ;Start a dummy A/D conversion
CALIBRATION                         ;
        BTFSC   ADCON0,GO           ;Wait for the dummy conversion to finish
        BRA     CALIBRATION         ;
        BCF     ADCON1,ADCAL        ;Calibration done, turn off calibration enable
                                    ;Proceed with the actual A/D conversion
```

TABLE 22-2: SUMMARY OF A/D REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR1	PMPIF(1)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF
PIE1	PMPIE(1)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE
IPR1	PMPIP(1)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP
PIR2	OSCFIF	CM2IF	CM1IF	—	BCL1IF	HLVDIF	TMR3IF	CCP2IF
PIE2	OSCFIE	CM2IE	CM1IE	—	BCL1IE	HLVDIE	TMR3IE	CCP2IE
IPR2	OSCFIP	CM2IP	CM1IP	—	BCL1IP	HLVDIP	TMR3IP	CCP2IP
ADCTRIG	—	—	—	—	—	—	TRIGSEL1	TRIGSEL0
ADRESH	A/D Result Register High Byte							
ADRESL	A/D Result Register Low Byte							
ADCON0	VCFG1	VCFG0	CHS3	CHS3	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
ANCON0	PCFG7(1)	PCFG6(1)	PCFG5(1)	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
ADCON1	ADFM	ADCAL	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
ANCON1	VBGEN	r	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
CCP2CON	P2M1	P2M0	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
PORTA	RA7	RA6	RA5	—	RA3	RA2	RA1	RA0
TRISA	TRISA7	TRISA6	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0

Legend: — = unimplemented, read as '`0`', r = reserved. Shaded cells are not used for A/D conversion.

Note 1: These bits are only available on 44-pin devices.

23.0 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 voltage references. The digital outputs are available at the pin level and can also be read through the control register. Multiple output and interrupt event generation is also available. Figure 23-1 provides a generic single comparator from the module.

Key features of the module are:

- Independent comparator control
- Programmable input configuration
- Output to both pin and register levels
- Programmable output polarity

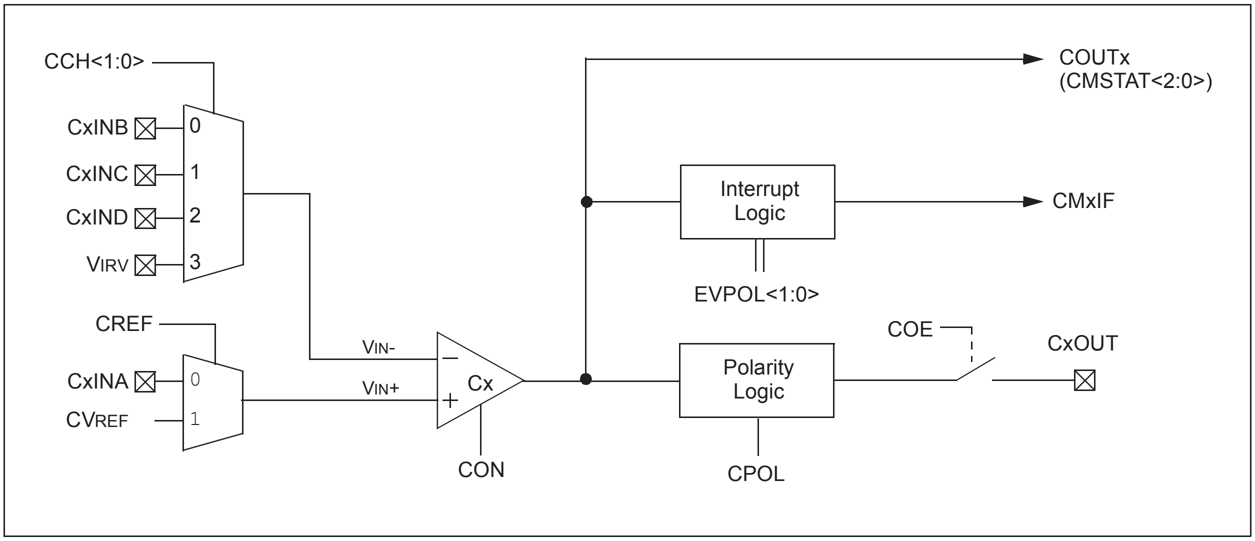
- Independent interrupt generation for each comparator with configurable interrupt-on-change

23.1 Registers

The CMxCON registers (Register 23-1) select the input and output configuration for each comparator, as well as the settings for interrupt generation.

The CMSTAT register (Register 23-2) provides the output results of the comparators. The bits in this register are read-only.

FIGURE 23-1: COMPARATOR SIMPLIFIED BLOCK DIAGRAM


CCH<1:0>
CxINB
CxINC
CxIND
VIRV
0
1
2
3
CREF
CxINA
CVREF
0
1
VIN-
VIN+
Cx
CON
COUTx
(CMSTAT<2:0>)
Interrupt Logic
CMxIF
EVPOL<1:0>
Polarity Logic
CPOL
COE
CxOUT

REGISTER 23-1: CMxCON: COMPARATOR CONTROL 1/2/3 REGISTER (1, ACCESS FD2h; 2, FD1h; 3, BANKED F25h)

R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0
bit 7							bit 0

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

bit 7 **CON:** Comparator Enable bit
1 = Comparator is enabled
0 = Comparator is disabled

bit 6 **COE:** Comparator Output Enable bit
1 = Comparator output is present on the CxOUT pin (assigned in the PPS module)
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 the internal CVREF voltage
0 = Non-inverting input connects to the CxINA pin

bit 1-0 **CCH<1:0>:** Comparator Channel Select bits
11 = Inverting input of comparator connects to VIRV (0.6V)
10 = Inverting input of comparator connects to the CxIND pin
01 = Inverting input of comparator connects to the CxINC pin
00 = Inverting input of comparator connects to the CxINB pin

Note 1: The CMxIF is automatically set any time this mode is selected and must be cleared by the application after the initial configuration.

REGISTER 23-2: CMSTAT: COMPARATOR STATUS REGISTER (ACCESS F70h)

U-0	U-0	U-0	U-0	U-0	R-1	R-1	R-1
—	—	—	—	—	COUT3	COUT2	COUT1
bit 7							bit 0

Legend:

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

bit 7-3 **Unimplemented:** Read as ‘`0`’

bit 2-0 **COUT<3:1>:** Comparator x Status bits

(For example, COUT3 gives the status for Comparator 3.)

If CPOL (CMxCON<5>) = `0` (non-inverted polarity):

`1` = Comparator VIN+ > VIN-

`0` = Comparator VIN+ < VIN-

If CPOL = `1` (inverted polarity):

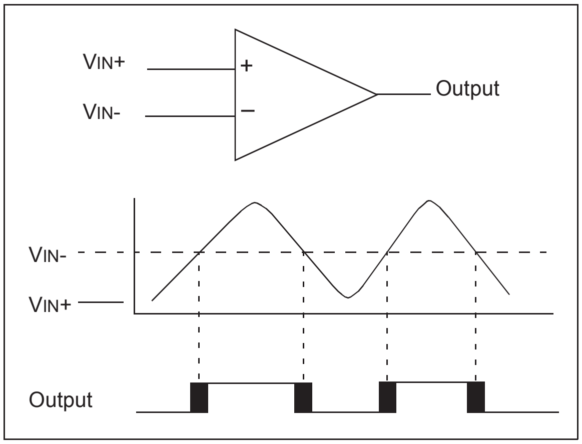
`1` = Comparator VIN+ < VIN-

`0` = Comparator VIN+ > VIN-

23.2 Comparator Operation

A single comparator is shown in Figure 23-2, along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input, VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input, VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator, in Figure 23-2, represent the uncertainty due to input offsets and response time.

FIGURE 23-2: SINGLE COMPARATOR



23.3 Comparator Response Time

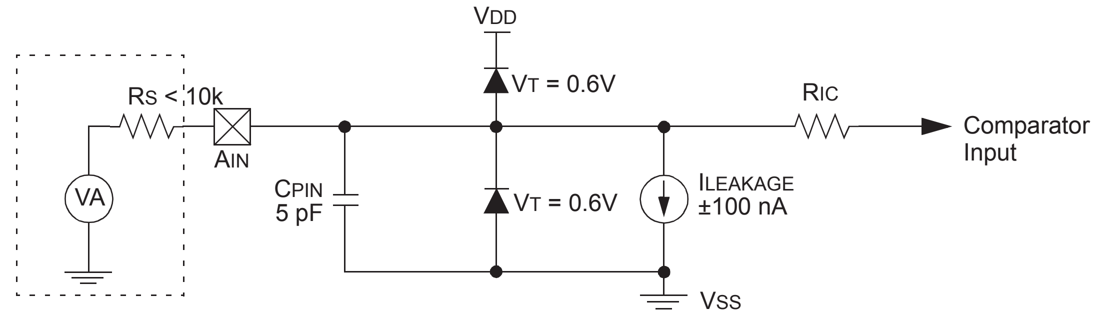
Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level.

The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response to a comparator input change. Otherwise, the maximum delay of the comparators should be used (see **Section 30.0 "Electrical Characteristics"**).

23.4 Analog Input Connection Considerations

Figure 23-3 provides a simplified circuit for an analog input. Since the analog pins are connected to a digital output, they have reverse biased diodes to VDD and VSS. The analog input, therefore, must be between VSS and VDD. If the input voltage deviates from this range, by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up condition may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

FIGURE 23-3: COMPARATOR ANALOG INPUT MODEL



Legend:
CPIN = Input Capacitance
VT = Threshold Voltage
ILEAKAGE = Leakage Current at the pin due to various junctions
RIC = Interconnect Resistance
RS = Source Impedance
VA = Analog Voltage

23.5 Comparator Control and Configuration

Each comparator has up to eight possible combinations of inputs: up to four external analog inputs and one of two internal voltage references.

Both comparators allow a selection of the signal from pin, CxINA, or the voltage from the comparator reference (CVREF) on the non-inverting channel. This is compared to either CxINB, CxINC, CxIND, CTMU or the microcontroller's fixed internal reference voltage (VIRV, 0.6V nominal) on the inverting channel.

Table 23-1 provides the comparator inputs and outputs tied to fixed I/O pins.

TABLE 23-1: COMPARATOR INPUTS AND OUTPUTS

Comparator	Input or Output	I/O Pin
1	C1INA (VIN+)	RA0
	C1INB (VIN-)	RA3
	C1INC (VIN-)	RA5
	C1IND (VIN-)	RA2
	C1OUT	Remapped RPn
2	C2INA(VIN+)	RA1
	C2INB(VIN-)	RA2
	C2INC (VIN-)	RB2
	C2IND (VIN-)	RC2
	C2OUT	Remapped RPn
3	C3INA(VIN+)	RB3
	C3INB(VIN-)	RA2
	C3INC (VIN-)	RB1
	C3IND (VIN-)	RB0
	C3OUT	Remapped RPn

23.5.1 COMPARATOR ENABLE AND INPUT SELECTION

Setting the CON bit of the CMxCON register (CMxCON<7>) enables the comparator for operation. Clearing the CON bit disables the comparator, minimizing current consumption.

The CCH<1:0> bits in the CMxCON register (CMxCON<1:0>) direct either one of three analog input pins, or the Internal Reference Voltage (VIRV), to the comparator, VIN-. Depending on the comparator operating mode, either an external or internal voltage reference may be used. The analog signal present at VIN- is compared to the signal at VIN+ and the digital output of the comparator is adjusted accordingly.

The external reference is used when CREF = `0` (CMxCON<2>) and VIN+ is connected to the CxINA pin. When external voltage references are used, the comparator module can be configured to have the reference sources externally. The reference signal must be between VSS and VDD, and can be applied to either pin of the comparator.

The comparator module also allows the selection of an internally generated voltage reference (CVREF) from the comparator voltage reference module. This module is described in more detail in **Section 23.0 "Comparator Module"**. The reference from the comparator voltage reference module is only available when CREF = `1`. In this mode, the internal voltage reference is applied to the comparator's VIN+ pin.

Note:	The comparator input pin selected by CCH<1:0> must be configured as an input by setting both the corresponding TRIS bits and PCFG bits in the ANCON1 register.

23.5.2 COMPARATOR ENABLE AND OUTPUT SELECTION

The comparator outputs are read through the CMSTAT register. The CMSTAT<0> reads the Comparator 1 output, CMSTAT<1> reads the Comparator 2 output and CMSTAT<2> reads the Comparator 3 output. These bits are read-only.

The comparator outputs may also be directly output to the RPn 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.

By default, the comparator's output is at logic high whenever the voltage on VIN+ is greater than on VIN-. The polarity of the comparator outputs can be inverted using the CPOL bit (CMxCON<5>).

The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications, as discussed in **Section 23.2 "Comparator Operation"**.

23.6 Comparator Interrupts

The comparator interrupt flag is set whenever any of the following occurs:

- Low-to-high transition of the comparator output
- High-to-low transition of the comparator output
- Any change in the comparator output

The comparator interrupt selection is done by the EVPOL<1:0> bits in the CMxCON register (CMxCON<4:3>).

In order to provide maximum flexibility, the output of the comparator may be inverted using the CPOL bit in the CMxCON register (CMxCON<5>). This is functionally identical to reversing the inverting and non-inverting inputs of the comparator for a particular mode.

An interrupt is generated on the low-to-high or high-to-low transition of the comparator output. This mode of interrupt generation is dependent on EVPOL<1:0> in the CMxCON register. 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<1:0>, to determine the actual change that occurred. The CMxIF bits (PIR2<6:5>) are the Comparator x Interrupt Flags. The CMxIF bits must be reset by clearing them. Since it is also possible to write a ‘`1`’ to this register, a simulated interrupt may be initiated.

Table 23-2 provides the interrupt generation corresponding to comparator input voltages and EVPOL bit settings.

Both the CMxIE bits (PIE2<6:5>) and the PEIE bit (INTCON<6>) must be set to enable the interrupt. In addition, the GIE bit (INTCON<7>) must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMxIF bits will still be set if an interrupt condition occurs.

Figure 23-3 provides a simplified diagram of the interrupt section.

TABLE 23-2: COMPARATOR INTERRUPT GENERATION

CPOL	EVPOL<1:0>	Comparator Input Change	COUTx Transition	Interrupt Generated
0	00	VIN+ > VIN-	Low-to-High	No
		VIN+ < VIN-	High-to-Low	No
	01	VIN+ > VIN-	Low-to-High	Yes
		VIN+ < VIN-	High-to-Low	No
	10	VIN+ > VIN-	Low-to-High	No
		VIN+ < VIN-	High-to-Low	Yes
	11	VIN+ > VIN-	Low-to-High	Yes
		VIN+ < VIN-	High-to-Low	Yes
1	00	VIN+ > VIN-	High-to-Low	No
		VIN+ < VIN-	Low-to-High	No
	01	VIN+ > VIN-	High-to-Low	No
		VIN+ < VIN-	Low-to-High	Yes
	10	VIN+ > VIN-	High-to-Low	Yes
		VIN+ < VIN-	Low-to-High	No
	11	VIN+ > VIN-	High-to-Low	Yes
		VIN+ < VIN-	Low-to-High	Yes

23.7 Comparator Operation During Sleep

When a comparator is active and the device is placed in Sleep mode, the comparator remains active and the interrupt is functional, if enabled. This interrupt will wake-up the device from Sleep mode when enabled. Each operational comparator will consume additional current. To minimize power consumption while in Sleep mode, turn off the comparators (CON = `0`) before entering Sleep. If the device wakes up from Sleep, the contents of the CMxCON register are not affected.

23.8 Effects of a Reset

A device Reset forces the CMxCON registers to their Reset state. This forces both comparators and the voltage reference to the OFF state.

TABLE 23-3: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
PIR2	OSCFIF	CM2IF	CM1IF	—	BCL1IF	HLVDIF	TMR3IF	CCP2IF
PIR5	—	—	CM3IF	TMR8IF	TMR6IF	TMR5IF	TMR5GIF	TMR1GIF
PIE2	OSCFIE	CM2IE	CM1IE	—	BCL1IE	HLVDIE	TMR3IE	CCP2IE
PIE5	—	—	CM3IE	TMR8IE	TMR6IE	TMR5IE	TMR5GIE	TMR1GIE
IPR2	OSCFIP	CM2IP	CM1IP	—	BCL1IP	HLVDIP	TMR3IP	CCP2IP
IPR5	—	—	CM3IP	TMR8IP	TMR6IP	TMR5IP	TMR5GIP	TMR1GIP
CMxCON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0
CMSTAT	—	—	—	—	—	COUT3	COUT2	COUT1
ANCON0	PCFG7(1)	PCFG6(1)	PCFG5(1)	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
ANCON1	VBGEN	—	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
TRISA	TRISA7	TRISA6	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0

Legend: — = unimplemented, read as ‘`0`’. Shaded cells are not related to comparator operation.

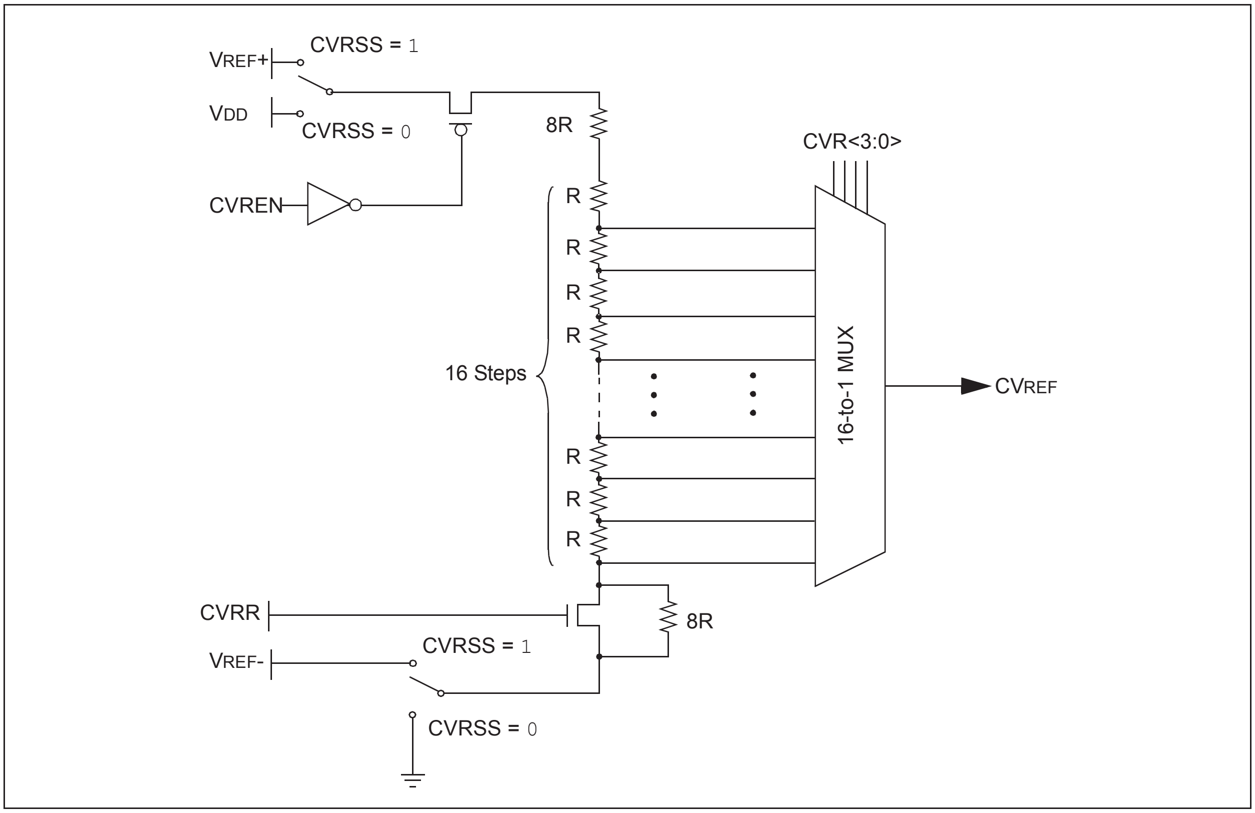
Note 1: These bits are not implemented on 28-pin devices.

24.0 COMPARATOR VOLTAGE REFERENCE MODULE

The comparator voltage reference is a 16-tap resistor ladder network that provides a selectable reference voltage. Although its primary purpose is to provide a reference for the analog comparators, it may also be used independently of them.

Figure 24-1 provides a block diagram of the module. The resistor ladder is segmented to provide two ranges of CVREF values and has a power-down function to conserve power when the reference is not being used. The module's supply reference can be provided from either device VDD/VSS or an external voltage reference.

FIGURE 24-1: COMPARATOR VOLTAGE REFERENCE BLOCK DIAGRAM


VREF+
CVRSS = 1
VDD
CVRSS = 0
8R
CVR<3:0>
CVREN
R
R
R
R
16 Steps
16-to-1 MUX
CVREF
R
R
R
CVRR
8R
CVRSS = 1
VREF-
CVRSS = 0

24.1 Configuring the Comparator Voltage Reference

The comparator voltage reference module is controlled through the CVRCON register (Register 24-1). The comparator voltage reference provides two ranges of output voltage, each with 16 distinct levels. The range to be used is selected by the CVRR bit (CVRCON<5>). The primary difference between the ranges is the size of the steps selected by the CVREF Selection bits (CVR<3:0>), with one range offering finer resolution. The equations used to calculate the output of the comparator voltage reference are as follows:

EQUATION 24-1: CALCULATING OUTPUT OF THE COMPARATOR VOLTAGE REFERENCE

When CVRR = `1` and CVRSS = `0`:
CVREF = ((CVR<3:0>)/24) x (AVDD - AVSS)
When CVRR = `0` and CVRSS = `0`:
CVREF = (CVRSRC/4) + ((CVR<3:0>)/32) x (AVDD - AVSS)
When CVRR = `1` and CVRSS = `1`:
CVREF = ((CVR<3:0>)/24) x ((VREF+) – VREF-)
When CVRR = `0` and CVRSS = `1`:
CVREF = (CVRSRC/4) + ((CVR<3:0>)/32) x ((VREF+) – VREF-)

The comparator reference supply voltage can come from either VDD and VSS, or the external VREF+ and VREF- that are multiplexed with RA2 and RA3. The voltage source is selected by the CVRSS bit (CVRCON<4>).

The settling time of the comparator voltage reference must be considered when changing the CVREF output (see Table 30-2 in **Section 30.0 "Electrical Characteristics"**).

REGISTER 24-1: CVRCON: COMPARATOR VOLTAGE REFERENCE CONTROL REGISTER (F53h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVREN	CVROE[1]	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0
bit 7							bit 0

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

bit 7 **CVREN:** Comparator Voltage Reference Enable bit
`1` = CVREF circuit is powered on
`0` = CVREF circuit is powered down

bit 6 **CVROE:** Comparator VREF Output Enable bit[1]
`1` = CVREF voltage level is also output on the RA2/AN2//C2INB/C1IND/C3INB/VREF-/CVREF pin
`0` = CVREF voltage is disconnected from the RA2/AN2//C2INB/C1IND/C3INB/VREF-/CVREF pin

bit 5 **CVRR:** Comparator VREF Range Selection bit
`1` = 0 to 0.667 CVRSRC with CVRSRC/24 step size (low range)
`0` = 0.25 CVRSRC to 0.75 CVRSRC with CVRSRC/32 step size (high range)

bit 4 **CVRSS:** Comparator VREF Source Selection bit
`1` = Comparator reference source, CVRSRC = (VREF+) – (VREF-)
`0` = Comparator reference source, CVRSRC = AVDD – AVSS

bit 3-0 **CVR<3:0>:** Comparator VREF Value Selection bits (0 ≤ (CVR<3:0>) ≤ 15)
When CVRR = `1`:
CVREF = ((CVR<3:0>)/24) • (CVRSRC)
When CVRR = `0`:
CVREF = (CVRSRC/4) + ((CVR<3:0>)/32) • (CVRSRC)

Note 1: CVROE overrides the TRIS bit setting.

24.2 Voltage Reference Accuracy/Error

The full range of voltage reference cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (see Figure 24-1) keep CVREF from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the CVREF output changes with fluctuations in that source. The tested absolute accuracy of the voltage reference can be found in **Section 30.0 "Electrical Characteristics"**.

24.3 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RA2 pin if the CVROE bit is set. Enabling the voltage reference output onto RA2 when it is configured as a digital input will increase current consumption. Connecting RA2 as a digital output with CVRSS enabled will also increase current consumption.

The RA2 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. See Figure 24-2 for an example buffering technique.

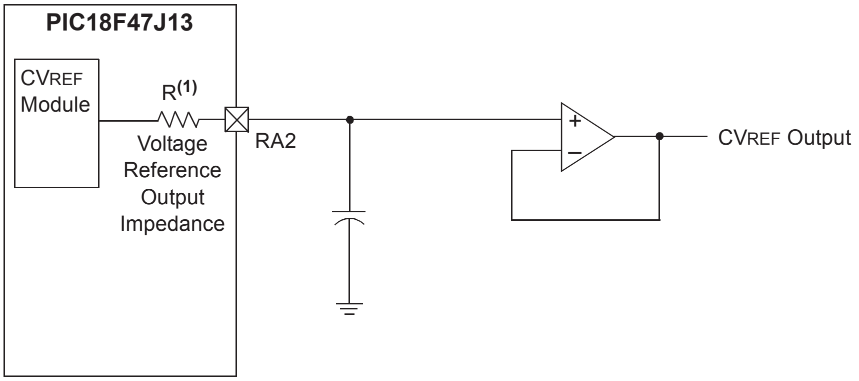
24.4 Operation During Sleep

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

24.5 Effects of a Reset

A device Reset disables the voltage reference by clearing bit, CVREN (CVRCON<7>). This Reset also disconnects the reference from the RA2 pin by clearing bit, CVROE (CVRCON<6>) and selects the high-voltage range by clearing bit, CVRR (CVRCON<5>). The CVR value select bits are also cleared.

FIGURE 24-2: COMPARATOR VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE



Note 1: R is dependent upon the Comparator Voltage Reference Configuration bits, CVRCON<5> and CVRCON<3:0>.

TABLE 24-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CVRCON	CVREN	CVROE	CVRR	CVRSS	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
TRISA	TRISA7	TRISA6	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0
ANCON0	PCFG7(1)	PCFG6(1)	PCFG5(1)	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0

Legend: — = unimplemented, read as '0', r = reserved. Shaded cells are not used with the comparator voltage reference.

Note 1: These bits are only available on 44-pin devices.

25.0 HIGH/LOW VOLTAGE DETECT (HLVD)

The High/Low-Voltage Detect (HLVD) module can be used to monitor the absolute voltage on VDD or the HLVDIN pin. This is a programmable circuit that allows the user to specify both a device voltage trip point and the direction of change from that point.

If the module detects an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution will branch to the interrupt vector address and the software can then respond to the interrupt.

The High/Low-Voltage Detect Control register (Register 25-1) 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.

Figure 25-1 provides a block diagram for the HLVD module.

REGISTER 25-1: HLVDCON: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER (ACCESS F85h)

R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	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 the voltage equals or exceeds the trip point (HLVDL<3:0>)
`0` = Event occurs when the voltage equals or falls below the trip point (HLVDL<3:0>)

bit 6 **BGVST:** Band Gap Reference Voltages Stable Status Flag bit
`1` = Indicates internal band gap voltage references is stable
`0` = Indicates internal band gap voltage reference is 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: See Table 30-8 in **Section 30.0 "Electrical Characteristics"** for specifications.

The module is enabled by setting the HLVDEN bit. Each time the module is enabled, the circuitry requires some time to stabilize. The IRVST bit is a read-only bit that indicates when the circuit is stable. The module can generate an interrupt only after the circuit is stable and IRVST is set.

The VDIRMAG bit 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.

25.1 Operation

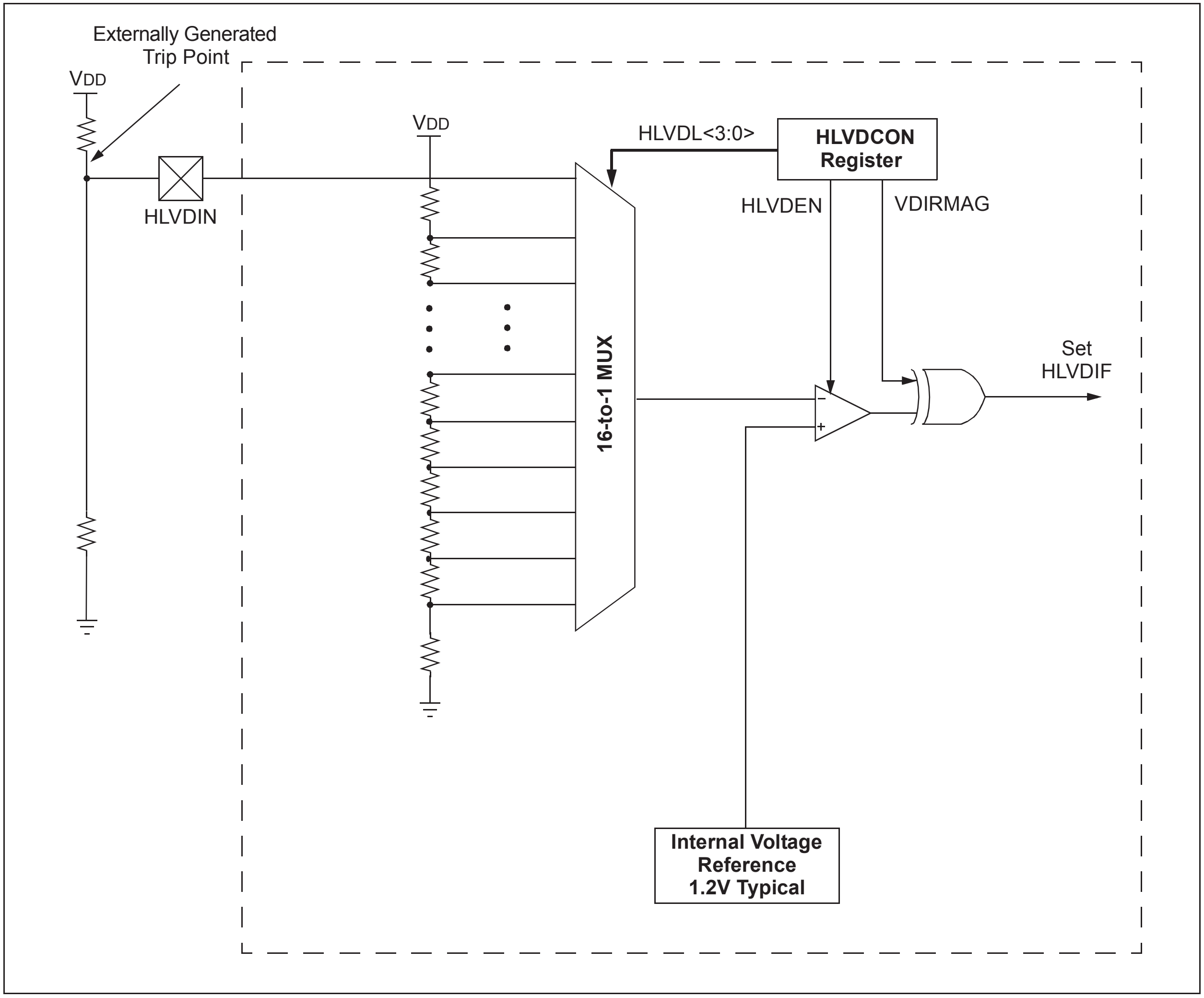
When the HLVD module is enabled, a comparator uses an internally generated reference voltage as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The “trip point” voltage is the voltage level at which the device detects 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 one of 16 values. The trip point is selected by programming the HLVDL<3:0> bits (HLVDCON<3:0>).

Additionally, the HLVD module 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 flexibility because it allows them to configure the HLVD interrupt to occur at any voltage in the valid operating range.

FIGURE 25-1: HLVD MODULE BLOCK DIAGRAM (WITH EXTERNAL INPUT)



25.2 HLVD Setup

To set up the HLVD module:

1. Disable the module by clearing the HLVDEN bit (HLVDCON<4>).
2. Write the value to the HLVDL<3:0> bits that select the desired HLVD trip point.
3. Set the VDIRMAG bit to detect one of the following:
   - High voltage (VDIRMAG = `1`)
   - Low voltage (VDIRMAG = `0`)
4. Enable the HLVD module by setting the HLVDEN bit.
5. Clear the HLVD Interrupt Flag, HLVDIF (PIR2<2>), which may have been set from a previous interrupt.
6. If interrupts are desired, enable the HLVD interrupt by setting the HLVDIE and GIE/GIEH bits (PIE2<2> and INTCON<7>).

   An interrupt will not be generated until the IRVST bit is set.

25.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled, and will consume static current. The total current consumption, when enabled, is specified in electrical specification parameter, D022B (ΔI_{HLVD}) (**Section 30.2 "DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial)"**).

Depending on the application, the HLVD module does not need to operate constantly. To decrease the current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After doing the check, the HLVD module may be disabled.

25.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification parameter D420 (see Table 30-8 in **Section 30.0 "Electrical Characteristics"**), may be used by other internal circuitry, such as the Programmable Brown-out Reset (BOR).

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, T_{IRVST}, is an interval that is independent of device clock speed. It is specified in electrical specification, parameter 36 (Table 30-15).

The HLVD interrupt flag is not enabled until T_{IRVST} 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. Refer to Figure 25-2 or Figure 25-3.

FIGURE 25-2: LOW-VOLTAGE DETECT OPERATION (VDIRMAG = 0)

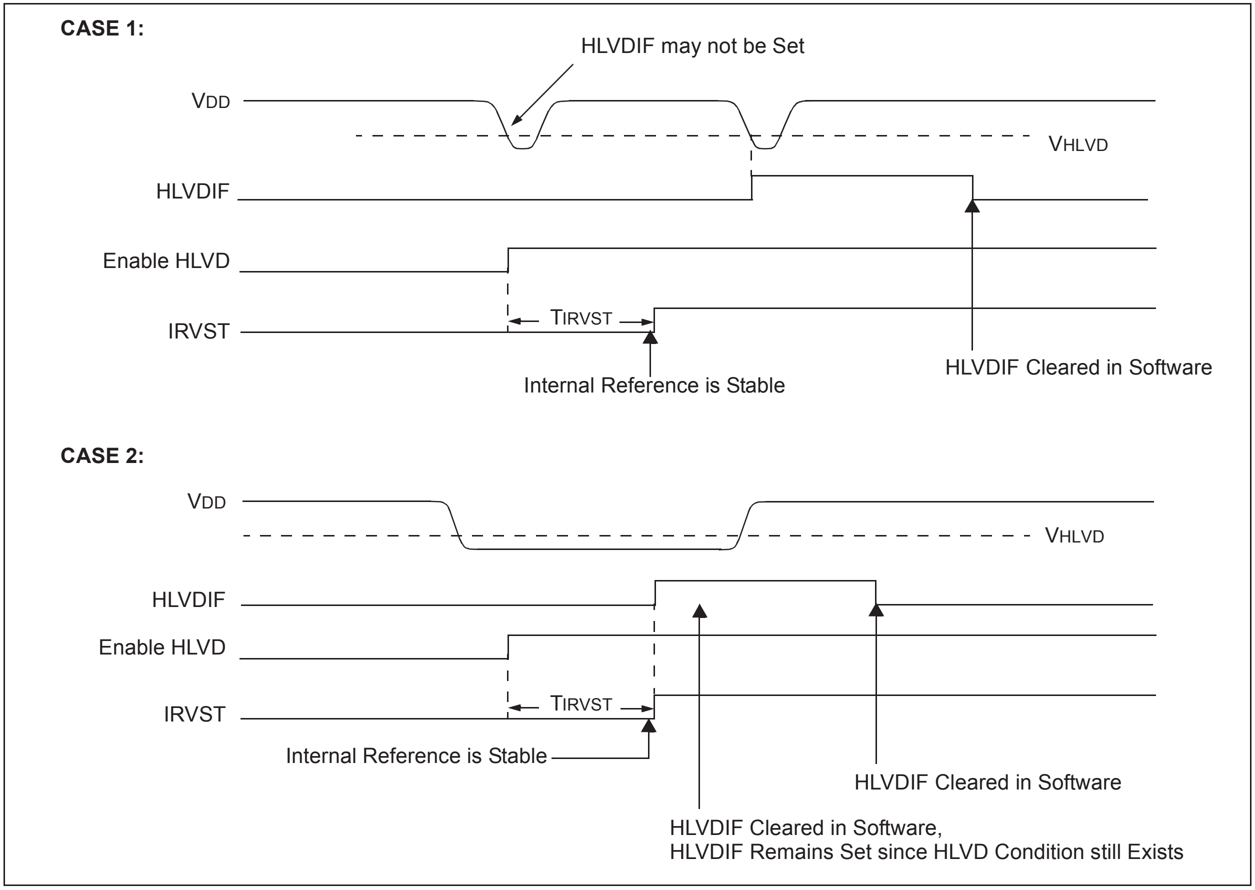
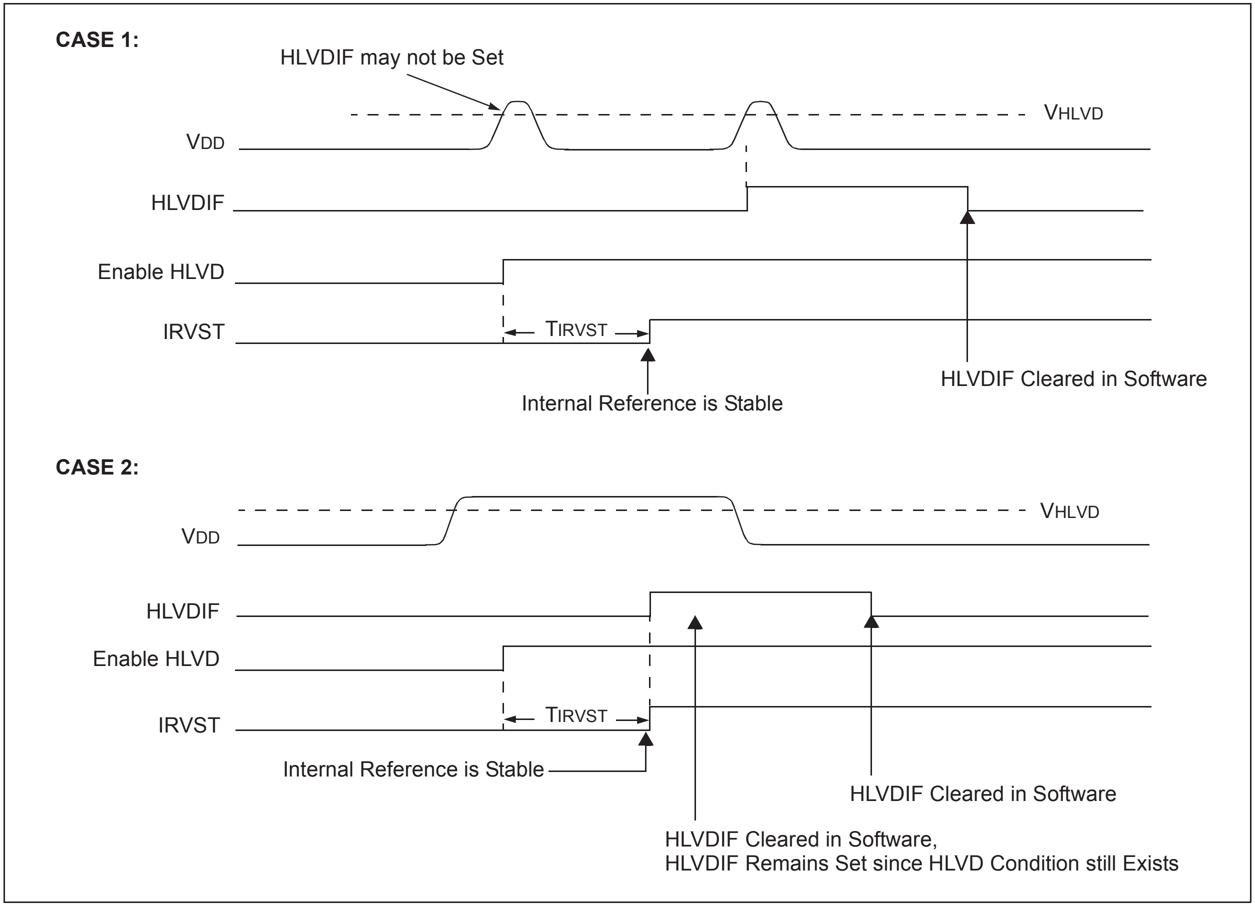

CASE 1:
HLVDIF may not be Set
VDD
VHLVD
HLVDIF
Enable HLVD
TIRVST
IRVST
Internal Reference is Stable
HLVDIF Cleared in Software
CASE 2:
VDD
VHLVD
HLVDIF
Enable HLVD
TIRVST
IRVST
Internal Reference is Stable
HLVDIF Cleared in Software
HLVDIF Cleared in Software,
HLVDIF Remains Set since HLVD Condition still Exists

FIGURE 25-3: HIGH-VOLTAGE DETECT OPERATION (VDIRMAG = `1`)


CASE 1:
HLVDIF may not be Set
VHLVD
VDD
HLVDIF
Enable HLVD
IRVST
TIRVST
HLVDIF Cleared in Software
Internal Reference is Stable
CASE 2:
VHLVD
VDD
HLVDIF
Enable HLVD
IRVST
TIRVST
Internal Reference is Stable
HLVDIF Cleared in Software
HLVDIF Cleared in Software,
HLVDIF Remains Set since HLVD Condition still Exists

25.5 Applications

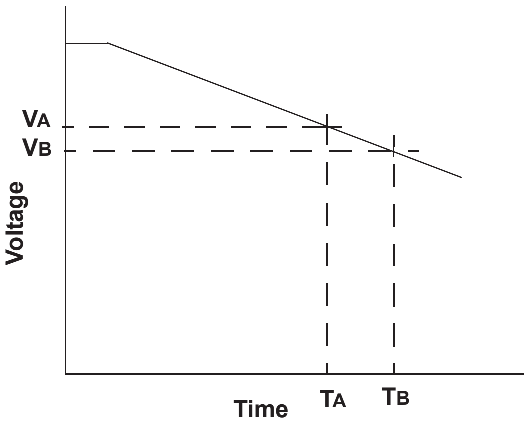
In many applications, it is desirable to have the ability to detect a drop below, or rise above, a particular threshold.

For general battery applications, Figure 25-4 provides 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 ISR, which would allow the application to perform “housekeeping tasks” and perform a controlled shutdown before the device voltage exits the valid operating range at TB.

The HLVD, thus, would give the application a time window, represented by the difference between TA and TB, to safely exit.

FIGURE 25-4: TYPICAL HIGH/LOW-VOLTAGE DETECT APPLICATION



Legend: VA = HLVD trip point
VB = Minimum valid device operating voltage

25.6 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.

25.7 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

TABLE 25-1: 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	CM2IF	CM1IF	—	BCL1IF	HLVDIF	TMR3IF	CCP2IF
PIE2	OSCFIE	CM2IE	CM1IE	—	BCL1IE	HLVDIE	TMR3IE	CCP2IE
IPR2	OSCFIP	CM2IP	CM1IP	—	BCL1IP	HLVDIP	TMR3IP	CCP2IP

Legend: — = unimplemented, read as '`0`'. Shaded cells are unused by the HLVD module.

26.0 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 be used to precisely measure time, measure capacitance, measure relative changes in capacitance or generate output pulses with a specific time delay. The CTMU is ideal for interfacing with capacitive-based sensors.

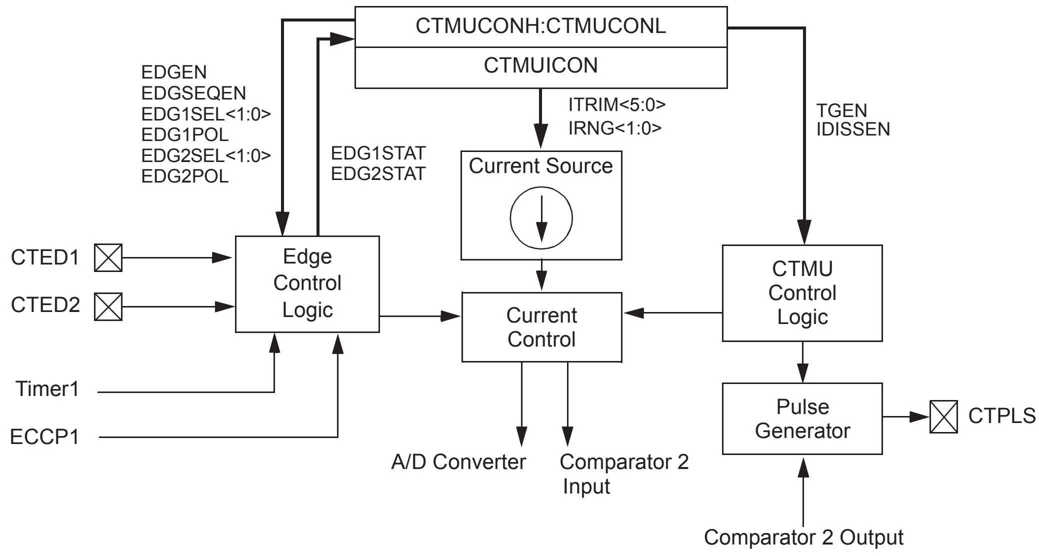
The module includes the following key features:

- Up to 13 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 1 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 13 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, Timer1 or Output Compare Module 1.

Figure 26-1 provides a block diagram of the CTMU.

FIGURE 26-1: CTMU BLOCK DIAGRAM


CTMUCONH:CTMUCONL
CTMUICON
EDGEN
EDGSEQEN
EDG1SEL<1:0>
EDG1POL
EDG2SEL<1:0>
EDG2POL
EDG1STAT
EDG2STAT
ITRIM<5:0>
IRNG<1:0>
TGEN
IDISSEN
Current Source
CTED1
CTED2
Edge Control Logic
Current Control
CTMU Control Logic
Timer1
ECCP1
Pulse Generator
CTPLS
A/D Converter
Comparator 2 Input
Comparator 2 Output

26.1 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 is then a measurement of the capacitance of the circuit. 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 then 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.

26.1.1 THEORY OF OPERATION

The operation of the CTMU is based on the following equation for charge:

$$I = C \bullet \frac{dV}{dT}$$

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:

$$I \cdot t = C \cdot V.$$

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:

$$t = (C \cdot V)/I$$

or by:

$$C = (I \cdot t)/V$$

using a fixed time that the current source is applied to the circuit.

26.1.2 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 ‘`01`’ 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. Note that 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 ‘`100001`’ is the maximum negative adjustment (approximately -62%) and ‘`011111`’ is the maximum positive adjustment (approximately +62%).

26.1.3 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), Timer1 or Output Compare Module 1. 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> and <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>).

26.1.4 EDGE STATUS

The CTMUCONL 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 is the same as 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 (but not both) of the status bits is set, and shuts current 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 is also the user’s application to manually enable or disable the current source. Setting either one (but not both) of the bits enables the current source. Setting or clearing both bits at once disables the source.

26.1.5 INTERRUPTS

The CTMU sets its interrupt flag (PIR3<2>) whenever the current source is enabled, then disabled. An interrupt is generated only if the corresponding interrupt enable bit (PIE3<2>) 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.

26.2 CTMU Module Initialization

The following sequence is a general guideline used to initialize the CTMU module:

1. Select the current source range using the IRNG bits (CTMUICON<1:0>).
2. Adjust the current source trim using the ITRIM 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>).
4. Configure the input polarities for the edge inputs using the EDG1POL and EDG2POL 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 (CTMUCONH<4>). The default mode is Time/Capacitance Measurement.
7. Discharge the connected circuit by setting the IDISSEN bit (CTMUCONH<1>). After waiting a sufficient time for the circuit to discharge, clear IDISSEN.
8. Disable the module by clearing the CTMUEN bit (CTMUCONH<7>).
9. Enable the module by setting the CTMUEN bit.
10. Clear the Edge Status bits: EDG2STAT and EDG1STAT (CTMUCONL<1:0>). Both bits should be cleared simultaneously, if possible, to avoid re-enabling the CTMU current source.
11. Enable both edge inputs by setting the EDGEN bit (CTMUCONH<3>).

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, both Timer1 and the Output Compare/PWM1 module 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.

26.3 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 this type of application would include 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, and the circuit being measured needs calibration to measure and/or nullify all other capacitance other than that to be measured.

26.3.1 CURRENT SOURCE CALIBRATION

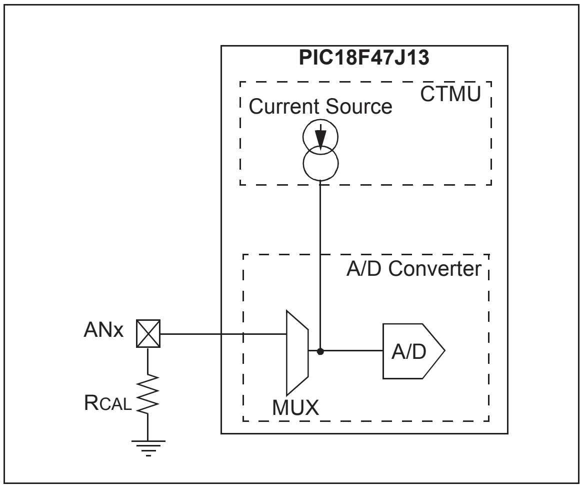
The current source on board the CTMU module has a range of ±62% nominal for each of three current ranges. Therefore, for precise measurements, it is possible to measure and adjust this current source by placing a high-precision resistor, R_{CAL}, onto an unused analog channel. An example circuit is shown in Figure 26-2. The current source measurement is performed using the following steps:

1. Initialize the A/D Converter.
2. Initialize the CTMU.
3. Enable the current source by setting EDG1STAT (CTMUCONL<0>).
4. Issue a time delay for voltage across R_{CAL} to stabilize and the ADC sample/hold capacitor to charge.
5. Perform A/D conversion.
6. Calculate the present source current using $I = V/R_{CAL}$, where R_{CAL} is a high-precision resistance and V is measured by performing an A/D conversion.

The CTMU current source may be trimmed with the trim bits in CTMUICON using an iterative process to get an exact desired current. Alternatively, the nominal value without adjustment may be used; it may be stored by the software for use in all subsequent capacitive or time measurements.

To calculate the optimal value for R_{CAL}, the nominal current must be chosen. 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 $R_{CAL} = 2.31\text{V}/0.55\ \mu\text{A}$, for a value of 4.2 MΩ. Similarly, if the current source is chosen to be 5.5 µA, R_{CAL} would be 420,000Ω, and 42,000Ω if the current source is set to 55 µA.

FIGURE 26-2: CTMU CURRENT SOURCE CALIBRATION CIRCUIT



A value of 70% of full-scale voltage is chosen to make sure that the A/D Converter is in a range that is well above the noise floor. Keep in mind that if an exact current is chosen that is to incorporate the trimming bits from CTMUICON, the resistor value of R_{CAL} may need to be adjusted accordingly. R_{CAL} may also be adjusted to allow for available resistor values. R_{CAL} should be of the highest precision available, keeping in mind the amount of precision needed for the circuit that the CTMU will be used to measure. A recommended minimum would be 0.1% tolerance.

The following examples show one typical method for performing a CTMU current calibration. Example 26-1 demonstrates how to initialize the A/D Converter and the CTMU. This routine is typical for applications using both modules. Example 26-2 demonstrates one method for the actual calibration routine.

EXAMPLE 26-1: 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,


    //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 = 0xFB;
    // ANCON1
    ANCON1 = 0x1F;

    // ADCON1
    ADCON1bits.ADFM=1;                // Result format 1= Right justified
    ADCON1bits.ADCAL=0;               // Normal A/D conversion operation
    ADCON1bits.ACQT=1;                // Acquisition time 7 = 20TAD 2 = 4TAD 1=2TAD
    ADCON1bits.ADCS=2;                // Clock conversion bits 6= FOSC/64 2=FOSC/32


        // ADCON0
    ADCON0bits.VCFG0 =0;              // Vref+ = AVdd
    ADCON0bits.VCFG1 =0;              // Vref- = AVss
    ADCON0bits.CHS=2;                 // Select ADC channel

    ADCON0bits.ADON=1;                // Turn on ADC

}
```

EXAMPLE 26-2: 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
       CTMUCONLbits.EDG1STAT = 0;            // Set Edge status bits to zero
       CTMUCONLbits.EDG2STAT = 0;
    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

}
```

26.3.2 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. The measurement is then performed using the following steps:

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:

$$C_{OFFSET} = C_{STRAY} + C_{AD} = (I \cdot t)/V$$

where I is known from the current source measurement step, t is a fixed delay and 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 need to be used 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, then 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, then t would be:

$$(4 \text{ pF} + 11 \text{ pF}) \bullet 2.31\text{V}/0.55\ \mu\text{A}$$

or 63 μs.

See Example 26-3 for a typical routine for CTMU capacitance calibration.

EXAMPLE 26-3: 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
       CTMUCONLbits.EDG1STAT = 0;           // Set Edge status bits to zero
       CTMUCONLbits.EDG2STAT = 0;
    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;
}
```

26.4 Measuring Capacitance with the CTMU

There are two separate methods of measuring capacitance with the CTMU. The first is the absolute method, in which the actual capacitance value is desired. The second is the relative method, in which the actual capacitance is not needed, rather an indication of a change in capacitance is required.

26.4.1 ABSOLUTE CAPACITANCE MEASUREMENT

For absolute capacitance measurements, both the current and capacitance calibration steps found in **Section 26.3 "Calibrating the CTMU Module"** should be followed. Capacitance measurements are then performed using the following steps:

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 (see **Section 26.3.1 "Current Source Calibration"**), T is a fixed delay and V is measured by performing an A/D conversion.
8. Subtract the stray and A/D capacitance (C_{OFFSET} from **Section 26.3.2 "Capacitance Calibration"**) from C_{TOTAL} to determine the measured capacitance.

26.4.2 CAPACITIVE TOUCH SENSE USING RELATIVE CHARGE MEASUREMENT

An application may not require precise capacitance measurements. For example, when detecting a valid press of a capacitance-based touch sense button, detecting a relative change of capacitance is of interest. In this type of application, when the touch sense pad is not being pressed, the total capacitance is the capacitance of the combination of the board traces, the A/D Converter, etc. A larger voltage will be measured by the A/D Converter. When the touch sense pad is pressed, the total capacitance is larger due to the addition of the capacitance of the human body, therefore, a smaller voltage will be measured by the A/D Converter.

Detecting capacitance changes is easily accomplished with the CTMU using these steps:

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. Note that in this case, no calibration of the current source or circuit capacitance measurement is needed. See Example 26-4 for a sample software routine for a capacitive touch switch.

EXAMPLE 26-4: 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
    CTMUCONLbits.EDG1STAT = 0;          // Set Edge status bits to zero
    CTMUCONLbits.EDG2STAT = 0;

    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;
    }
}
```

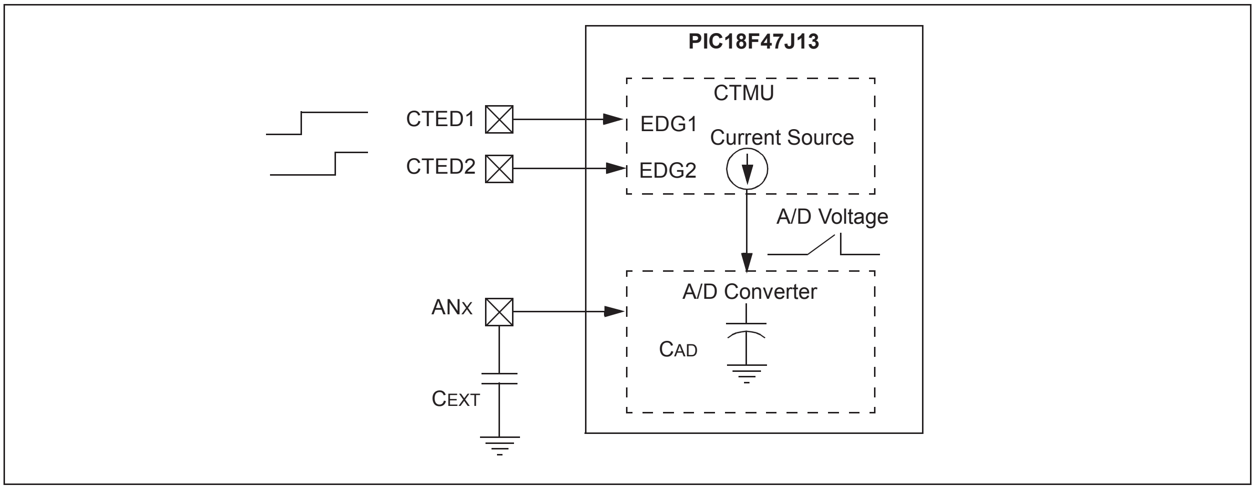
26.5 Measuring Time with the CTMU Module

Time can be precisely measured after the ratio (*C/I*) is measured from the current and capacitance calibration step by following these steps:

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 26.3.1 "Current Source Calibration"**), C is calculated in the capacitance calibration step (**Section 26.3.2 "Capacitance Calibration"**) and V is measured by performing the A/D conversion.

It is assumed that the time measured is small enough that the capacitance, $C_{AD} + C_{EXT}$, 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 (4-5 pF). To measure longer time intervals, an external capacitor may be connected to an A/D channel and this channel selected when making a time measurement.

FIGURE 26-3: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR TIME MEASUREMENT



26.6 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 26-4 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. C_{DELAY} 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 26.3.1 "Current Source Calibration"**) and V is the internal reference voltage (CVREF).

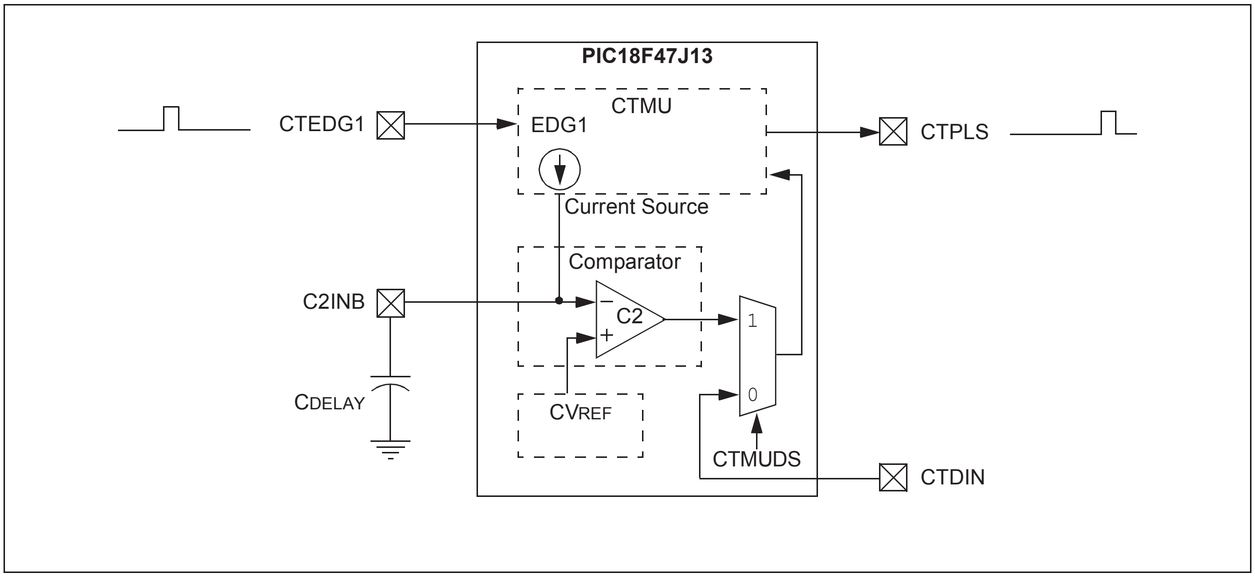
An example use of the external capacitor feature is interfacing with variable capacitive-based sensors, such as a humidity sensor. As the humidity varies, the pulse-width output on CTPLS will vary. An example use of the CTDIN feature is interfacing with a digital sensor. The CTPLS output pin can be connected 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 C_{DELAY} 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 26-4: TYPICAL CONNECTIONS AND INTERNAL CONFIGURATION FOR PULSE DELAY GENERATION



26.7 Operation During Sleep/Idle Modes

26.7.1 SLEEP MODE AND DEEP SLEEP MODES

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.

26.7.2 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.

26.8 Effects of a Reset on CTMU

Upon Reset, all registers of the CTMU are cleared. This leaves the CTMU module disabled; its current source is turned off and all configuration options return 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, and should be properly discharged before the CTMU makes subsequent attempts to make a measurement. The circuit is discharged by setting and then clearing the IDISSEN bit (CTMUCONH<1>) while the A/D Converter is connected to the appropriate channel.

26.9 Registers

There are three control registers for the CTMU:

- CTMUCONH
- CTMUCONL
- CTMUICON

The CTMUCONH and CTMUCONL registers (Register 26-1 and Register 26-2) 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 26-3) has bits for selecting the current source range and current source trim.

REGISTER 26-1: CTMUCONH: CTMU CONTROL REGISTER HIGH (ACCESS FB3h)

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:** CTMU Special Event Trigger bit
1 = CTMU Special Event Trigger is enabled
0 = CTMU Special Event Trigger is disabled

REGISTER 26-2: CTMUCONL: CTMU CONTROL REGISTER LOW (ACCESS FB2h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x	R/W-x
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 output compare module
`00` = Timer1 module

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 output compare module
`00` = Timer1 module

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 26-3: CTMUICON: CTMU CURRENT CONTROL REGISTER (ACCESS FB1h)

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 (+62% typ.) from nominal current
`011110`
.
.
.
`000001` = Minimum positive change (+2% typ.) from nominal current
`000000` = Nominal current output specified by IRNG<1:0>
`111111` = Minimum negative change (-2% typ.) from nominal current
.
.
.
`100010`
`100001` = Maximum negative change (-62% typ.) from nominal current

bit 1-0 **IRNG<1:0>:** Current Source Range Select bits

`11` = 100 × Base current
`10` = 10 × Base current
`01` = Base current level (0.55 μA nominal)
`00` = Current source disabled

REGISTER 26-4: ODCON3: PERIPHERAL OPEN-DRAIN CONTROL REGISTER 3 (BANKED F40h)

R/W-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
CTMUDS	—	—	—	—	—	SPI2OD	SPI1OD
bit 7							bit 0

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

bit 7 **CTMUDS:** CTMU Pulse Delay Enable bit

`1` = Pulse delay input for CTMU enabled on pin RA1

bit 6-2 **Unimplemented:** Read as ‘`0`’

bit 1 **SPI2OD:** SPI2 Open-Drain Output Enable bit

`1` = Open-drain capability is enabled
`0` = Open-drain capability is disabled

bit 0 **SPI1OD:** SPI1 Open-Drain Output Enable bit

`1` = Open-drain capability is enabled
`0` = Open-drain capability is disabled

TABLE 26-1: 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	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP
ODCON3	CTMUDS	—	—	—	—	—	SPI2OD	SPI1OD

Legend: — = unimplemented, read as '0'. Shaded cells are not used during CTMU module.

27.0 SPECIAL FEATURES OF THE CPU

PIC18F47J13 Family devices include several features intended to maximize reliability and minimize cost through elimination of external components. These are:

- Oscillator Selection
- Resets:
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Fail-Safe Clock Monitor (FSCM)
- Two-Speed Start-up
- Code Protection
- 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 PIC18F47J13 Family of devices has a configurable Watchdog Timer (WDT), which is controlled in software.

The inclusion of an internal RC oscillator also provides the additional benefits of a Fail-Safe Clock Monitor (FSCM) and Two-Speed Start-up. FSCM provides for background monitoring of the peripheral clock and automatic switchover in the event of its failure. Two-Speed Start-up enables code to be executed almost immediately on start-up, while the primary clock source completes its start-up delays.

All of these features are enabled and configured by setting the appropriate Configuration register bits.

27.1 Configuration Bits

The Configuration bits can be programmed to select various device configurations. The configuration data is stored in the last four words of Flash program memory; Figure 6-1 depicts this. The configuration data gets loaded into the volatile Configuration registers, CONFIG1L through CONFIG4H, which are readable and mapped to program memory, starting at location 300000h.

Table 27-2 provides a complete list of the Configuration bits and Device IDs. A detailed explanation of the various bit functions is provided in Register 27-1 through Register 27-6.

27.1.1 CONSIDERATIONS FOR CONFIGURING THE PIC18F47J13 FAMILY DEVICES

Unlike some previous PIC18 microcontrollers, PIC18F47J13 Family devices do not use persistent memory registers to store configuration information. The Configuration registers, CONFIG1L through CONFIG4H, are implemented as volatile memory.

Immediately after power-up, or after a device Reset, the microcontroller hardware automatically loads the CONFIG1L through CONFIG4L registers with configuration data stored in nonvolatile Flash program memory. The last four words of Flash program memory, known as the Flash Configuration Words (FCW), are used to store the configuration data.

Table 27-1 provides the Flash program memory, which will be loaded into the corresponding Configuration register.

When creating applications for these devices, users should always specifically allocate the location of the FCW for configuration data. This is to make certain that program code is not stored in this address when the code is compiled.

The four Most Significant bits (MSb) of the FCW, corresponding to CONFIG1H, CONFIG2H, CONFIG3H and CONFIG4H, should always be programmed to '`1111`'. This makes these FCWs appear to be `NOP` instructions in the remote event that their locations are ever executed by accident.

The four MSbs of the CONFIG1H, CONFIG2H, CONFIG3H and CONFIG4H registers are not implemented, so writing '`1`'s to their corresponding FCW has no effect on device operation.

To prevent inadvertent configuration changes during code execution, the Configuration registers, CONFIG1L through CONFIG4L, are loaded only once per power-up or Reset cycle. User's firmware can still change the configuration by using self-reprogramming to modify the contents of the FCW.

Modifying the FCW will not change the active contents being used in the CONFIG1L through CONFIG4H registers until after the device is reset.

TABLE 27-1: MAPPING OF THE FLASH CONFIGURATION WORDS TO THE CONFIGURATION REGISTERS

Configuration Register (Volatile)	Configuration Register Address	Flash Configuration Byte Address
CONFIG1L	300000h	XXXF8h
CONFIG1H	300001h	XXXF9h
CONFIG2L	300002h	XXXFAh
CONFIG2H	300003h	XXXFBh
CONFIG3L	300004h	XXXFCh
CONFIG3H	300005h	XXXFDh
CONFIG4L	300006h	XXXFEh
CONFIG4H	300007h	XXXFFh

TABLE 27-2: CONFIGURATION BITS AND DEVICE IDs

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprog. Value(1)
300000h	CONFIG1L	$\overline{\text{DEBUG}}$	XINST	STVREN	CFGPLLEN	PLLDIV2	PLLDIV1	PLLDIV0	WDTEN	`1111 1111`
300001h	CONFIG1H	—(2)	—(2)	—(2)	—(2)	—	CP0	—	—	`1111 -1--`
300002h	CONFIG2L	IESO	FCMEN	CLKOEC	SOSCSEL1	SOSCSEL0	FOSC2	FOSC1	FOSC0	`1111 1111`
300003h	CONFIG2H	—(2)	—(2)	—(2)	—(2)	WDTPS3	WDTPS2	WDTPS1	WDTPS0	`1111 1111`
300004h	CONFIG3L	DSWDTPS3	DSWDTPS2	DSWDTPS1	DSWDTPS0	DSWDTEN	DSBOREN	RTCOSC	DSWDTOSC	`1111 1111`
300005h	CONFIG3H	—(2)	—(2)	—(2)	—(2)	MSSPMSK	PLLSEL	ADCSEL	IOL1WAY	`1111 1111`
300006h	CONFIG4L	WPCFG	WPFP6	WPFP5	WPFP4	WPFP3	WPFP2	WPFP1	WPFP0	`1111 1111`
300007h	CONFIG4H	—(2)	—(2)	—(2)	—(2)	—	—	WPEND	WPDIS	`1111 --11`
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	`xx1x xxxx`(3)
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	`0101 10x1`(3)

Legend: `x` = unknown, `u` = unchanged, `-` = unimplemented. Shaded cells are unimplemented, read as '`0`'.

Note 1: Values reflect the unprogrammed state as received from the factory and following Power-on Resets. In all other Reset states, the configuration bytes maintain their previously programmed states.

2: The value of these bits in program memory should always be programmed to '`1`'. This ensures that the location is executed as a `NOP` if it is accidentally executed.

3: See Register 27-9 and Register 27-10 for DEVID values. These registers are read-only and cannot be programmed by the user.

REGISTER 27-1: CONFIG1L: CONFIGURATION REGISTER 1 LOW (BYTE ADDRESS 300000h)

R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1
$\overline{\text{DEBUG}}$	XINST	STVREN	$\overline{\text{CFGPLLEN}}$	PLLDIV2	PLLDIV1	PLLDIV0	WDTEN
bit 7							bit 0

Legend:

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

bit 7 **$\overline{\text{DEBUG}}$:** Background Debugger Enable bit
`1` = Background debugger is disabled; RB6 and RB7 are configured as general purpose I/O pins
`0` = Background debugger is enabled; RB6 and RB7 are dedicated to In-Circuit Debug

bit 6 **XINST:** Extended Instruction Set Enable bit
`1` = Instruction set extension and Indexed Addressing mode are enabled
`0` = Instruction set extension and Indexed Addressing mode are disabled

bit 5 **STVREN**: Stack Overflow/Underflow Reset Enable bit
`1` = Reset on stack overflow/underflow is enabled
`0` = Reset on stack overflow/underflow is disabled

bit 4 **$\overline{\text{CFGPLLEN}}$:** PLL Enable bit
`1` = PLL is disabled until firmware sets the PLLEN bit at run time (OSCTUNE<6>)
`0` = PLL is enabled at start-up

bit 3-1 **PLLDIV<2:0>:** Oscillator Selection bits
Divider must be configured to provide a 4 MHz input into the 96 MHz PLL. These bits have no effect on 4x PLL operation.
`111` = No divide – oscillator is used directly (4 MHz input)
`110` = Oscillator divided by 2 (8 MHz input)
`101` = Oscillator divided by 3 (12 MHz input)
`100` = Oscillator divided by 4 (16 MHz input)
`011` = Oscillator divided by 5 (20 MHz input)
`010` = Oscillator divided by 6 (24 MHz input)
`001` = Oscillator divided by 10 (40 MHz input)
`000` = Oscillator divided by 12 (48 MHz input)

bit 0 **WDTEN:** Watchdog Timer Enable bit
`1` = WDT is enabled
`0` = WDT is disabled (control is placed on the SWDTEN bit)

REGISTER 27-2: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

U-1	U-1	U-1	U-1	U-0	R/WO-1	U-1	U-1
—	—	—	—	—	CP0	—	—
bit 7							bit 0

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

bit 7-4	**Unimplemented:** Program the corresponding Flash Configuration bit to ‘`1`’
bit 3	**Unimplemented:** Maintain as ‘`0`’
bit 2	**CP0:** Code Protection bit `1` = Program memory is not code-protected `0` = Program memory is code-protected
bit 1-0	**Unimplemented:** Maintain as ‘`1`’

REGISTER 27-3: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)

R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1
IESO	FCMEN	CLKOEC	SOSCSEL1	SOSCSEL0	FOSC2	FOSC1	FOSC0
bit 7							bit 0

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

bit 7	**IESO:** Two-Speed Start-up (Internal/External Oscillator Switchover) Control bit `1` = Two-Speed Start-up is enabled `0` = Two-Speed Start-up is disabled
bit 6	**FCMEN:** Fail-Safe Clock Monitor Enable bit `1` = Fail-Safe Clock Monitor is enabled `0` = Fail-Safe Clock Monitor is disabled
bit 5	**CLKOEC:** CLKO Enable Configuration bit (Applicable only when the FOSC<2:1> bits are configured for EC or ECPLL mode.) `1` = CLKO output signal is active on the RA6 pin `0` = CLKO output is disabled
bit 4-3	**SOSCSEL<1:0>:** T1OSC/SOSC Power Selection Configuration bits `11` = High-power T1OSC/SOSC circuit is selected `10` = Digital (SCLKI) mode is enabled `01` = Low-power T1OSC/SOSC circuit is selected `00` = Reserved
bit 2-0	**FOSC<2:0>:** Oscillator Selection bits `111` = ECPLL oscillator with PLL software controlled; CLKO on RA6 `110` = EC oscillator with CLKO on RA6 `101` = HSPLL oscillator with PLL software controlled `100` = HS oscillator `011` = INTOSCPLLO, internal oscillator with PLL software controlled; CLKO on RA6; port function on RA7 `010` = INTOSCPLL, internal oscillator with PLL software controlled; port function on RA6 and RA7 `001` = INTOSCO internal oscillator block (INTRC/INTOSC) with CLKO on RA6; port function on RA7 `000` = INTOSC internal oscillator block (INTRC/INTOSC); port function on RA6 and RA7

REGISTER 27-4: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)

U-1	U-1	U-1	U-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1
—	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0
bit 7							bit 0

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

bit 7-4 **Unimplemented:** Program the corresponding Flash Configuration bit to ‘`1`’

bit 3-0 **WDTPS<3:0>:** Watchdog Timer Postscale Select bits

`1111` = 1:32,768
`1110` = 1:16,384
`1101` = 1:8,192
`1100` = 1:4,096
`1011` = 1:2,048
`1010` = 1:1,024
`1001` = 1:512
`1000` = 1:256
`0111` = 1:128
`0110` = 1:64
`0101` = 1:32
`0100` = 1:16
`0011` = 1:8
`0010` = 1:4
`0001` = 1:2
`0000` = 1:1

REGISTER 27-5: CONFIG3L: CONFIGURATION REGISTER 3 LOW (BYTE ADDRESS 300004h)

R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1
DSWDTPS3[1]	DSWDTPS2[1]	DSWDTPS1[1]	DSWDTPS0[1]	DSWDTEN[1]	DSBOREN	RTCOSC	DSWDTOSC[1]
bit 7							bit 0

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

bit 7-4 **DSWDTPS<3:0>:** Deep Sleep Watchdog Timer Postscale Select bits[1]

The DSWDT prescaler is 32. This creates an approximate base time unit of 1 ms.

`1111` = 1:2,147,483,648 (25.7 days)
`1110` = 1:536,870,912 (6.4 days)
`1101` = 1:134,217,728 (38.5 hours)
`1100` = 1:33,554,432 (9.6 hours)
`1011` = 1:8,388,608 (2.4 hours)
`1010` = 1:2,097,152 (36 minutes)
`1001` = 1:524,288 (9 minutes)
`1000` = 1:131,072 (135 seconds)
`0111` = 1:32,768 (34 seconds)
`0110` = 1:8,192 (8.5 seconds)
`0101` = 1:2,048 (2.1 seconds)
`0100` = 1:512 (528 ms)
`0011` = 1:128 (132 ms)
`0010` = 1:32 (33 ms)
`0001` = 1:8 (8.3 ms)
`0000` = 1:2 (2.1 ms)

bit 3 **DSWDTEN:** Deep Sleep Watchdog Timer Enable bit[1]

`1` = DSWDT is enabled
`0` = DSWDT is disabled

bit 2 **DSBOREN:** “F” Device Deep Sleep BOR Enable bit, “LF” Device VDD BOR Enable bit

For “F” Devices:
`1` = VDD sensing BOR is enabled in Deep Sleep
`0` = VDD sensing BOR circuit is always disabled

For “LF” Devices:
`1` = VDD sensing BOR circuit is always enabled
`0` = VDD sensing BOR circuit is always disabled

bit 1 **RTCOSC:** RTCC Reference Clock Select bit

`1` = RTCC uses T1OSC/T1CKI as the reference clock
`0` = RTCC uses INTRC as the reference clock

bit 0 **DSWDTOSC:** DSWDT Reference Clock Select bit[1]

`1` = DSWDT uses INTRC as the reference clock
`0` = DSWDT uses T1OSC/T1CKI as the reference clock

Note 1: Functions are not available on “LF” devices.

REGISTER 27-6: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

U-1	U-1	U-1	U-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1
—	—	—	—	MSSPMSK	PLLSEL	ADCSEL	IOL1WAY
bit 7							bit 0

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

bit 7-4 **Unimplemented:** Program the corresponding Flash Configuration bit to ‘`1`’

bit 3 **MSSPMSK:** MSSP 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 **PLLSEL:** PLL Circuit Selection bit

`1` = 4x PLL circuit is selected
`0` = 96 MHz PLL circuit is selected

bit 1 **ADCSEL:** A/D Converter Mode bit

`1` = 10-Bit Conversion mode is enabled
`0` = 12-Bit Conversion mode is enabled

bit 0 **IOL1WAY:** IOLOCK One-Way Set Enable bit

`1` = IOLOCK bit (PPSCON<0>) can be set once, provided the unlock sequence has been completed; once set, the Peripheral Pin Select registers cannot be written to a second time
`0` = IOLOCK bit (PPSCON<0>) can be set and cleared as needed, provided the unlock sequence has been completed

REGISTER 27-7: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1
WPCFG	WPFP6	WPFP5	WPFP4	WPFP3	WPFP2	WPFP1	WPFP0
bit 7							bit 0

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

bit 7 **WPCFG:** Write/Erase Protect Configuration Region Select bit (valid when WPDIS = `0`)

`1` = Configuration Words page is not erase/write-protected unless WPEND and WPFP<5:0> settings include the Configuration Words page(1)
`0` = Configuration Words page is erase/write-protected, regardless of WPEND and WPFP<5:0>(1)

bit 6-0 **WPFP<6:0>:** Write/Erase Protect Page Start/End Location bits

Used with the WPEND bit to define which pages in Flash will be erase/write-protected.

Note 1: The “Configuration Words page” contains the FCWs and is the last page of implemented Flash memory on a given device. Each page consists of 1,024 bytes. For example, on a device with 64 Kbytes of Flash, the first page is 0 and the last page (Configuration Words page) is 63 (3Fh).

REGISTER 27-8: CONFIG4H: CONFIGURATION REGISTER 4 HIGH (BYTE ADDRESS 300007h)

U-1	U-1	U-1	U-1	U-0	U-0	R/WO-1	R/WO-1
—	—	—	—	—	—	WPEND	WPDIS
bit 7							bit 0

Legend:

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

bit 7-4	**Unimplemented:** Program the corresponding Flash Configuration bit to ‘1’
bit 3-2	**Unimplemented:** Read as ‘0’
bit 1	**WPEND:** Write-Protect Disable bit 1 = Flash pages, WPFP<6:0> to (Configuration Words page), are erase/write-protected 0 = Flash pages 0 to WPFP<6:0> are erase/write-protected
bit 0	**WPDIS:** Write-Protect Disable bit 1 = WPFP<5:0>, WPEND and WPCFG bits are ignored; all Flash memory may be erased or written 0 = WPFP<5:0>, WPEND and WPCFG bits are enabled; erase/write-protect is active for the selected region(s)

REGISTER 27-9: DEVID1: DEVICE ID REGISTER 1 FOR PIC18F47J13 FAMILY DEVICES (BYTE ADDRESS 3FFFFEh)

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 These bits are used with DEV<10:3> bits in Device ID Register 2 to identify the part number. See Register 27-10.
bit 4-0	**REV<4:0>:** Revision ID bits These bits are used to indicate the device revision.

REGISTER 27-10: DEVID2: DEVICE ID REGISTER 2 FOR PIC18F47J13 FAMILY DEVICES (BYTE ADDRESS 3FFFFFh)

R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
bit 7							bit 0

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

bit 7-0 **DEV<10:3>:** Device ID bits

These bits are used with the DEV<2:0> bits in the Device ID Register 1 to identify the part number.

DEV<10:3> (DEVID2<7:0>)	DEV<2:0> (DEVID1<7:5>)	Device
`0101 1001`	`111`	PIC18F47J13
`0101 1001`	`101`	PIC18F46J13
`0101 1001`	`011`	PIC18F27J13
`0101 1001`	`001`	PIC18F26J13
`0101 1011`	`111`	PIC18LF47J13
`0101 1011`	`101`	PIC18LF46J13
`0101 1011`	`011`	PIC18LF27J13
`0101 1011`	`001`	PIC18LF26J13

27.2 Watchdog Timer (WDT)

PIC18F47J13 Family devices have both a conventional WDT circuit and a dedicated, Deep Sleep capable Watchdog Timer. When enabled, the conventional WDT operates in normal Run, Idle and Sleep modes. This data sheet section describes the conventional WDT circuit.

The dedicated, Deep Sleep capable WDT can only be enabled in Deep Sleep mode. This timer is described in **Section 4.6.4 "Deep Sleep Watchdog Timer (DSWDT)"**.

The conventional WDT is driven by the INTRC oscillator. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by the WDTPS bits in Configuration Register 2H. Available periods range from about 4 ms to 135 seconds (2.25 minutes depending on voltage, temperature and WDT postscaler). The WDT and postscaler are cleared whenever a `SLEEP` or `CLRWDT` instruction is executed, or a clock failure (primary or Timer1 oscillator) has occurred.

Note 1: The `CLRWDT` and `SLEEP` instructions clear the WDT and postscaler counts when executed.

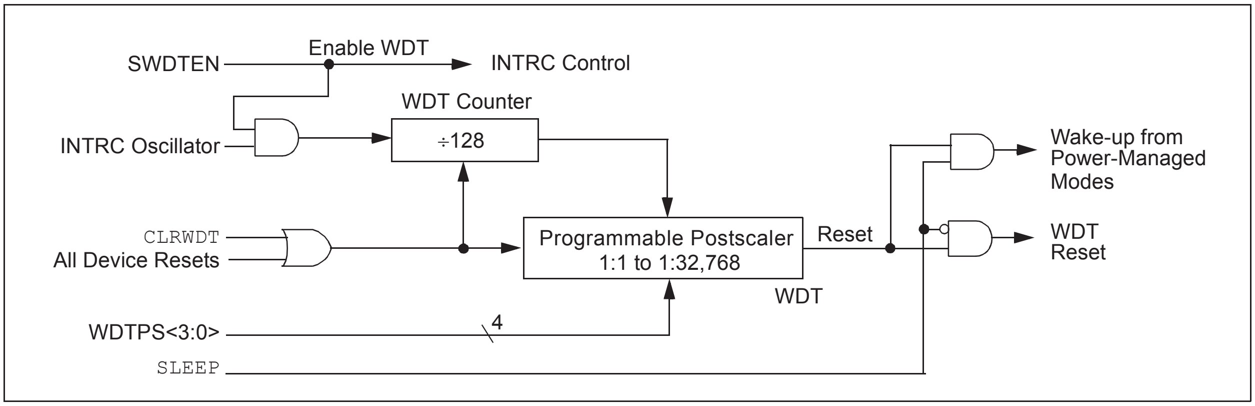
2: When a `CLRWDT` instruction is executed, the postscaler count will be cleared.

27.2.1 CONTROL REGISTER

The WDTCON register (Register 27-11) is a readable and writable register. The SWDTEN bit enables or disables WDT operation. This allows software to override the WDTEN Configuration bit and enable the WDT only if it has been disabled by the Configuration bit.

LVDSTAT is a read-only status bit that is continuously updated and provides information about the current level of VDDCORE. This bit is only valid when the on-chip voltage regulator is enabled.

FIGURE 27-1: WDT BLOCK DIAGRAM



REGISTER 27-11: WDTCON: WATCHDOG TIMER CONTROL REGISTER (ACCESS FC0h)

R/W-1	R-x	R-x	R/W-0	R-0	R/W-0	R/W-0	R/W-0
REGSLP	LVDSTAT[2]	ULPLVL	VBGOE	DS[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:** Voltage Regulator Low-Power Operation Enable bit

`1` = On-chip regulator enters low-power operation when device enters Sleep mode
`0` = On-chip regulator is active even in Sleep mode

bit 6 **LVDSTAT:** Low-Voltage Detect Status bit[2]

`1` = VDDCORE > 2.45V nominal
`0` = VDDCORE < 2.45V nominal

bit 5 **ULPLVL:** Ultra Low-Power Wake-up Output bit (not valid unless ULPEN = `1`)

`1` = Voltage on RA0 > ~0.5V
`0` = Voltage on RA0 < ~0.5V

bit 4 **VBGOE**: Band Gap Reference Voltage (VBG) Output Enable bit

`1` = Band gap reference output is enabled on the RA1 pin
`0` = Band gap reference output is disabled

bit 3 **DS:** Deep Sleep Wake-up Status bit (used in conjunction with RCON, POR and BOR bits to determine Reset source)[2]

`1` = If the last exit from Reset was caused by a normal wake-up from Deep Sleep
`0` = If the last exit from Reset was not due to a wake-up from Deep Sleep

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

`1` = Ultra low-power wake-up current sink is enabled (if ULPEN = `1`)
`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 bit, WDTEN, is enabled.
2: Not available on devices where the on-chip voltage regulator is disabled (“LF” devices).

TABLE 27-3: SUMMARY OF WATCHDOG TIMER REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RCON	IPEN	—	$\overline{\text{CM}}$	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
WDTCON	REGSLP	LVDSTAT	ULPLVL	VBGOE	DS	ULPEN	ULPSINK	SWDTEN

Legend: — = unimplemented, read as ‘`0`’. Shaded cells are not used by the Watchdog Timer.

27.3 On-Chip Voltage Regulator

Note 1: The on-chip voltage regulator is only available in parts designated with an “F”, such as PIC18**F**26J13. The on-chip regulator is disabled on devices with “LF” in their part number.

2: The VDDCORE/VCAP pin must never be left floating. On “F” devices, it must be connected to a capacitor of size, CEFC, to ground. On “LF” devices, VDDCORE/VCAP must be connected to a power supply source between 2.0V and 2.7V.

The digital core logic of the PIC18F47J13 Family devices is designed on an advanced manufacturing process, which requires 2.0V to 2.7V. The digital core logic obtains power from the VDDCORE/VCAP power supply pin.

However, in many applications it may be inconvenient to run the I/O pins at the same core logic voltage, as it would restrict the ability of the device to interface with other higher voltage devices, such as those run at a nominal 3.3V. Therefore, all PIC18F47J13 Family devices implement a dual power supply rail topology. The core logic obtains power from the VDDCORE/VCAP pin, while the general purpose I/O pins obtain power from the VDD pin of the microcontroller, which may be supplied with a voltage between 2.15V to 3.6V (“F” device) or 2.0V to 3.6V (“LF” device).

This dual supply topology allows the microcontroller to interface with standard 3.3V logic devices, while running the core logic at a lower voltage of nominally 2.5V.

In order to make the microcontroller more convenient to use, an integrated 2.5V low dropout, low quiescent current linear regulator has been integrated on the die inside PIC18F47J13 Family devices. This regulator is designed specifically to supply the core logic of the device. It allows PIC18F47J13 Family devices to effectively run from a single power supply rail, without the need for external regulators.

The on-chip voltage regulator is always enabled on “F” devices. The VDDCORE/VCAP pin simultaneously serves as the regulator output pin and the core logic supply power input pin. A capacitor should be connected to the VDDCORE/VCAP pin to ground and is necessary for regulator stability. For example connections for PIC18F and PIC18LF devices, see Figure 27-2.

On “LF” devices, the on-chip regulator is always disabled. This allows the device to save a small amount of quiescent current consumption, which may be advantageous in some types of applications, such as those which will entirely be running at a nominal 2.5V. On “LF” devices, the VDDCORE/VCAP pin still serves as the core logic power supply input pin, and therefore, must be connected to a 2.0V to 2.7V supply rail at the application circuit board level. On these devices, the I/O pins may still optionally be supplied with a voltage between 2.0V to 3.6V, provided that VDD is always greater than, or equal to, VDDCORE/VCAP. For example connections for PIC18F and PIC18LF devices, see Figure 27-2.

Note: In parts designated with an “LF”, such as PIC18LF47J13, VDDCORE must never exceed VDD.

The specifications for core voltage and capacitance are listed in **Section 30.3 “DC Characteristics: PIC18F47J13 Family (Industrial)”**.

27.3.1 VOLTAGE REGULATOR TRACKING MODE AND LOW-VOLTAGE DETECTION

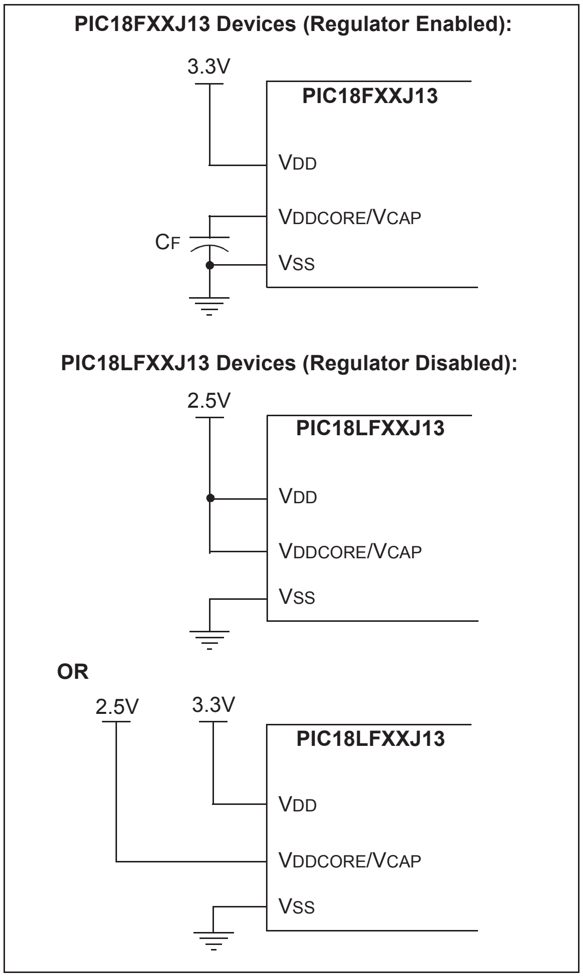
On “F” devices, the on-chip regulator provides a constant voltage of 2.5V nominal to the digital core logic. The regulator can provide this level from a VDD of about 2.5V, all the way up to the device’s VDDMAX. It does not have the capability to boost VDD levels below 2.5V. When the VDD supply input voltage drops too low to regulate 2.5V, the regulator enters Tracking mode. In Tracking mode, the regulator output follows VDD, with a typical voltage drop of 100 mV or less.

The on-chip regulator includes a simple Low-Voltage Detect (LVD) circuit. This circuit is separate and independent of the High/Low-Voltage Detect (HLVD) module described in **Section 25.0 “High/Low Voltage Detect (HLVD)”**. The on-chip regulator LVD circuit continuously monitors the VDDCORE voltage level and updates the LVDSTAT bit in the WDTCON register. The LVD detect threshold is set slightly below the normal regulation set point of the on-chip regulator.

Application firmware may optionally poll the LVDSTAT bit to determine when it is safe to run at the maximum rated frequency, so as not to inadvertently violate the voltage versus frequency requirements provided by Figure 30-1.

The VDDCORE monitoring LVD circuit is only active when the on-chip regulator is enabled. On “LF” devices, the Analog-to-Digital Converter and the HLVD module can still be used to provide firmware with VDD and VDDCORE voltage level information.

FIGURE 27-2: CONNECTIONS FOR THE ON-CHIP REGULATOR



27.3.2 ON-CHIP REGULATOR AND BOR

When the on-chip regulator is enabled, PIC18F47J13 Family devices also have a simple brown-out capability. If the voltage supplied to the regulator is inadequate to maintain a minimum output level, the regulator Reset circuitry will generate a Brown-out Reset (BOR). This event is captured by the $\overline{\text{BOR}}$ flag bit (RCON<0>).

The operation of the BOR is described in more detail in **Section 5.4 "Brown-out Reset (BOR)"** and **Section 5.4.1 "Detecting BOR"**. The brown-out voltage levels are specific in **Section 30.1 "DC Characteristics: Supply Voltage PIC18F47J13 Family (Industrial)"**.

27.3.3 POWER-UP REQUIREMENTS

The on-chip regulator is designed to meet the power-up requirements for the device. If the application does not use the regulator, then strict power-up conditions must be adhered to. While powering up, VDDCORE should not exceed VDD by 0.3 volts.

27.3.4 OPERATION IN SLEEP MODE

When enabled, the on-chip regulator always consumes a small incremental amount of current over IDD. This includes when the device is in Sleep mode, even though the core digital logic does not require much power. To provide additional savings in applications where power resources are critical, the regulator can be configured to automatically enter a lower quiescent draw Standby mode whenever the device goes into Sleep mode. This feature is controlled by the REGSLP bit (WDTCON<7>, Register 27-11). If this bit is set upon entry into Sleep mode, the regulator will transition into a lower power state. In this state, the regulator still provides a regulated output voltage necessary to maintain SRAM state information, but consumes less quiescent current.

Substantial Sleep mode power savings can be obtained by setting the REGSLP bit, but device wake-up time will increase in order to insure the regulator has enough time to stabilize.

27.4 Two-Speed Start-up

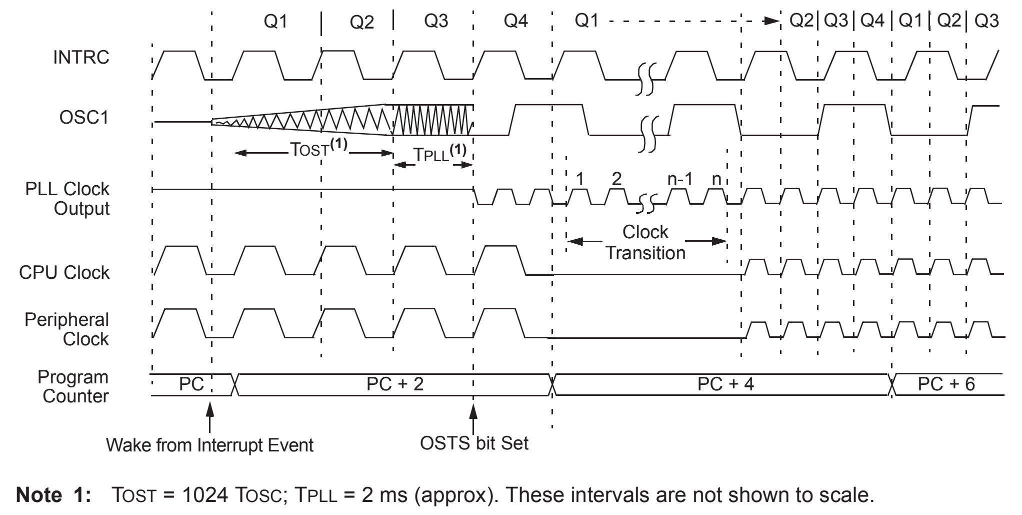
The Two-Speed Start-up feature helps to minimize the latency period, from oscillator start-up to code execution, by allowing the microcontroller to use the INTRC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO Configuration bit.

Two-Speed Start-up should be enabled only if the primary oscillator mode is HS or HSPLL (Crystal-Based) modes. Since the EC and ECPLL modes do not require an Oscillator Start-up Timer (OST) delay, Two-Speed Start-up should be disabled.

When enabled, Resets and wake-ups from Sleep mode cause the device to configure itself to run from the internal oscillator block as the clock source, following the time-out of the Power-up Timer after a Power-on Reset is enabled. This allows almost immediate code execution while the primary oscillator starts and the OST is running. Once the OST times out, the device automatically switches to PRI_RUN mode.

In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

FIGURE 27-3: TIMING TRANSITION FOR TWO-SPEED START-UP (INTRC TO HSPLL)



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

27.4.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTRC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power-managed modes, including serial `SLEEP` instructions (refer to **Section 4.1.4 "Multiple Sleep Commands"**). In practice, this means that user code can change the SCS<1:0> bit settings or issue `SLEEP` instructions before the OST times out. This would allow an application to briefly wake-up, perform routine "housekeeping" tasks and return to Sleep before the device starts to operate from the primary oscillator.

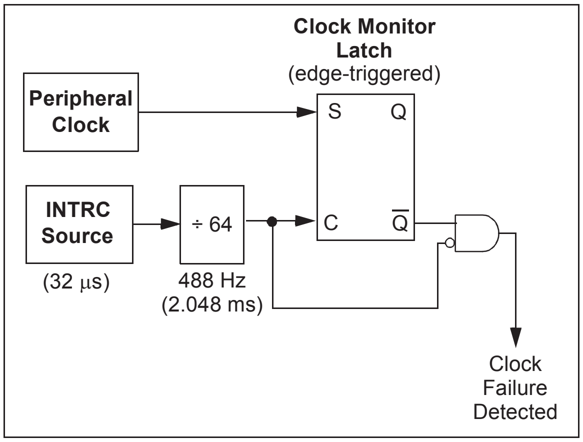
User code can also check if the primary clock source is currently providing the device clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

27.5 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the microcontroller to continue operation in the event of an external oscillator failure by automatically switching the device clock to the internal oscillator block. The FSCM function is enabled by setting the FCMEN Configuration bit.

When FSCM is enabled, the INTRC oscillator runs at all times to monitor clocks to peripherals and provide a backup clock in the event of a clock failure. Clock monitoring (shown in Figure 27-4) is accomplished by creating a sample clock signal, which is the INTRC output divided by 64. This allows ample time between FSCM sample clocks for a peripheral clock edge to occur. The peripheral device clock and the sample clock are presented as inputs to the clock monitor latch. The clock monitor is set on the falling edge of the device clock source but cleared on the rising edge of the sample clock.

FIGURE 27-4: FSCM BLOCK DIAGRAM


Clock Monitor
Latch
(edge-triggered)
Peripheral
Clock
S
Q
INTRC
Source
(32 µs)
÷ 64
488 Hz
(2.048 ms)
C
Q̄
Clock
Failure
Detected

Clock failure is tested for on the falling edge of the sample clock. If a sample clock falling edge occurs while the clock monitor is still set, and a clock failure has been detected (Figure 27-5), the following results:

- The FSCM generates an oscillator fail interrupt by setting bit, OSCFIF (PIR2<7>);
- The device clock source is switched to the internal oscillator block (OSCCON is not updated to show the current clock source – this is the fail-safe condition); and
- The WDT is reset.

During switchover, the postscaler frequency from the internal oscillator block may not be sufficiently stable for timing-sensitive applications. In these cases, it may be desirable to select another clock configuration and enter an alternate power-managed mode. This can be done to attempt a partial recovery or execute a controlled shutdown. See **Section 4.1.4 "Multiple Sleep Commands"** and **Section 27.4.1 "Special Considerations For Using Two-speed Start-up"** for more details.

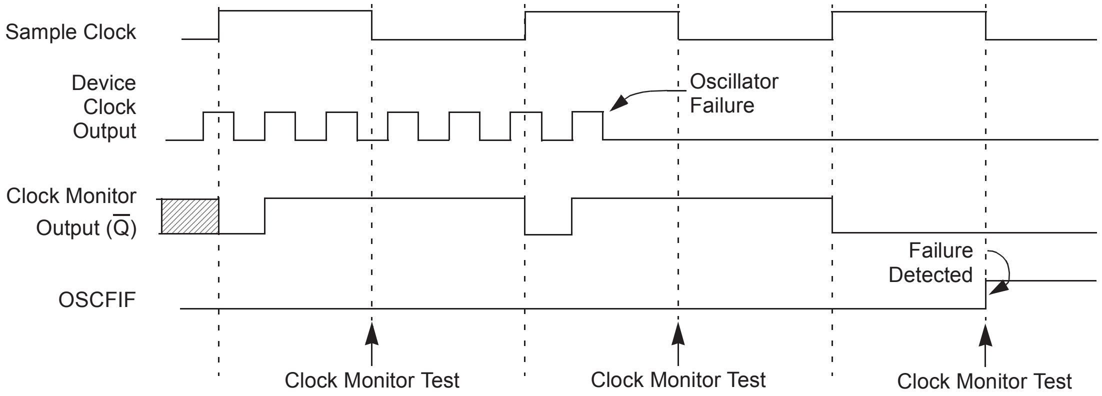
The FSCM will detect failures of the primary or secondary clock sources only. If the internal oscillator block fails, no failure would be detected, nor would any action be possible.

27.5.1 FSCM AND THE WATCHDOG TIMER

Both the FSCM and the WDT are clocked by the INTRC oscillator. Since the WDT operates with a separate divider and counter, disabling the WDT has no effect on the operation of the INTRC oscillator when the FSCM is enabled.

As already noted, the clock source is switched to the INTRC clock when a clock failure is detected; this may mean a substantial change in the speed of code execution. If the WDT is enabled with a small prescale value, a decrease in clock speed allows a WDT time-out to occur and a subsequent device Reset. For this reason, Fail-Safe Clock Monitor events also reset the WDT and postscaler, allowing it to start timing from when execution speed was changed and decreasing the likelihood of an erroneous time-out.

FIGURE 27-5: FSCM TIMING DIAGRAM


Sample Clock
Device
Clock
Output
Oscillator
Failure
Clock Monitor
Output (Q̄)
Failure
Detected
OSCFIF
Clock Monitor Test
Clock Monitor Test
Clock Monitor Test

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.

27.5.2 EXITING FAIL-SAFE OPERATION

The Fail-Safe Clock Monitor condition is terminated by either a device Reset or by entering a power-managed mode. On Reset, the controller starts the primary clock source specified in Configuration Register 2H (with any required start-up delays that are required for the oscillator mode, such as the OST or PLL timer). The INTRC oscillator provides the device clock until the primary clock source becomes ready (similar to a Two-Speed Start-up). The clock source is then switched to the primary clock (indicated by the OSTS bit in the OSCCON register becoming set). The FSCM then resumes monitoring the peripheral clock.

The primary clock source may never become ready during start-up. In this case, operation is clocked by the INTRC oscillator. The OSCCON register will remain in its Reset state until a power-managed mode is entered.

27.5.3 FSCM INTERRUPTS IN POWER-MANAGED MODES

By entering a power-managed mode, the clock multiplexer selects the clock source selected by the OSCCON register. FSCM of the power-managed clock source resumes in the power-managed mode.

If an oscillator failure occurs during power-managed operation, the subsequent events depend on whether or not the oscillator failure interrupt is enabled. If enabled (OSCFIF = `1`), code execution will be clocked by the INTRC multiplexer. An automatic transition back to the failed clock source will not occur.

If the interrupt is disabled, subsequent interrupts while in Idle mode will cause the CPU to begin executing instructions while being clocked by the INTRC source.

27.5.4 POR OR WAKE-UP FROM SLEEP

The FSCM is designed to detect oscillator failure at any point after the device has exited Power-on Reset (POR) or low-power Sleep mode. When the primary device clock is either the EC or INTRC modes, monitoring can begin immediately following these events.

For HS or HSPLL modes, the situation is somewhat different. Since the oscillator may require a start-up time considerably longer than the FSCM sample clock time, a false clock failure may be detected. To prevent this, the internal oscillator block is automatically configured as the device clock and functions until the primary clock is stable (the OST and PLL timers have timed out). This is identical to Two-Speed Start-up mode. Once the primary clock is stable, the INTRC returns to its role as the FSCM source.

Note: The same logic that prevents false oscillator failure interrupts on POR, or wake-up from Sleep, will also prevent the detection of the oscillator's failure to start at all following these events. This can be avoided by monitoring the OSTS bit and using a timing routine to determine if the oscillator is taking too long to start. Even so, no oscillator failure interrupt will be flagged.

As noted in **Section 27.4.1 "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.

27.6 Program Verification and Code Protection

For all devices in the PIC18F47J13 Family of devices, the on-chip program memory space is treated as a single block. Code protection for this block is controlled by one Configuration bit, CP0. This bit inhibits external reads and writes to the program memory space. It has no direct effect in normal execution mode.

27.6.1 CONFIGURATION REGISTER PROTECTION

The Configuration registers are protected against untoward changes or reads in two ways. The primary protection is the write-once feature of the Configuration bits, which prevents reconfiguration once the bit has been programmed during a power cycle. To safeguard against unpredictable events, Configuration bit changes resulting from individual cell level disruptions (such as ESD events) will cause a parity error and trigger a device Reset. This is seen by the user as a Configuration Mismatch (CM) Reset.

The data for the Configuration registers is derived from the FCW in program memory. When the CP0 bit is set, the source data for device configuration is also protected as a consequence.

27.7 In-Circuit Serial Programming™ (ICSP™)

PIC18F47J13 Family microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data, and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

27.8 In-Circuit Debugger

When the $\overline{\text{DEBUG}}$ Configuration bit is programmed to a ‘0’, the In-Circuit Debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some resources are not available for general use.

Table 27-4 lists the resources required by the background debugger.

TABLE 27-4: DEBUGGER RESOURCES

I/O Pins:	RB6, RB7
Stack:	TOSx registers reserved.

28.0 INSTRUCTION SET SUMMARY

The PIC18F47J13 Family of devices incorporate the standard set of 75 PIC18 core instructions, as well as an extended set of 8 new instructions for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

28.1 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 28-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 28-1 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 28-1 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 28-2, lists the standard instructions recognized by the Microchip MPASM™ Assembler.

Section 28.1.1 “Standard Instruction Set” provides a description of each instruction.

TABLE 28-1: OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM Access bit: a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7).
BSR	Bank Select Register. Used to select the current RAM bank.
C, DC, Z, OV, N	ALU Status bits: **C**arry, **D**igit **C**arry, **Z**ero, **Ov**erflow, **N**egative.
d	Destination Select bit: d = 0: store result in WREG d = 1: store result in file register f
dest	Destination: either the WREG register or the specified register file location.
f	8-bit Register file address (00h to FFh), or 2-bit FSR designator (0h to 3h).
f_s	12-bit Register file address (000h to FFFh). This is the source address.
f_d	12-bit Register file address (000h to FFFh). This is the destination address.
GIE	Global Interrupt Enable bit.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
label	Label name.
mm	The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions:
*	No change to register (such as TBLPTR with table reads and writes)
*+	Post-Increment register (such as TBLPTR with table reads and writes)
*-	Post-Decrement register (such as TBLPTR with table reads and writes)
+*	Pre-Increment register (such as TBLPTR with table reads and writes)
n	The relative address (2's complement number) for relative branch instructions or the direct address for Call/Branch and Return instructions.
PC	Program Counter.
PCL	Program Counter Low Byte.
PCH	Program Counter High Byte.
PCLATH	Program Counter High Byte Latch.
PCLATU	Program Counter Upper Byte Latch.
$\overline{\text{PD}}$	Power-Down bit.
PRODH	Product of Multiply High Byte.
PRODL	Product of Multiply Low Byte.
s	Fast Call/Return Mode Select bit: s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
TBLPTR	21-bit Table Pointer (points to a program memory location).
TABLAT	8-bit Table Latch.
$\overline{\text{TO}}$	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).

FIGURE 28-1: GENERAL FORMAT FOR INSTRUCTIONS

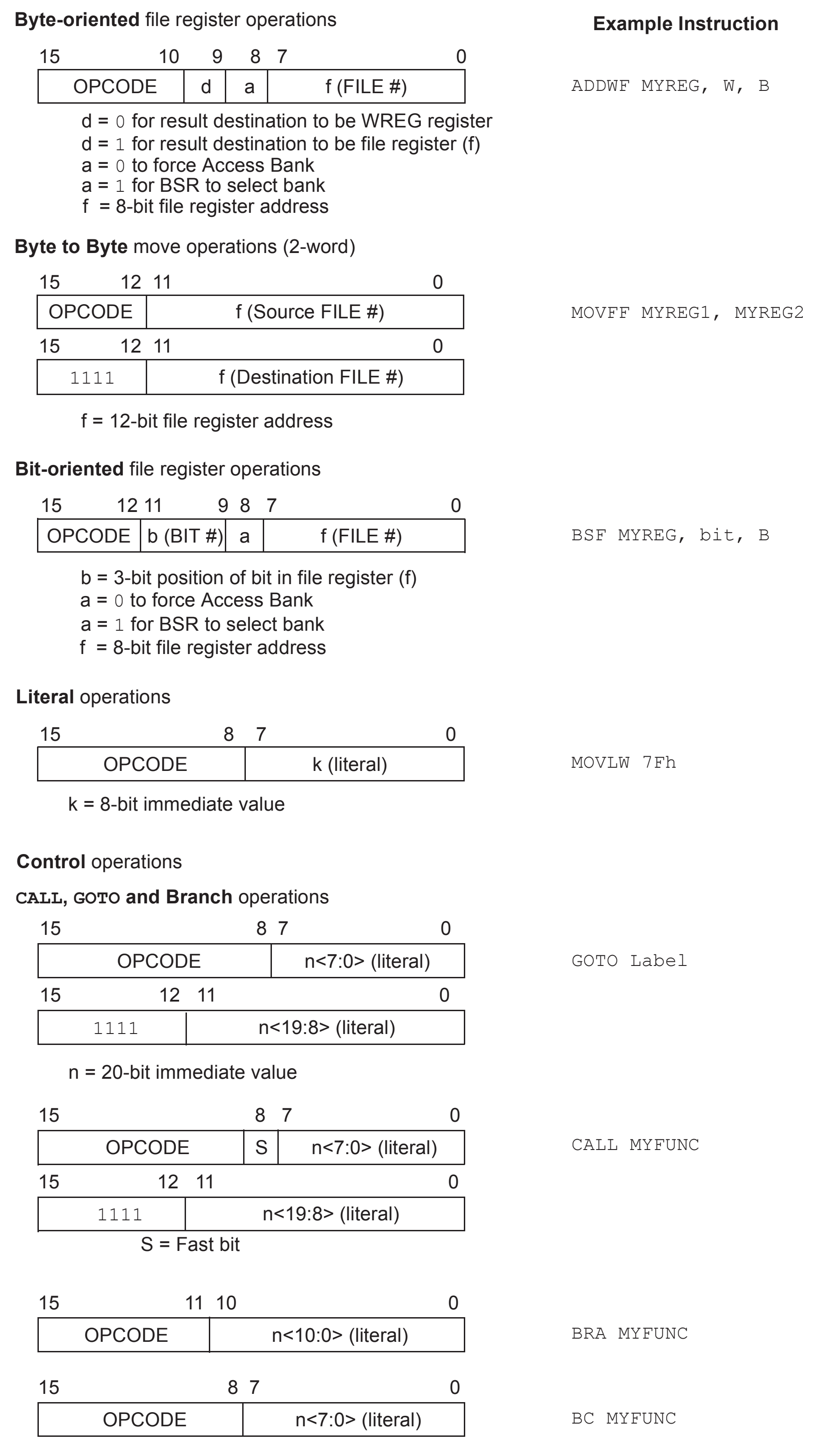

Byte-oriented file register operations
Example Instruction
15 10 9 8 7 0
OPCODE
d
a
f (FILE #)
ADDWF MYREG, W, B
d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address
Byte to Byte move operations (2-word)
15 12 11 0
OPCODE
f (Source FILE #)
MOVFF MYREG1, MYREG2
15 12 11 0
1111
f (Destination FILE #)
f = 12-bit file register address
Bit-oriented file register operations
15 12 11 9 8 7 0
OPCODE
b (BIT #)
a
f (FILE #)
BSF MYREG, bit, B
b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address
Literal operations
15 8 7 0
OPCODE
k (literal)
MOVLW 7Fh
k = 8-bit immediate value
Control operations
CALL, GOTO and Branch operations
15 8 7 0
OPCODE
n<7:0> (literal)
GOTO Label
15 12 11 0
1111
n<19:8> (literal)
n = 20-bit immediate value
15 8 7 0
OPCODE
S
n<7:0> (literal)
CALL MYFUNC
15 12 11 0
1111
n<19:8> (literal)
S = Fast bit
15 11 10 0
OPCODE
n<10:0> (literal)
BRA MYFUNC
15 8 7 0
OPCODE
n<7:0> (literal)
BC MYFUNC

TABLE 28-2: PIC18F47J13 FAMILY INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BYTE-ORIENTED OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, Skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, Skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, Skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word	2	1100	ffff	ffff	ffff	None	
		f_d (destination) 2nd word		1111	ffff	ffff	ffff		
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with Borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with Borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	Swap Nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, Skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '`1`' for a pin configured as an 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 28-2: PIC18F47J13 FAMILY INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
BIT-ORIENTED OPERATIONS									
BCF	f, b, a	Bit Clear f	1	`1001`	`bbba`	`ffff`	`ffff`	None	1, 2
BSF	f, b, a	Bit Set f	1	`1000`	`bbba`	`ffff`	`ffff`	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	`1011`	`bbba`	`ffff`	`ffff`	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	`1010`	`bbba`	`ffff`	`ffff`	None	3, 4
BTG	f, b, a	Bit Toggle f	1	`0111`	`bbba`	`ffff`	`ffff`	None	1, 2
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	`1110`	`0010`	`nnnn`	`nnnn`	None	
BN	n	Branch if Negative	1 (2)	`1110`	`0110`	`nnnn`	`nnnn`	None	
BNC	n	Branch if Not Carry	1 (2)	`1110`	`0011`	`nnnn`	`nnnn`	None	
BNN	n	Branch if Not Negative	1 (2)	`1110`	`0111`	`nnnn`	`nnnn`	None	
BNOV	n	Branch if Not Overflow	1 (2)	`1110`	`0101`	`nnnn`	`nnnn`	None	
BNZ	n	Branch if Not Zero	1 (2)	`1110`	`0001`	`nnnn`	`nnnn`	None	
BOV	n	Branch if Overflow	1 (2)	`1110`	`0100`	`nnnn`	`nnnn`	None	
BRA	n	Branch Unconditionally	2	`1101`	`0nnn`	`nnnn`	`nnnn`	None	
BZ	n	Branch if Zero	1 (2)	`1110`	`0000`	`nnnn`	`nnnn`	None	
CALL	n, s	Call Subroutine 1st word	2	`1110`	`110s`	`kkkk`	`kkkk`	None	
		2nd word		`1111`	`kkkk`	`kkkk`	`kkkk`		
CLRWDT	—	Clear Watchdog Timer	1	`0000`	`0000`	`0000`	`0100`	$\overline{\text{TO}}$, $\overline{\text{PD}}$	
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`	$\overline{\text{TO}}$, $\overline{\text{PD}}$	

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 an 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 28-2: PIC18F47J13 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 an 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.

28.1.1 STANDARD INSTRUCTION SET

ADDLW	**ADD Literal to W**
Syntax:	ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → 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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(W) + (f) → 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’ (default). If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default). If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “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 (default). 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 28.2.3 "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

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’ (default). If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default). 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 28.2.3 “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 2-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 (default). 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 28.2.3 “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 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal ‘n’	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal ‘n’	Process Data	No operation

Example: `HERE BN Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
If Negative = `1;`
PC = address `(Jump)`
If Negative = `0;`
PC = address `(HERE + 2)`

BNC **Branch if Not Carry**

Syntax:	BNC n
Operands:	-128 ≤ n ≤ 127
Operation:	if Carry bit is ‘`0`’, (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 2-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 2-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 2-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 2-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNZ Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
If Zero = `0;`
PC = address `(Jump)`
If Zero = `1;`
PC = address `(HERE + 2)`

BRA Unconditional Branch

Syntax: BRA n

Operands: -1024 ≤ 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 2-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 (default).

If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 "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 2-cycle instruction.

If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default).

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 28.2.3 “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 2-cycle instruction.

If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default).

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 28.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1(2)

Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE     BTFSS    FLAG, 1, 0
FALSE    :
TRUE     :
```

Before Instruction
PC = address `(HERE)`
After Instruction
If FLAG<1> = `0;`
PC = address `(FALSE)`
If FLAG<1> = `1;`
PC = address `(TRUE)`

BTG Bit Toggle f

Syntax:	BTG f, b {,a}
Operands:	$0 \leq f \leq 255$ $0 \leq b < 7$ $a \in [0,1]$
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 (default). If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “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:	$-128 \leq n \leq 127$
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 2-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 2-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 (default). Then, the 20-bit value ‘k’ is loaded into PC<20:1>. `CALL` is a 2-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’<7:0>,	Push PC to stack	Read literal ’k’<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `HERE CALL THERE,1`

Before Instruction
PC = address (HERE)
After Instruction
PC = address (THERE)
TOS = address (HERE + 4)
WS = W
BSRS = BSR
STATUSS = STATUS

CLRF	**Clear f**
Syntax:	CLRF f {,a}
Operands:	0 ≤ f ≤ 255 a ∈ [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 (default). If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “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` → $\overline{\text{TO}}$, `1` → $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Encoding:	`0000` | `0000` | `0000` | `0100`
Description:	`CLRWDT` instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{\text{TO}}$ and $\overline{\text{PD}}$, 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`
$\overline{\text{TO}}$ = `1`
$\overline{\text{PD}}$ = `1`

COMF Complement f

Syntax: COMF f {,d {,a}}

Operands: $0 \leq f \leq 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $\overline{f} \rightarrow$ 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' (default).

If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank (default).

If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 28.2.3 "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: $0 \leq f \leq 255$
$a \in [0,1]$

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 2-cycle instruction.

If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank (default).

If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 28.2.3 "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)`

CPFSGT Compare f with W, Skip if f > W

Syntax:	CPFSGT f {,a}
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$
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 2-cycle instruction. If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default). If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 28.2.3 “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:	$0 \leq f \leq 255$ $a \in [0,1]$
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 2-cycle instruction. If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default).
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: $0 \leq f \leq 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

0000	01da	ffff	ffff

Description: Decrement register ‘f’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default).

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default).

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 28.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

Example: `DECF CNT, 1, 0`

Before Instruction
CNT = 01h
Z = 0
After Instruction
CNT = 00h
Z = 1

DECFSZ	**Decrement f, Skip if 0**
Syntax:	DECFSZ f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [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' (default). If the result is '`0`', the next instruction which is already fetched is discarded and a `NOP` is executed instead, making it a 2-cycle instruction. If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank (default). If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 "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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [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' (default). If the result is not '`0`', the next instruction which is already fetched is discarded and a `NOP` is executed instead, making it a 2-cycle instruction. If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank (default). If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 "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)`

GOTO	**Unconditional Branch**
Syntax:	GOTO k
Operands:	0 ≤ k ≤ 1048575
Operation:	k → PC<20:1>
Status Affected:	None
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	`1110` `1111` k$_7$kkk kkkk$_0$ `1111` k$_{19}$kkk kkkk kkkk$_8$
Description:	`GOTO` allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. `GOTO` is always a 2-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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) + `1` → dest
Status Affected:	C, DC, N, OV, Z
Encoding:	`0010` `10da` `ffff` `ffff`
Description:	The contents of register 'f' are incremented. If 'd' is '`0`', the result is placed in W. If 'd' is '`1`', the result is placed back in register 'f' (default). If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank (default). If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `INCF CNT, 1, 0`

Before Instruction
CNT = FFh
Z = `0`
C = ?
DC = ?
After Instruction
CNT = 00h
Z = `1`
C = `1`
DC = `1`

INCFSZ — Increment f, Skip if 0

Syntax: INCFSZ f {,d {,a}}

Operands: 0 ≤ 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’. (default)

If the result is ‘`0`’, the next instruction which is already fetched is discarded and a `NOP` is executed instead, making it a 2-cycle instruction.

If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default).

If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “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’ (default).

If the result is not ‘`0`’, the next instruction which is already fetched is discarded and a `NOP` is executed instead, making it a 2-cycle instruction.

If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default).

If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “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 ≤ k ≤ 255
Operation:	(W) .OR. k → W
Status Affected:	N, Z
Encoding:	`0000` `1001` `kkkk` `kkkk`
Description:	The contents of W are ORed with the 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: `IORLW 35h`

Before Instruction
W = 9Ah
After Instruction
W = BFh

IORWF	**Inclusive OR W with f**
Syntax:	IORWF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(W) .OR. (f) → dest
Status Affected:	N, Z
Encoding:	`0001` `00da` `ffff` `ffff`
Description:	Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `IORWF RESULT, 0, 1`

Before Instruction
RESULT = 13h
W = 91h
After Instruction
RESULT = 13h
W = 93h

LFSR Load FSR

Syntax: LFSR f, k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: k → FSRf

Status Affected: None

Encoding:

`1110`	`1110`	`00ff`	k_{11}`kkk`
`1111`	`0000`	k_7`kkk`	`kkkk`

Description: The 12-bit literal ‘k’ is loaded into the file select register pointed to by ‘f’.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’ MSB	Process Data	Write literal ‘k’ MSB to FSRfH
Decode	Read literal ‘k’ LSB	Process Data	Write literal ‘k’ to FSRfL

Example: `LFSR 2, 3ABh`

After Instruction
FSR2H = 03h
FSR2L = ABh

MOVF Move f

Syntax: MOVF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: f → dest

Status Affected: N, Z

Encoding:

`0101`	`00da`	`ffff`	`ffff`

Description: The contents of register ‘f’ are moved to a destination dependent upon the status of ‘d’. If ‘d’ is ‘`0`’, the result is placed in W. If ‘d’ is ‘`1`’, the result is placed back in register ‘f’ (default). Location ‘f’ can be anywhere in the 256-byte bank.

If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default).

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 28.2.3 “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:	$0 \leq f_s \leq 4095$ $0 \leq f_d \leq 4095$
Operation:	$(f_s) \rightarrow f_d$
Status Affected:	None

Encoding:

1st word (source)	`1100`	`ffff`	`ffff`	`ffff`$_s$
2nd word (destin.)	`1111`	`ffff`	`ffff`	`ffff`$_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f_d' can also be anywhere from 000h to FFFh.

Either source or destination can be W (a useful special situation).

`MOVFF` is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).

The `MOVFF` instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register

Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f' (src)	Process Data	No operation
Decode	No operation 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:	MOVLW k
Operands:	$0 \leq k \leq 255$
Operation:	$k \rightarrow BSR$
Status Affected:	None

Encoding:

`0000`	`0001`	`kkkk`	`kkkk`

Description: The 8-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '`0`' regardless of the value of k_7:k_4.

Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR

Example: `MOVLB 5`

Before Instruction
BSR Register = 02h
After Instruction
BSR Register = 05h

MOVLW	**Move Literal to W**
Syntax:	MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → W
Status Affected:	None
Encoding:	`0000` `1110` `kkkk` `kkkk`
Description:	The 8-bit literal ‘k’ is loaded into W.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to W

Example: `MOVLW 5Ah`

After Instruction
W = 5Ah

MOVWF	**Move W to f**
Syntax:	MOVWF f {,a}
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(W) → f
Status Affected:	None
Encoding:	`0110` `111a` `ffff` `ffff`
Description:	Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank. If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default). If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write register ‘f’

Example: `MOVWF REG, 0`

Before Instruction
W = 4Fh
REG = FFh
After Instruction
W = 4Fh
REG = 4Fh

MULLW	Multiply Literal with W
Syntax:	MULLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) x k → PRODH:PRODL
Status Affected:	None
Encoding:	`0000` `1101` `kkkk` `kkkk`
Description:	An unsigned multiplication is carried out between the contents of W and the 8-bit literal ‘k’. The 16-bit result is placed in 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 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(W) x (f) → PRODH:PRODL
Status Affected:	None
Encoding:	`0000` `001a` `ffff` `ffff`
Description:	An unsigned multiplication is carried out between the contents of W and the register file location ‘f’. The 16-bit result is stored in 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 (default). If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write registers PRODH: PRODL

Example: `MULWF REG, 1`

Before Instruction
W = C4h
REG = B5h
PRODH = ?
PRODL = ?
After Instruction
W = C4h
REG = B5h
PRODH = 8Ah
PRODL = 94h

NEGF	**Negate f**
Syntax:	NEGF f {,a}
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$
Operation:	$(\overline{f}) + 1 \rightarrow f$
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 (default). If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 28.2.3 “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

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 2-cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' 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 (default).
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	POP PC from stack 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:	$0 \leq k \leq 255$
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

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 (default).
Words:	1
Cycles:	2

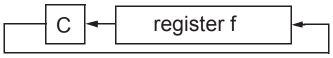
Q Cycle Activity:

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' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RLCF REG, 0, 0`

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 1100 1100
C = 1

RLNCF Rotate Left f (No Carry)

Syntax:	RLNCF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f<n>) → dest<n + 1>, (f<7>) → dest<0>
Status Affected:	N, Z

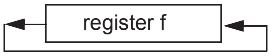
Encoding:

0100	01da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RLNCF REG, 1, 0`

Before Instruction
REG = 1010 1011
After Instruction
REG = 0101 0111

RRCF Rotate Right f through Carry

Syntax:	RRCF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f<n>) → dest<n – 1>, (f<0>) → C, (C) → dest<7>
Status Affected:	C, N, Z

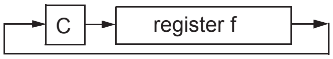
Encoding:

0011	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RRCF REG, 0, 0`

Before Instruction
REG = 1110 0110
C = 0
After Instruction
REG = 1110 0110
W = 0111 0011
C = 0

RRNCF — Rotate Right f (No Carry)

Syntax: RRNCF f {,d {,a}}

Operands: $0 \leq f \leq 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

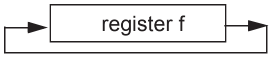
Encoding:

`0100`	`00da`	`ffff`	`ffff`

Description: The contents of register ‘f’ are rotated one bit to the right. If ‘d’ is ‘`0`’, the result is placed in W. If ‘d’ is ‘`1`’, the result is placed back in register ‘f’ (default).

If ‘a’ is ‘`0`’, the Access Bank will be selected, overriding the BSR value. If ‘a’ is ‘`1`’, then the bank will be selected as per the BSR value (default).

If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 28.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

Example 1: `RRNCF REG, 1, 0`

Before Instruction
REG = `1101 0111`
After Instruction
REG = `1110 1011`

Example 2: `RRNCF REG, 0, 0`

Before Instruction
W = ?
REG = `1101 0111`
After Instruction
W = `1110 1011`
REG = `1101 0111`

SETF — Set f

Syntax: SETF f {,a}

Operands: $0 \leq f \leq 255$
$a \in [0,1]$

Operation: FFh → f

Status Affected: None

Encoding:

`0110`	`100a`	`ffff`	`ffff`

Description: The contents of the specified register are set to FFh.

If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default).

If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 28.2.3 “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` → $\overline{\text{WDT}}$ postscaler, `1` → $\overline{\text{TO}}$, `0` → $\overline{\text{PD}}$
Status Affected:	$\overline{\text{TO}}$, $\overline{\text{PD}}$
Encoding:	`0000` `0000` `0000` `0011`
Description:	The Power-Down status bit ($\overline{\text{PD}}$) is cleared. The Time-out status bit ($\overline{\text{TO}}$) 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
$\overline{\text{TO}}$ = ?
$\overline{\text{PD}}$ = ?
After Instruction
$\overline{\text{TO}}$ = `1` †
$\overline{\text{PD}}$ = `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) – ($\overline{\text{C}}$) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	`0101` `01da` `ffff` `ffff`
Description:	Subtract register 'f' and Carry flag (borrow) from W (2's complement method). If 'd' is '`0`', the result is stored in W. If 'd' is '`1`', the result is stored in register 'f' (default). If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank (default). If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: `SUBFWB REG, 1, 0`

Before Instruction
REG = 3
W = 2
C = 1
After Instruction
REG = FF
W = 2
C = 0
Z = 0
N = 1 ; result is negative

Example 2: `SUBFWB REG, 0, 0`

Before Instruction
REG = 2
W = 5
C = 1
After Instruction
REG = 2
W = 3
C = 1
Z = 0
N = 0 ; result is positive

Example 3: `SUBFWB REG, 1, 0`

Before Instruction
REG = 1
W = 2
C = 0
After Instruction
REG = 0
W = 2
C = 1
Z = 1 ; result is zero
N = 0

SUBLW	**Subtract W from Literal**
Syntax:	SUBLW k
Operands:	0 ≤ k ≤ 255
Operation:	k – (W) → 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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) – (W) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	`0101` `11da` `ffff` `ffff`
Description:	Subtract W from register ‘f’ (2’s complement method). If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default). If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

Example 1: `SUBWF REG, 1, 0`

Before Instruction
REG = 3
W = 2
C = ?
After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBWF REG, 0, 0`

Before Instruction
REG = 2
W = 2
C = ?
After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBWF REG, 1, 0`

Before Instruction
REG = 1
W = 2
C = ?
After Instruction
REG = FFh ;(2’s complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

SUBWFB Subtract W from f with Borrow

Syntax:	SUBWFB f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [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’ (default).

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default).

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [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’ (default).

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank (default).

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

Example: `SWAPF REG, 1, 0`

Before Instruction
REG = 53h
After Instruction
REG = 35h

TBLRD Table Read

Syntax: TBLRD (*; *+; *-; +*)

Operands: None

Operation:
if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT;
TBLPTR – No Change
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) + 1 → TBLPTR
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) – 1 → TBLPTR
if TBLRD +*,
(TBLPTR) + 1 → TBLPTR;
(Prog Mem (TBLPTR)) → TABLAT

Status Affected: None

Encoding:

`0000`	`0000`	`0000`	`10nn`
			`nn=0 *`
			`=1 *+`
			`=2 *-`
			`=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 nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction uses the 3 LSBs of TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 6.0 "Memory Organization"** for additional details on programming Flash memory.)

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = `0`:	Least Significant Byte of Program Memory Word
TBLPTR[0] = `1`:	Most Significant Byte of Program Memory Word

The `TBLWT` instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

TBLWT — Table Write (Continued)

Example 1: `TBLWT *+;`

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
HOLDING REGISTER (00A356h) = FFh

After Instructions (table write completion)
TABLAT = 55h
TBLPTR = 00A357h
HOLDING REGISTER (00A356h) = 55h

Example 2: `TBLWT +*;`

Before Instruction
TABLAT = 34h
TBLPTR = 01389Ah
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = FFh

After Instruction (table write completion)
TABLAT = 34h
TBLPTR = 01389Bh
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = 34h

TSTFSZ	**Test f, Skip if 0**
Syntax:	TSTFSZ f {,a}
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	skip if f = `0`
Status Affected:	None
Encoding:	`0110` `011a` `ffff` `ffff`
Description:	If ‘f’ = `0`, the next instruction fetched during the current instruction execution is discarded and a `NOP` is executed, making this a 2-cycle instruction. If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank (default). If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 28.2.3 “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 ≤ k ≤ 255
Operation:	(W) .XOR. k → W
Status Affected:	N, Z
Encoding:	`0000` `1010` `kkkk` `kkkk`
Description:	The contents of W are XORed with the 8-bit literal ‘k’. The result is placed in W.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to W

Example:

```
XORLW    0xAF
```

Before Instruction
W = B5h
After Instruction
W = 1Ah

XORWF	**Exclusive OR W with f**
Syntax:	XORWF f {,d {,a}}
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) .XOR. (f) → dest
Status Affected:	N, Z
Encoding:	`0001` `10da` `ffff` `ffff`
Description:	Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 28.2.3 "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

28.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, the PIC18F47J13 Family of devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment Indirect and Indexed Addressing operations and the implementation of Indexed Literal Offset Addressing for many of the standard PIC18 instructions.

The additional features of the extended instruction set are enabled by default on unprogrammed devices. Users must properly set or clear the XINST Configuration bit during programming to enable or disable these features.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers or use them for Indexed Addressing. Two of the instructions, `ADDFSR` and `SUBFSR`, each have an additional special instantiation for using FSR2. These versions (`ADDULNK` and `SUBULNK`) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- Dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- Function Pointer invocation
- Software Stack Pointer manipulation
- Manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 28-3. Detailed descriptions are provided in **Section 28.2.2 "Extended Instruction Set"**. The opcode field descriptions in Table 28-1 (page 421) 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 the 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.

28.2.1 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 28.2.3.1 "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 28-3: 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

28.2.2 EXTENDED INSTRUCTION SET

ADDFSR	**Add Literal to FSR**
Syntax:	ADDFSR f, k
Operands:	$0 \leq k \leq 63$ $f \in [0, 1, 2]$
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, 0x23`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 0422h

ADDULNK	**Add Literal to FSR2 and Return**
Syntax:	ADDULNK k
Operands:	$0 \leq k \leq 63$
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 0x23`

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 0422h
PC = (TOS)

Note: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

CALLW — Subroutine Call Using WREG

Syntax:	CALLW
Operands:	None
Operation:	(PC + 2) → 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:	$((FSR2) + z_s) \rightarrow f_d$
Status Affected:	None
Encoding: 1st word (source) 2nd word (destin.)	`1110` `1011` `0zzz` $zzzz_s$ `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 No dummy read	No operation	Write register 'f' (dest)

Example: `MOVSF [0x05], REG2`

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
REG2 = 11h

After Instruction
FSR2 = 80h
Contents of 85h = 33h
REG2 = 33h

MOVSS Move Indexed to Indexed

Syntax:	MOVSS [z_s], [z_d]
Operands:	$0 \le z_s \le 127$ $0 \le z_d \le 127$
Operation:	$((FSR2) + z_s) \rightarrow ((FSR2) + z_d)$
Status Affected:	None

Encoding:				
1st word (source)	`1110`	`1011`	`1zzz`	`zzzz`$_s$
2nd word (dest.)	`1111`	`xxxx`	`xzzz`	`zzzz`$_d$

Description

The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets, 'z_s' or 'z_d', respectively, to the value of FSR2. Both registers can be located anywhere in the 4096-byte data memory space (000h to FFFh).

The `MOVSS` instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

If the resultant source address points to an Indirect Addressing register, the value returned will be 00h. If the resultant destination address points to an Indirect Addressing register, the instruction will execute as a `NOP`.

Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Determine source addr	Determine source addr	Read source reg
Decode	Determine dest addr	Determine dest addr	Write to dest reg

Example: `MOVSS [0x05], [0x06]`

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:	$k \rightarrow (FSR2)$, $FSR2 - 1 \rightarrow FSR2$
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 0x08`

Before Instruction
FSR2H:FSR2L = 01ECh
Memory (01ECh) = 00h

After Instruction
FSR2H:FSR2L = 01EBh
Memory (01ECh) = 08h

SUBFSR **Subtract Literal from FSR**

Syntax:	SUBFSR f, k
Operands:	0 ≤ k ≤ 63 f ∈ [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, 0x23`

Before Instruction
FSR2 = 03FFh
After Instruction
FSR2 = 03DCh

SUBULNK **Subtract Literal from FSR2 and Return**

Syntax:	SUBULNK k
Operands:	0 ≤ k ≤ 63
Operation:	FSR2 – k → FSR2, (TOS) → PC
Status Affected:	None
Encoding:	`1110` `1001` `11kk` `kkkk`
Description:	The 6-bit literal ‘k’ is subtracted from the contents of the FSR2. A `RETURN` is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a `NOP` is performed during the second cycle. This may be thought of as a special case of the `SUBFSR` instruction, where f = 3 (binary ‘`11`’); it operates only on FSR2.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination
No Operation	No Operation	No Operation	No Operation

Example: `SUBULNK 0x23`

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 03DCh
PC = (TOS)

28.2.3 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 28.2.3.1 "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.

28.2.3.1 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.

28.2.4 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 PIC18F47J13 Family, it is very important to consider the type of code. A large, re-entrant application that is written in C and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.

ADDWF

ADD W to Indexed (Indexed Literal Offset mode)

Syntax: ADDWF [k] {,d}

Operands: $0 \le k \le 95$
$d \in [0,1]$

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’ (default).

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

28.2.5 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 PIC18F47J13 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 '`1`', enabling 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.

29.0 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

29.1 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

29.2 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

29.3 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

29.4 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

29.5 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

29.6 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.

29.7 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.

29.8 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.

29.9 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™).

29.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.

29.11 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.

29.12 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®

30.0 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 or $\overline{\text{MCLR}}$ with respect to VSS (except VDD)	-0.3V to 6.0V
Voltage on any combined digital and analog pin with respect to VSS (except VDD)	-0.3V to (VDD + 0.3V)
Voltage on VDDCORE with respect to VSS	-0.3V to 2.75V
Voltage on VDD with respect to VSS	-0.3V to 4.0V
Total power dissipation **(Note 1)**	1.0W
Maximum current out of VSS pin	300 mA
Maximum current into VDD pin	250 mA
Maximum output current sunk by any PORTB, PORTC and RA6 I/O pin	25 mA
Maximum output current sunk by any PORTA (except RA6), PORTD and PORTE I/O pin	8 mA
Maximum output current sourced by any PORTB, PORTC and RA6 I/O pin	25 mA
Maximum output current sourced by any PORTA (except RA6), PORTD and PORTE I/O pin	8 mA
Maximum current sunk by all ports	200 mA
Maximum current sourced by all ports	200 mA

Note 1: Power dissipation is calculated as follows:

$$P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$$

† **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 30-1: PIC18F47J13 FAMILY VDD FREQUENCY GRAPH (INDUSTRIAL)

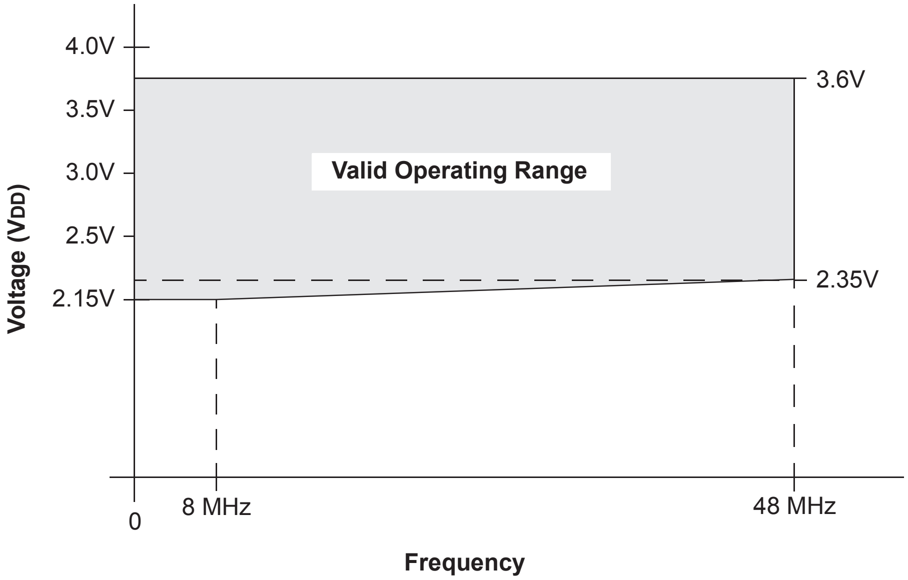
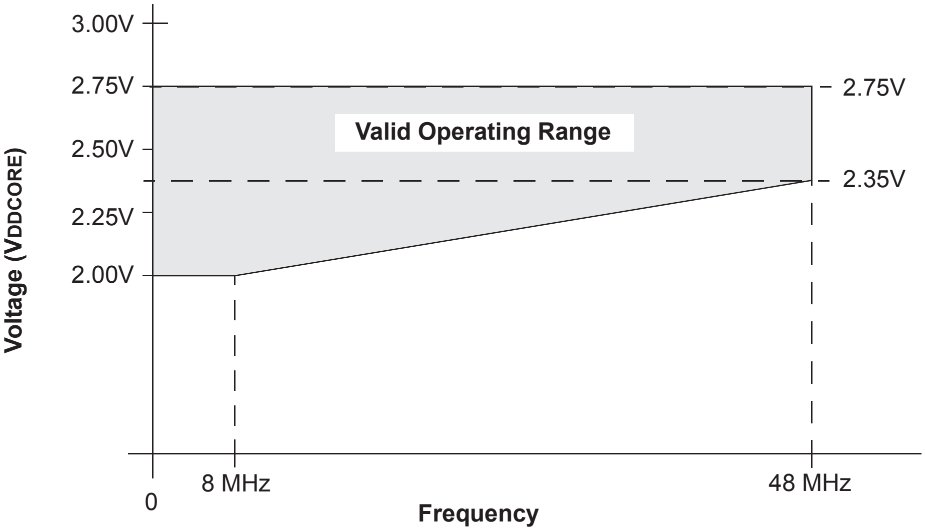


FIGURE 30-2: PIC18LF47J13 FAMILY VDDCORE FREQUENCY GRAPH (INDUSTRIAL)(1)



Note 1: VDD and VDDCORE must be maintained so that VDDCORE ≤ VDD.

30.1 DC Characteristics: Supply Voltage PIC18F47J13 Family (Industrial)

PIC18F47J13 Family			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
Param No.	**Symbol**	**Characteristic**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
D001	VDD	Supply Voltage	2.15	—	3.6	V	PIC18F4XJ13, PIC18F2XJ13
D001A	VDD	Supply Voltage	2.0	—	3.6	V	PIC18LF4XJ13, PIC18LF2XJ13
D001B	VDDCORE	External Supply for Microcontroller Core	2.0	—	2.75	V	PIC18LF4XJ13, PIC18LF2XJ13
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	VBOR	VDDCORE Brown-out Reset Voltage	—	2.0	2.15	V	PIC18F4XJ13, PIC18F2XJ13 only
D006	VDSBOR	VDD Brown-out Reset Voltage	—	1.8	—	V	DSBOREN = `1` on “LF” device or “F” device in Deep Sleep

Note 1: This is the limit to which VDDCORE can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial)

PIC18LF47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial
PIC18F47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial

Param No.	Device	Typ	Max	Units	Conditions		
	Power-Down Current (IPD)(1) – Sleep mode						
	PIC18LFXXJ13	0.1	1.6	µA	-40°C	VDD = 2.0V, VDDCORE = 2.0V	**Sleep mode**, REGSLP = 1
		0.2	1.6	µA	+25°C		
		0.8	7.0	µA	+60°C		
		2.1	11.5	µA	+85°C		
	PIC18LFXXJ13	0.2	2.0	µA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		0.5	2.0	µA	+25°C		
		1.4	9.0	µA	+60°C		
		3.2	15.0	µA	+85°C		
	PIC18FXXJ13	3.0	6.0	µA	-40°C	VDD = 2.15V VDDCORE = 10 µF Capacitor	
		3.8	6.0	µA	+25°C		
		4.7	9.0	µA	+60°C		
		6.4	18.5	µA	+85°C		
	PIC18FXXJ13	3.3	9.0	µA	-40°C	VDD = 3.3V VDDCORE = 10 µF Capacitor	
		4.2	9.0	µA	+25°C		
		5.5	12.0	µA	+60°C		
		7.8	22.0	µA	+85°C		
	Power-Down Current (IPD)(1) – Deep Sleep mode						
	PIC18FXXJ13	2	25	nA	-40°C	VDD = 2.15V, VDDCORE = 10 µF Capacitor	**Deep Sleep** mode
		9	100	nA	+25°C		
		72	250	nA	+60°C		
		262	1000	nA	+85°C		
	PIC18FXXJ13	17	50	nA	-40°C	VDD = 3.3V, VDDCORE = 10 µF Capacitor	
		53	150	nA	+25°C		
		186	400	nA	+60°C		
		503	2000	nA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). The test conditions for all IDD measurements in active operation mode are:

$\overline{\text{OSC1}}$ = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD/VSS;
$\overline{\text{MCLR}}$ = VDD; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial) (Continued)

PIC18LF47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18F47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Supply Current (IDD)[2]						
	PIC18LFXXJ13	5.5	14.2	µA	-40°C	VDD = 2.0V, VDDCORE = 2.0V	FOSC = 31 kHz **RC_RUN** mode, Internal RC Oscillator, INTSRC = 0
		5.8	14.2	µA	+25°C		
		7.9	19.0	µA	+85°C		
	PIC18LFXXJ13	8.4	16.5	µA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		8.5	16.5	µA	+25°C		
		11.3	25.0	µA	+85°C		
	PIC18FXXJ13	23.7	60.0	µA	-40°C	VDD = 2.15V VDDCORE = 10 µF Capacitor	
		27.8	60.0	µA	+25°C		
		34.0	70.0	µA	+85°C		
	PIC18FXXJ13	26.1	70.0	µA	-40°C	VDD = 3.3V VDDCORE = 10 µF Capacitor	
		29.6	70.0	µA	+25°C		
		36.2	96.0	µA	+85°C		
	PIC18LFXXJ13	0.87	1.5	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0	FOSC = 4 MHz, **RC_RUN** mode, Internal RC Oscillator
		0.91	1.5	mA	+25°C		
		0.95	1.6	mA	+85°C		
	PIC18LFXXJ13	1.23	2.0	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		1.24	2.0	mA	+25°C		
		1.25	2.0	mA	+85°C		
	PIC18FXXJ13	0.99	2.4	mA	-40°C	VDD = 2.15V, VDDCORE = 10 µF capacitor	
		1.02	2.4	mA	+25°C		
		1.06	2.6	mA	+85°C		
	PIC18FXXJ13	1.31	2.6	mA	-40°C	VDD = 3.3V, VDDCORE = 10 µF capacitor	
		1.25	2.6	mA	+25°C		
		1.26	2.7	mA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). 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/VSS;
$\overline{\text{MCLR}}$ = VDD; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial) (Continued)

PIC18LF47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial
PIC18F47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial

Param No.	Device	Typ	Max	Units	Conditions		
	Supply Current (IDD)(2)						
	PIC18LFXXJ13	1.45	3.0	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V	FOSC = 8 MHz, **RC_RUN** mode, Internal RC Oscillator
		1.48	3.0	mA	+25°C		
		1.52	3.0	mA	+85°C		
	PIC18LFXXJ13	2.12	3.8	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		2.10	3.8	mA	+25°C		
		2.10	3.9	mA	+85°C		
	PIC18FXXJ13	1.65	3.6	mA	-40°C	VDD = 2.15V, VDDCORE = 10 μF	
		1.68	3.6	mA	+25°C		
		1.71	3.9	mA	+85°C		
	PIC18FXXJ13	2.26	4.3	mA	-40°C	VDD = 3.3V, VDDCORE = 10 μF	
		2.13	4.3	mA	+25°C		
		2.10	4.6	mA	+85°C		
	PIC18LFXXJ13	1.5	11	μA	-40°C	VDD = 2.0V, VDDCORE = 2.0V	FOSC = 31 kHz, **RC_IDLE** mode, Internal RC Oscillator, INTSRC = 0
		1.7	11	μA	+25°C		
		3.9	20	μA	+85°C		
	PIC18LFXXJ13	2.0	12	μA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		2.4	13	μA	+25°C		
		5.4	23	μA	+85°C		
	PIC18FXXJ13	19.0	60	μA	-40°C	VDD = 2.15V, VDDCORE = 10 μF	
		23.2	60	μA	+25°C		
		29.9	75	μA	+85°C		
	PIC18FXXJ13	19.7	65	μA	-40°C	VDD = 3.3V, VDDCORE = 10 μF	
		24.0	65	μA	+25°C		
		31.4	90	μA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). 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/VSS;
$\overline{\text{MCLR}}$ = VDD; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial) (Continued)

PIC18LF47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial
PIC18F47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial

Param No.	Device	Typ	Max	Units	Conditions		
	Supply Current (IDD)(2)						
	PIC18LFXXJ13	0.41	0.98	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V	FOSC = 4 MHz, **RC_IDLE** mode, Internal RC Oscillator
		0.44	0.98	mA	+25°C		
		0.48	1.12	mA	+85°C		
	PIC18LFXXJ13	0.48	1.14	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		0.51	1.14	mA	+25°C		
		0.55	1.25	mA	+85°C		
	PIC18FXXJ13	0.45	1.21	mA	-40°C	VDD = 2.15V, VDDCORE = 10 μF	
		0.49	1.21	mA	+25°C		
		0.52	1.30	mA	+85°C		
	PIC18FXXJ13	0.52	1.20	mA	-40°C	VDD = 3.3V, VDDCORE = 10 μF	
		0.54	1.20	mA	+25°C		
		0.58	1.35	mA	+85°C		
	PIC18LFXXJ13	0.53	1.4	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V	FOSC = 8 MHz, **RC_IDLE** mode, Internal RC Oscillator
		0.56	1.5	mA	+25°C		
		0.60	1.6	mA	+85°C		
	PIC18LFXXJ13	0.63	2.0	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		0.67	2.0	mA	+25°C		
		0.72	2.2	mA	+85°C		
	PIC18FXXJ13	0.58	1.8	mA	-40°C	VDD = 2.15V, VDDCORE = 10 μF	
		0.62	1.8	mA	+25°C		
		0.66	2.0	mA	+85°C		
	PIC18FXXJ13	0.69	2.2	mA	-40°C	VDD = 3.3V, VDDCORE = 10 μF	
		0.70	2.2	mA	+25°C		
		0.74	2.3	mA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). 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/VSS;
$\overline{\text{MCLR}}$ = VDD; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial) (Continued)

PIC18LF47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18F47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Supply Current (IDD)[2]						
	PIC18LFXXJ13	0.61	1.25	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V	FOSC = 4 MHz, **PRI_RUN** mode, EC Oscillator
		0.62	1.25	mA	+25°C		
		0.64	1.35	mA	+85°C		
	PIC18LFXXJ13	0.99	1.70	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		0.96	1.70	mA	+25°C		
		0.94	1.82	mA	+85°C		
	PIC18FXXJ13	0.78	1.60	mA	-40°C	VDD = 2.15V, VDDCORE = 10 µF	
		0.78	1.60	mA	+25°C		
		0.78	1.70	mA	+85°C		
	PIC18FXXJ13	1.10	1.95	mA	-40°C	VDD = 3.3V, VDDCORE = 10 µF	
		1.02	1.90	mA	+25°C		
		1.00	2.00	mA	+85°C		
	PIC18LFXXJ13	9.8	14.8	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	FOSC = 48 MHz, **PRI_RUN** mode, EC Oscillator
		9.5	14.8	mA	+25°C		
		9.4	15.1	mA	+85°C		
	PIC18FXXJ13	10.9	19.5	mA	-40°C	VDD = 3.3V, VDDCORE = 10 µF	
		10.2	19.5	mA	+25°C		
		9.9	19.5	mA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). The test conditions for all IDD measurements in active operation mode are:

$\overline{\text{OSC1}}$ = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD/VSS;
$\overline{\text{MCLR}}$ = VDD; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial) (Continued)

PIC18LF47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial
PIC18F47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial

Param No.	Device	Typ	Max	Units	Conditions		
	PIC18LFXXJ13	0.18	0.70	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V	FOSC = 4 MHz, **PRI_IDLE** mode, EC Oscillator
		0.18	0.70	mA	+25°C		
		0.19	0.75	mA	+85°C		
	PIC18LFXXJ13	0.23	0.90	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	
		0.23	0.90	mA	+25°C		
		0.24	1.00	mA	+85°C		
	PIC18FXXJ13	0.23	0.80	μA	-40°C	VDD = 2.15V, VDDCORE = 10 μF Capacitor	
		0.24	0.80	μA	+25°C		
		0.25	0.85	μA	+85°C		
	PIC18FXXJ13	0.32	1.00	μA	-40°C	VDD = 3.3V, VDDCORE = 10 μF Capacitor	
		0.31	1.00	μA	+25°C		
		0.32	1.00	μA	+85°C		
	PIC18LFXXJ13	2.74	7.00	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	FOSC = 48 MHz, **PRI_IDLE** mode, EC Oscillator
		2.77	6.50	mA	+25°C		
		2.80	6.50	mA	+85°C		
	PIC18FXXJ13	3.48	11.00	mA	-40°C	VDD = 3.3V, VDDCORE = 10 μF Capacitor	
		3.48	10.00	mA	+25°C		
		3.44	10.00	mA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). 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/VSS;
$\overline{\text{MCLR}}$ = VDD; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial) (Continued)

PIC18LF47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18F47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	PIC18LFXXJ13	4.1	6.5	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	FOSC = 16 MHz, **PRI_RUN** mode, 4 MHz Internal Oscillator with PLL
		4.0	6.4	mA	+25°C		
		4.0	6.4	mA	+85°C		
	PIC18FXXJ13	4.5	7.5	mA	-40°C	VDD = 3.3V, VDDCORE = 10 μF	
		4.2	7.4	mA	+25°C		
		4.1	7.4	mA	+85°C		
	PIC18LFXXJ13	7.8	12.0	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	FOSC = 32 MHz, **PRI_RUN** mode, 8 MHz Internal Oscillator with PLL
		7.7	11.8	mA	+25°C		
		7.5	11.8	mA	+85°C		
	PIC18FXXJ13	8.6	17.5	mA	-40°C	VDD = 3.3V, VDDCORE = 10 μF	
		7.8	17.2	mA	+25°C		
		7.5	17.2	mA	+85°C		
	PIC18LFXXJ13	9.4	14.0	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V	FOSC = 48 MHz, **PRI_RUN** mode, 12 MHz External Oscillator with PLL
		9.3	14.0	mA	+25°C		
		9.1	14.0	mA	+85°C		
	PIC18FXXJ13	11.1	21.5	mA	-40°C	VDD = 3.3V, VDDCORE = 10 μF	
		10.3	20.5	mA	+25°C		
		9.9	20.5	mA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). 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/VSS;
$\overline{\text{MCLR}}$ = VDD; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial) (Continued)

PIC18LF47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial					
PIC18F47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	PIC18LFXXJ13	9	45	μA	-40°C	V_{DD} = 2.5V, V_{DDCORE} = 2.5V	F_{OSC} = 32 kHz[3], **SEC_RUN** mode, SOSCSEL = `0b01`
		9	45	μA	+25°C		
		12	61	μA	+85°C		
	PIC18FXXJ13	24	95	μA	-40°C	V_{DD} = 2.15V, V_{DDCORE} = 10 μF	
		28	95	μA	+25°C		
		35	105	μA	+85°C		
	PIC18FXXJ13	27	110	μA	-40°C	V_{DD} = 3.3V, V_{DDCORE} = 10 μF	
		31	110	μA	+25°C		
		35	150	μA	+85°C		
	PIC18LFXXJ13	2.5	31	μA	-40°C	V_{DD} = 2.5V, V_{DDCORE} = 2.5V	F_{OSC} = 32 kHz[3], **SEC_IDLE** mode, SOSCSEL = `0b01`
		3.0	31	μA	+25°C		
		6.1	50	μA	+85°C		
	PIC18FXXJ13	19	87	μA	-40°C	V_{DD} = 2.15V, V_{DDCORE} = 10 μF	
		24	89	μA	+25°C		
		31	97	μA	+85°C		
	PIC18FXXJ13	21	100	μA	-40°C	V_{DD} = 3.3V, V_{DDCORE} = 10 μF	
		25	100	μA	+25°C		
		31	140	μA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to V_{DD} or V_{SS}, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). The test conditions for all I_{DD} measurements in active operation mode are:

OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V_{DD}/V_{SS};
$\overline{MCLR}$ = V_{DD}; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial) (Continued)

PIC18LF47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial
PIC18F47J13 Family	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial

Param No.	Device	Typ	Max	Units	Conditions		
	Module Differential Currents (ΔI_{WDT}, ΔI_{HLVD}, ΔI_{OSCB}, ΔI_{AD})						
D022 (ΔI_{WDT})	**Watchdog Timer**	0.38	4.0	µA	-40°C	V_{DD} = 2.5V, V_{DDCORE} = 2.5V	PIC18LFXXJ13
		0.41	4.0	µA	+25°C		
		0.44	5.2	µA	+85°C		
		0.33	4.0	µA	-40°C	V_{DD} = 2.15V, V_{DDCORE} = 10 µF Capacitor	PIC18FXXJ13
		0.33	4.0	µA	+25°C		
		0.39	6.0	µA	+85°C		
		0.49	8.0	µA	-40°C	V_{DD} = 3.3V, V_{DDCORE} = 10 µF Capacitor	
		0.48	8.0	µA	+25°C		
		0.45	9.0	µA	+85°C		
D022B (ΔI_{HLVD})	**High/Low-Voltage Detect**	4.1	8.0	µA	-40°C	V_{DD} = 2.5V, V_{DDCORE} = 2.5V	PIC18LFXXJ13
		4.9	8.0	µA	+25°C		
		5.8	9.0	µA	+85°C		
		2.6	6.0	µA	-40°C	V_{DD} = 2.15V, V_{DDCORE} = 10 µF Capacitor	PIC18FXXJ13
		3.0	6.0	µA	+25°C		
		3.5	8.0	µA	+85°C		
		3.4	9.0	µA	-40°C	V_{DD} = 3.3V, V_{DDCORE} = 10 µF Capacitor	
		3.9	9.0	µA	+25°C		
		4.2	12	µA	+85°C		
D025 (ΔI_{OSCB})	**Real-Time Clock/Calendar with Low-Power Timer1 Oscillator**	0.5	4.5	µA	-40°C	V_{DD} = 2.15V, V_{DDCORE} = 10 µF Capacitor	PIC18FXXJ13 32.768 kHz[3], T1OSCEN = `1`, SOSCSEL = `0b01`
		0.7	5.0	µA	+25°C		
		0.8	5.0	µA	+60°C		
		0.9	5.0	µA	+85°C		
		0.6	4.0	µA	-40°C	V_{DD} = 2.5V, V_{DDCORE} = 10 µF Capacitor	
		0.8	4.5	µA	+25°C		
		0.9	4.5	µA	+60°C		
		1.0	4.5	µA	+85°C		
		0.8	6.5	µA	-40°C	V_{DD} = 3.3V, V_{DDCORE} = 10 µF Capacitor	
		1.0	6.5	µA	+25°C		
		1.1	8.0	µA	+60°C		
		1.3	8.0	µA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to V_{DD} or V_{SS}, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). The test conditions for all I_{DD} measurements in active operation mode are:

$\overline{\text{OSC1}}$ = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V_{DD}/V_{SS};
$\overline{\text{MCLR}}$ = V_{DD}; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.2 DC Characteristics: Power-Down and Supply Current PIC18F47J13 Family (Industrial) (Continued)

PIC18LF47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial					
PIC18F47J13 Family		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Module Differential Currents (ΔI_{WDT}, ΔI_{HLVD}, ΔI_{OSCB}, ΔI_{AD})						
D026 (ΔI_{AD})	**A/D Converter**	0.5	4	µA	-40°C	V_{DD} = 2.5V, V_{DDCORE} = 2.5V	PIC18LFXXJ13 A/D on, not converting
		0.5	4	µA	+25°C		
		0.5	4	µA	+85°C		
		1.1	5	µA	-40°C	V_{DD} = 2.15V, V_{DDCORE} = 10 µF Capacitor	PIC18FXXJ13 A/D on, not converting
		1.1	5	µA	+25°C		
		1.1	5	µA	+85°C		
		3.2	11	µA	-40°C	V_{DD} = 3.3V, V_{DDCORE} = 10 µF Capacitor	
		3.2	11	µA	+25°C		
		3.2	11	µA	+85°C		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to V_{DD} or V_{SS}, and all features that add delta current are disabled (such as WDT, Timer1 oscillator, BOR, etc.).

2: The supply current is mainly a function of 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. All features that add delta current are disabled (such as WDT). The test conditions for all I_{DD} measurements in active operation mode are:

$\overline{\text{OSC1}}$ = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to V_{DD}/V_{SS};
$\overline{\text{MCLR}}$ = V_{DD}; WDT disabled unless otherwise specified.

3: Low-power Timer1 with 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.

30.3 DC Characteristics: PIC18F47J13 Family (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial			
Param No.	**Symbol**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**				
		All I/O Ports:				
D030		with TTL Buffer	VSS	0.15 VDD	V	VDD < 3.3V
D030A		with TTL Buffer	VSS	0.8	V	3.3V ≤ VDD ≤ 3.6V
D031		with Schmitt Trigger Buffer	VSS	0.2 VDD	V	
D031A		SCLx/SDAx	—	0.3 VDD	V	I^{2}C enabled
D031B		SCLx/SDAx	—	0.8	V	SMBus enabled
D032		$\overline{\text{MCLR}}$	VSS	0.2 VDD	V	
D033		OSC1	VSS	0.3 VDD	V	HS, HSPLL modes
D033A		OSC1	VSS	0.2 VDD	V	EC, ECPLL modes
D034		T1OSI	VSS	0.3	V	T1OSCEN = 1
	VIH	**Input High Voltage**				
		I/O Ports without 5.5V Tolerance:				
D040		with TTL Buffer	0.25 VDD + 0.8V	VDD	V	VDD < 3.3V
D040A		with TTL Buffer	2.0	VDD	V	3.3V ≤ VDD ≤ 3.6V
D041		with Schmitt Trigger Buffer	0.8 VDD	VDD	V	
		I/O Ports with 5.5V Tolerance:$^{(4)}$				
Dxxx		with TTL Buffer	0.25 VDD + 0.8V	5.5	V	VDD < 3.3V
DxxxA			2.0	5.5	V	3.3V ≤ VDD ≤ 3.6V
Dxxx		with Schmitt Trigger Buffer	0.8 VDD	5.5	V	
D041A		SCLx/SDAx	0.7 VDD	—	V	I^{2}C enabled
D041B		SCLx/SDAx	2.1	—	V	SMBus enabled; VDD ≥ 3V
D042		$\overline{\text{MCLR}}$	0.8 VDD	5.5	V	
D043		OSC1	0.7 VDD	VDD	V	HS, HSPLL modes
D043A		OSC1	0.8 VDD	VDD	V	EC, ECPLL modes
D044		T1OSI	1.6	VDD	V	T1OSCEN = 1
	IPU	**Weak Pull-up Current**				
D070	IPURB	PORTB, PORTD$^{(3)}$ and PORTE$^{(3)}$ Weak Pull-up Current	80	400	μA	VDD = 3.3V, VPIN = VSS

Note 1: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

2: Negative current is defined as current sourced by the pin.

3: Only available in 44-pin devices.

4: Refer to Table 10-2 for pin tolerance levels.

30.3 DC Characteristics: PIC18F47J13 Family (Industrial) (Continued)

DC CHARACTERISTICS			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial			
Param No.	**Symbol**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
	V_{OL}	**Output Low Voltage**				
D080		I/O Ports:				
		PORTA (except RA6), PORTD(3), PORTE(3)	—	0.4	V	I_{OL} = 4 mA, V_{DD} = 3.3V, -40°C to +85°C
		PORTB, PORTC, RA6	—	0.4	V	I_{OL} = 8.5 mA, V_{DD} = 3.3V, -40°C to +85°C
	V_{OH}	**Output High Voltage**				
D090		I/O Ports:			V	
		PORTA (except RA6), PORTD(3), PORTE(3)	2.4	—	V	I_{OH} = -3 mA, V_{DD} = 3.3V, -40°C to +85°C
		PORTB, PORTC, RA6	2.4	—	V	I_{OH} = -6 mA, V_{DD} = 3.3V, -40°C to +85°C
		Capacitive Loading Specs on Output Pins				
D101	C_{IO}	All I/O Pins and OSC2	—	50	pF	To meet the AC Timing Specifications
D102	C_B	SCLx, SDAx	—	400	pF	I^2C Specification

Note 1: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

2: Negative current is defined as current sourced by the pin.

3: Only available in 44-pin devices.

4: Refer to Table 10-2 for pin tolerance levels.

30.4 DC Characteristics: PIC18F47J13 Family (Industrial)

DC CHARACTERISTICS			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial			
Param No.	**Symbol**	**Characteristic**	**Typ**	**Max**	**Units**	**Conditions**
	I_{IL}	**Input Leakage Current**$^{(1,2)}$				
D060		I/O Ports without 5.5V Tolerance	±5	±200	nA	$V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance
		I/O Ports with 5.5V Tolerance	±5	±200	nA	$V_{SS} \le V_{PIN} \le 5.5V$, Pin at high-impedance
D061		$\overline{\text{MCLR}}$	±5	±200	nA	$V_{ss} \le V_{PIN} \le V_{DD}$
D063		OSC1	±5	±200	nA	$V_{ss} \le V_{PIN} \le V_{DD}$

Note 1: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

2: Negative current is defined as current sourced by the pin.

TABLE 30-1: MEMORY PROGRAMMING REQUIREMENTS

DC CHARACTERISTICS			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Program Flash Memory					
D130	E_P	Cell Endurance	10K	—	—	E/W	-40°C to +85°C
D131	V_{PR}	V_{DD}core for Read	V_{MIN}	—	2.75	V	V_{MIN} = Minimum operating voltage
D132B	V_{PEW}	V_{DDCORE} for Self-Timed Erase or Write	2.25	—	2.75	V	
D133A	T_{IW}	Self-Timed Write Cycle Time	—	2.8	—	ms	64 bytes
D133B	T_{IE}	Self-Timed Block Erase Cycle Time	—	33.0	—	ms	
D134	T_{RETD}	Characteristic Retention	20	—	—	Year	Provided no other specifications are violated
D135	I_{DDP}	Supply Current during Programming	—	3	—	mA	

† Data in “Typ” column is at 3.3V, 25°C unless otherwise stated.

TABLE 30-2: COMPARATOR SPECIFICATIONS

Operating Conditions: 3.0V < VDD < 3.6V, -40°C < TA < +85°C (unless otherwise stated)

Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D300	VIOFF	Input Offset Voltage	—	±5	±25	mV	
D301	VICM	Input Common Mode Voltage	0	—	VDD	V	
	VIRV	Internal Reference Voltage	0.57	0.60	0.63	V	
D302	CMRR	Common Mode Rejection Ratio	55	—	—	dB	
D303	TRESP	Response Time(1)	—	150	400	ns	
D304	TMC2OV	Comparator Mode Change to Output Valid	—	—	10	μs	

Note 1: Response time measured with one comparator input at VDD/2, while the other input transitions from VSS to VDD.

TABLE 30-3: VOLTAGE REFERENCE SPECIFICATIONS

TABLE 4-1:

Operating Conditions: 3.0V < VDD < 3.6V, -40°C < TA < +85°C (unless otherwise stated)

Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
D310	VRES	Resolution	VDD/24	—	VDD/32	LSb	
D311	VRAA	Absolute Accuracy	—	—	1/2	LSb	
D312	VRUR	Unit Resistor Value (R)	—	2k	—	Ω	
310	TSET	Settling Time(1)	—	—	10	μs	

Note 1: Settling time measured while CVRR = `1` and CVR<3:0> bits transition from ‘`0000`’ to ‘`1111`’.

TABLE 30-4: INTERNAL VOLTAGE REGULATOR SPECIFICATIONS

TABLE 4-2:

Operating Conditions: -40°C < TA < +85°C (unless otherwise stated)

Param No.	Sym	Characteristics	Min	Typ	Max	Units	Comments
	VRGOUT	Regulator Output Voltage	2.35	2.5	2.7	V	Regulator enabled, VDD = 3.0V
	CEFC	External Filter Capacitor Value(1)	5.4	10	18	μF	ESR < 3Ω recommended ESR < 5Ω required

Note 1: CEFC applies for PIC18**F** devices in the family. For PIC18**LF** devices in the family, there is no specific minimum or maximum capacitance for VDDCORE, although proper supply rail bypassing should still be used.

TABLE 30-5: ULPWU SPECIFICATIONS

DC CHARACTERISTICS			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D100	IULP	**Ultra Low-Power Wake-up Current**	—	60	—	nA	Net of I/O leakage and current sink at 1.6V on pin, VDD = 3.3V See Application Note *AN879,* “*Using the Microchip Ultra Low-Power Wake-up Module*” (DS00879)

† Data in “Typ” column is at 3.3V, 25°C unless otherwise stated.

TABLE 30-6: CTMU CURRENT SOURCE SPECIFICATIONS

DC CHARACTERISTICS			**Standard Operating Conditions: 2.0V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
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`).

FIGURE 30-3: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

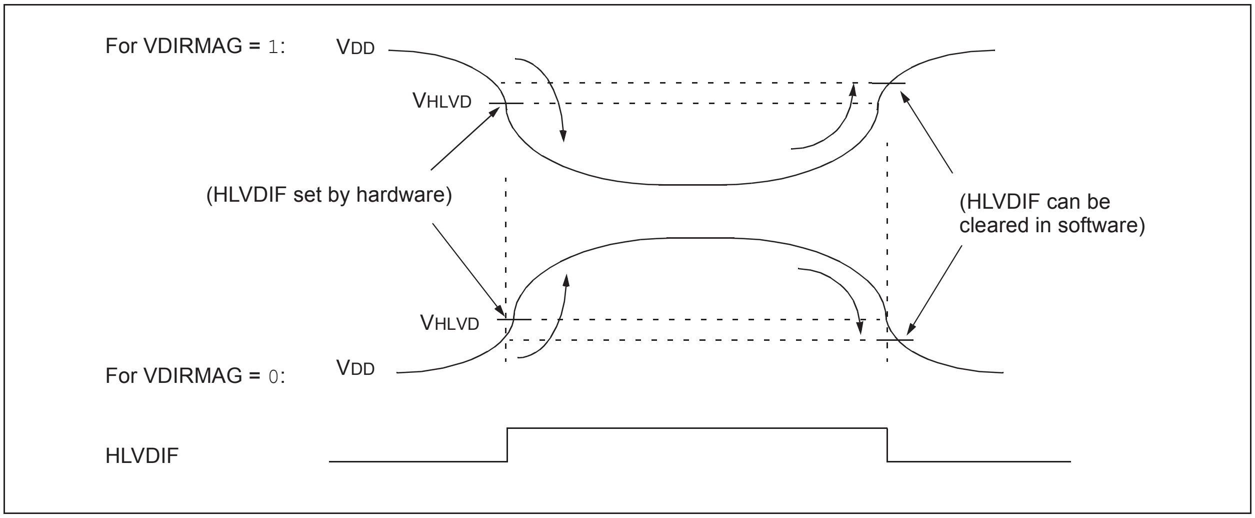

For VDIRMAG = 1:
VDD
VHLVD
(HLVDIF set by hardware)
(HLVDIF can be cleared in software)
VHLVD
For VDIRMAG = 0:
VDD
HLVDIF

TABLE 30-7: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

Standard Operating Conditions (unless otherwise stated)
Operating temperature -40°C ≤ TA ≤ +85°C for industrial

Param No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
D420		HLVD Voltage on VDD Transition High-to-Low	HLVDL<3:0> = `1000`	2.33	2.45	2.57	V	
			HLVDL<3:0> = `1001`	2.47	2.60	2.73	V	
			HLVDL<3:0> = `1010`	2.66	2.80	2.94	V	
			HLVDL<3:0> = `1011`	2.76	2.90	3.05	V	
			HLVDL<3:0> = `1100`	2.85	3.00	3.15	V	
			HLVDL<3:0> = `1101`	2.97	3.13	3.29	V	
			HLVDL<3:0> = `1110`	3.23	3.40	3.57	V	
D421	TIRVST	Time for Internal Reference Voltage to become Stable		—	50	—	μs	
D422	TLVD	High/Low-Voltage Detect Pulse Width		200	—	—	μs	

30.5 AC (Timing) Characteristics

30.5.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST (I^{2}C specifications only)
4. Ts (I^{2}C 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	T13CKI
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^{2}C only			
AA	output access	High	High
BUF	Bus free	Low	Low

TCC:ST (I^{2}C specifications only)

CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	Stop condition
STA	Start condition		

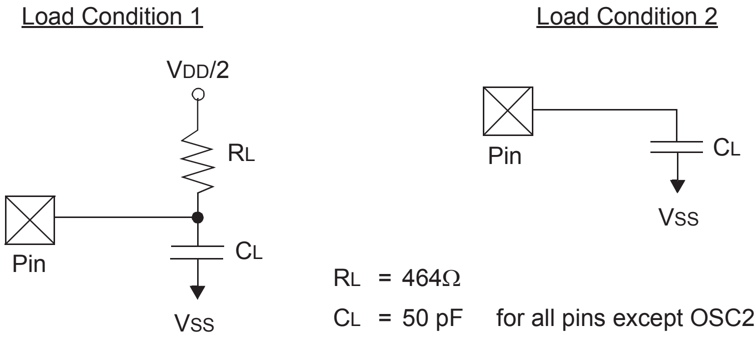
30.5.2 TIMING CONDITIONS

The temperature and voltages specified in Table 30-8 apply to all timing specifications unless otherwise noted. Figure 30-4 specifies the load conditions for the timing specifications.

TABLE 30-8: TEMPERATURE AND VOLTAGE SPECIFICATIONS – AC

AC CHARACTERISTICS	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial Operating voltage VDD range as described in **Section 30.1** and **Section 30.3**.

FIGURE 30-4: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS



Load Condition 1

Load Condition 2

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

30.5.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 30-5: EXTERNAL CLOCK TIMING

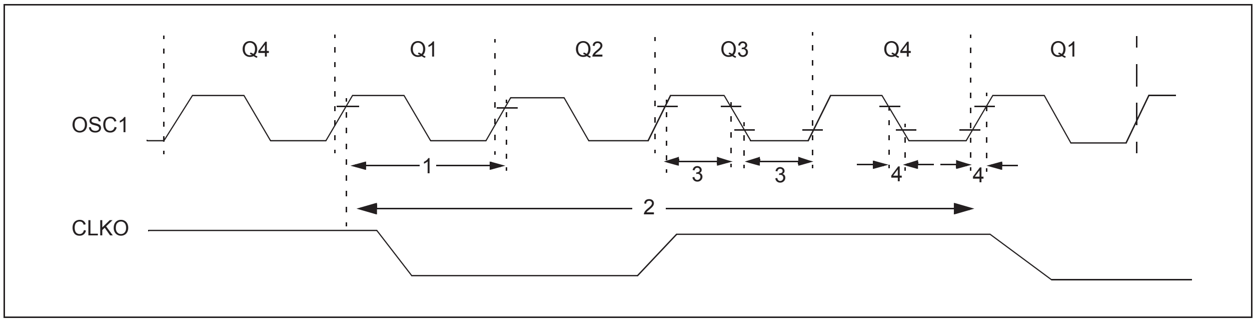


TABLE 30-9: EXTERNAL CLOCK TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	FOSC	External CLKI Frequency(1)	DC	48	MHz	EC Oscillator mode
			DC	48		ECPLL Oscillator mode(2)
		Oscillator Frequency(1)	4	16	MHz	HS Oscillator mode
			4	16(4)		HSPLL Oscillator mode(3)
1	TOSC	External CLKI Period(1)	20.8	—	ns	EC Oscillator mode
			20.8	—		ECPLL Oscillator mode(2)
		Oscillator Period(1)	62.5	250	ns	HS Oscillator mode
			62.5(4)	250		HSPLL Oscillator mode(3)
2	TCY	Instruction Cycle Time(1)	83.3	DC	ns	TCY = 4/FOSC, Industrial
3	TOSL, TOSH	External Clock in (OSC1) High or Low Time	10	—	ns	EC Oscillator mode
4	TOSR, TOSF	External Clock in (OSC1) Rise or Fall Time	—	7.5	ns	EC Oscillator mode

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations except PLL. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

2: In order to use the PLL, the external clock frequency must be either 4, 8, 12, 16, 20, 24, 40 or 48 MHz.

3: In order to use the PLL, the crystal/resonator must produce a frequency of either 4, 8, 12 or 16 MHz.

4: This is the maximum crystal/resonator driver frequency. The internal FOSC frequency, when running from the PLL, can be up to 48 MHz.

TABLE 30-10: 96 MHz PLL CLOCK TIMING SPECIFICATIONS (VDDCORE = 2.35V TO 2.75V)

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
F10	FOSC	Oscillator Frequency Range	4	—	48	MHz	
F11	FSYS	On-Chip VCO System Frequency	—	96	—	MHz	
F12	t_{rc}	PLL Start-up Time (lock time)	—	—	2	ms	

TABLE 30-11: 4x PLL CLOCK TIMING SPECIFICATIONS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
F10	FPLLIN	PLL Input Frequency Range	4	—	12	MHz	
F11	FPLLO	PLL Output Frequency (4x FPLLIN)	16	—	48	MHz	
F12	t_{rc}	PLL Start-up Time (lock time)	—	—	2	ms	

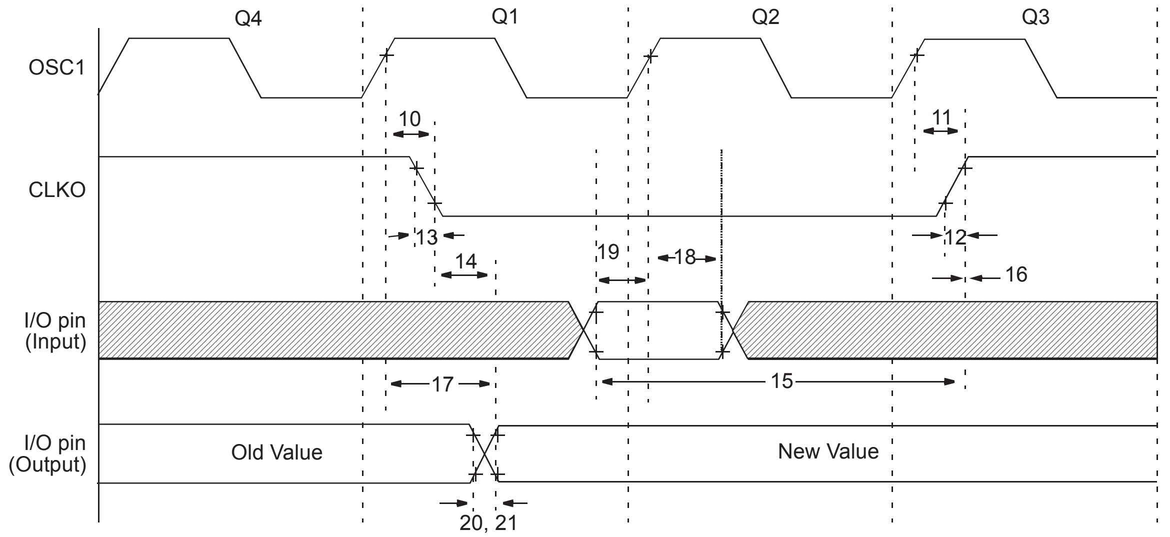
† Data in “Typ” column is at 3.3V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

TABLE 30-12: INTERNAL RC ACCURACY (INTOSC AND INTRC SOURCES)

Param No.	Device	Min	Typ	Max	Units	Conditions	
	INTOSC Accuracy @ Freq = 8 MHz, 4 MHz, 2 MHz, 1 MHz, 500 kHz, 250 kHz, 125 kHz, 31 kHz[(1)]						
	All Devices	-1	±0.15	+1	%	0°C to +85°C	VDD = 2.4V-3.6V, VDDCORE = 2.3V-2.7V
		-1	±0.25	+1	%	-40°C to +85°C	VDD = 2.0V-3.6V, VDDCORE = 2.0V-2.7V
	INTRC Accuracy @ Freq = 31 kHz[(1)]						
	All Devices	20.3	—	42.2	kHz	-40°C to +85°C	VDD = 2.0V-3.6V, VDDCORE = 2.0V-2.7V

Note 1: The accuracy specification of the 31 kHz clock is determined by which source is providing it at a given time. When INTSRC (OSCTUNE<7>) is ‘`1`’, use the INTOSC accuracy specification. When INTSRC is ‘`0`’, use the INTRC accuracy specification.

FIGURE 30-6: CLKO AND I/O TIMING



Note: Refer to Figure 30-4 for load conditions.

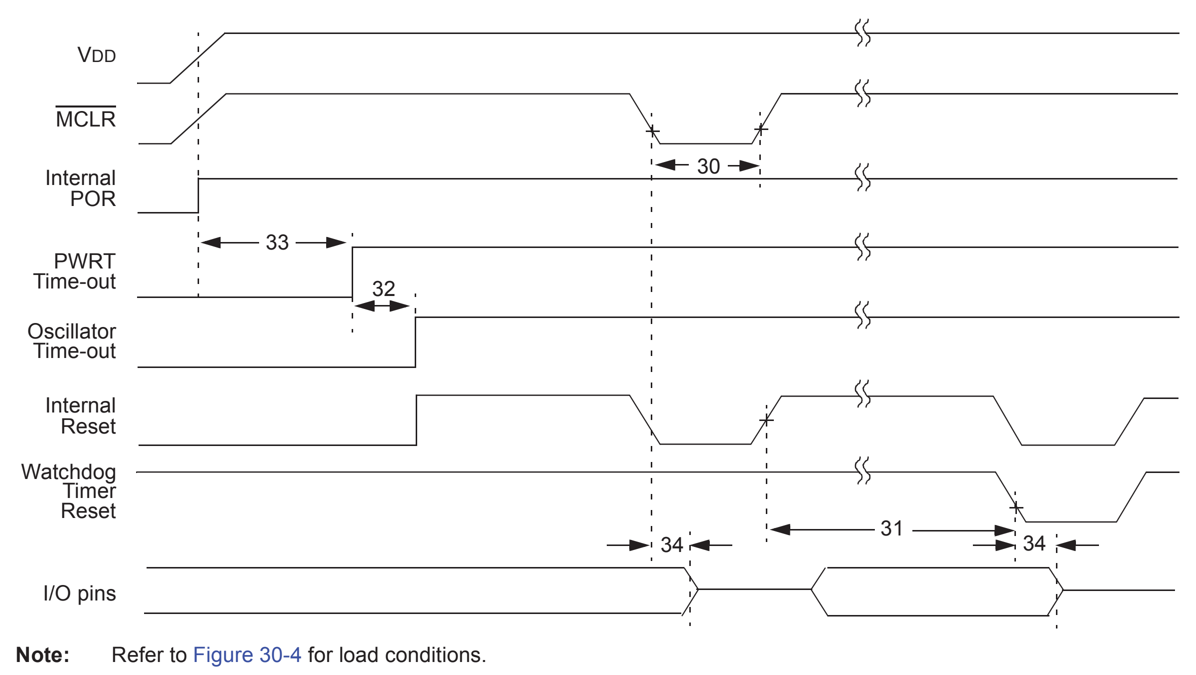
TABLE 30-13: CLKO AND I/O TIMING REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
10	TOSH2CKL	OSC1 ↑ to CLKO ↓	—	75	200	ns	**(Note 1)**
11	TOSH2CKH	OSC1 ↑ to CLKO ↑	—	75	200	ns	**(Note 1)**
12	TCKR	CLKO Rise Time	—	15	30	ns	**(Note 1)**
13	TCKF	CLKO Fall Time	—	15	30	ns	**(Note 1)**
14	TCKL2IOV	CLKO ↓ to Port Out Valid	—	—	0.5 TCY + 20	ns	
15	TIOV2CKH	Port In Valid before CLKO ↑	0.25 TCY + 25	—	—	ns	
16	TCKH2IOI	Port In Hold after CLKO ↑	0	—	—	ns	
17	TOSH2IOV	OSC1 ↑ (Q1 cycle) to Port Out Valid	—	50	150	ns	
18	TOSH2IOI	OSC1 ↑ (Q2 cycle) to Port Input Invalid (I/O in hold time)	100	—	—	ns	
19	TIOV2OSH	Port Input Valid to OSC1 ↑ (I/O in setup time)	0	—	—	ns	
20	TIOR	Port Output Rise Time	—	—	6	ns	
21	TIOF	Port Output Fall Time	—	—	5	ns	
22†	TINP	INTx pin High or Low Time	TCY	—	—	ns	
23†	TRBP	RB<7:4> Change INTx High or Low Time	TCY	—	—	ns	

† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in EC mode, where CLKO output is 4 x TOSC.

FIGURE 30-7: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING


VDD
MCLR
Internal POR
PWRT Time-out
Oscillator Time-out
Internal Reset
Watchdog Timer Reset
I/O pins
30
33
32
34
31
34

Note: Refer to Figure 30-4 for load conditions.

TABLE 30-14: 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)	2.67	4.0	5.53	ms	
32	TOST	Oscillator Start-up Timer Period	1024 TOSC	—	1024 TOSC	—	TOSC = OSC1 period
33	TPWRT	Power-up Timer Period	— —	500 46	— —	μs ms	F Devices LF Devices
34	TIOZ	I/O High-Impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset	—	—	3 TCY + 2	μs	**(Note 1)**
36	TIRVST	Time for Internal Reference Voltage to become Stable	—	20	—	μs	
37	TLVD	High/Low-Voltage Detect Pulse Width	—	200	—	μs	
38	TCSD	CPU Start-up Time	—	200	—	μs	**(Note 2)**

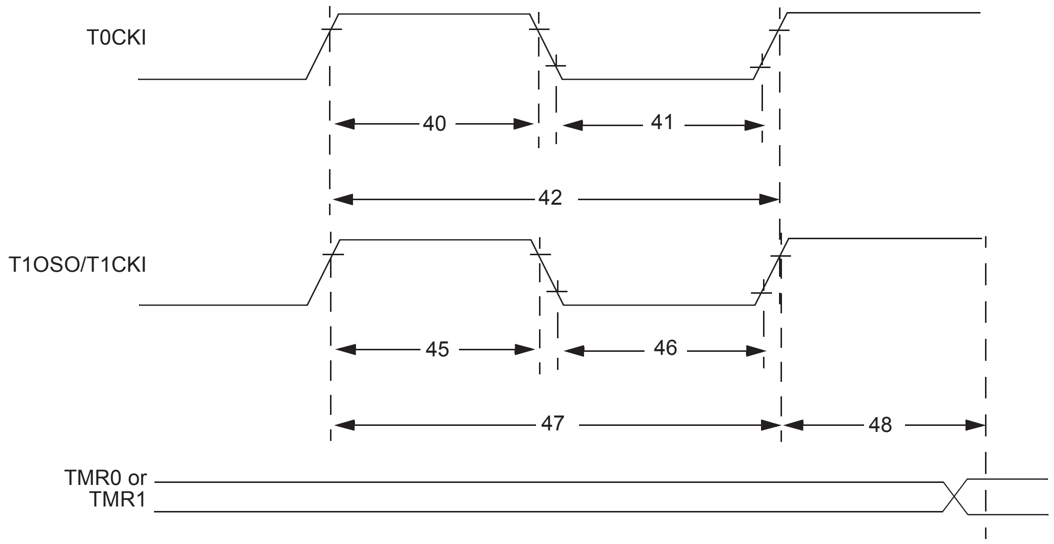
Note 1: The maximum TIOZ is the lesser of (3 TCY + 2 μs) or 700 μs.

2: $\overline{\text{MCLR}}$ rising edge to code execution, assuming TPWRT (and TOST if applicable) has already expired.

TABLE 30-15: LOW-POWER WAKE-UP TIME

Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
W1	WDS	Deep Sleep	—	500	—	μs	REGSLP = `1`
W2	WSLEEP	Sleep	—	35	—	μs	REGSLP = `1`, PLLEN = `0`, FOSC = 8 MHz INTOSC
W3	WDOZE1	Sleep	—	12	—	μs	REGSLP = `0`, PLLEN = `0`, FOSC = 8 MHz INTOSC
W4	WDOZE2	Sleep	—	1.1	—	μs	REGSLP = `0`, PLLEN = `0`, FOSC = 8 MHz EC
W5	WDOZE3	Sleep	—	230	—	ns	REGSLP = `0`, PLLEN = `0`, FOSC = 48 MHz EC
W6	WIDLE	Idle	—	230	—	ns	FOSC = 48 MHz EC

FIGURE 30-8: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS



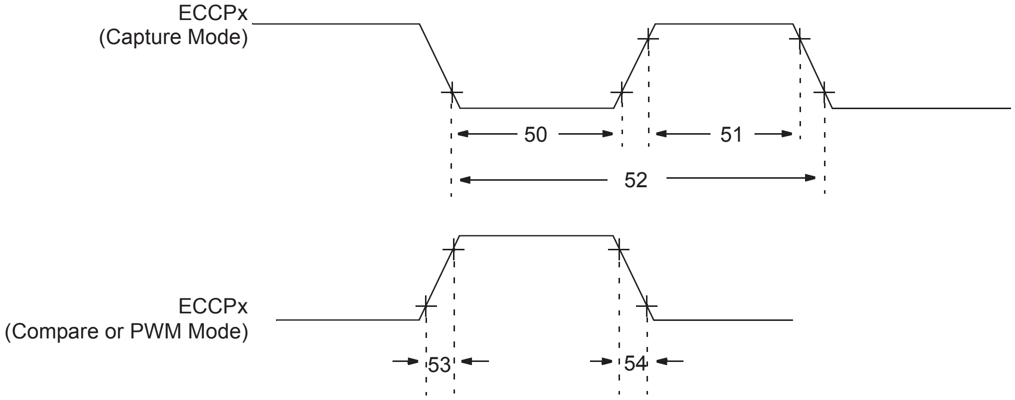
Note: Refer to Figure 30-4 for load conditions.

TABLE 30-16: 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/T3CKI High Time	Synchronous, no prescaler	0.5 TCY + 20	—	ns	
			Synchronous, with prescaler	10	—	ns	
			Asynchronous	30	—	ns	
46	TT1L	T1CKI/T3CKI Low Time	Synchronous, no prescaler	0.5 TCY + 5	—	ns	
			Synchronous, with prescaler	10	—	ns	
			Asynchronous	30	—	ns	
47	TT1P	T1CKI/T3CKI Input Period	Synchronous	Greater of: 20 ns or (TCY + 40)/N	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous	83	—	ns	
	FT1	T1CKI Input Frequency Range[(1)]		DC	12	MHz	
48	TCKE2TMRI	Delay from External T1CKI Clock Edge to Timer Increment		2 TOSC	7 TOSC	—	

Note 1: The Timer1 oscillator is designed to drive 32.768 kHz crystals. When T1CKI is used as a digital input, frequencies up to 12 MHz are supported.

FIGURE 30-9: ENHANCED CAPTURE/COMPARE/PWM TIMINGS



Note: Refer to Figure 30-4 for load conditions.

TABLE 30-17: ENHANCED CAPTURE/COMPARE/PWM REQUIREMENTS

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
50	TCCL	ECCPx Input Low Time	No prescaler	0.5 TCY + 20	—	ns	
			With prescaler	10	—	ns	
51	TCCH	ECCPx Input High Time	No prescaler	0.5 TCY + 20	—	ns	
			With prescaler	10	—	ns	
52	TCCP	ECCPx Input Period $\frac{3\,\text{TCY} + 40}{N}$		$\frac{3\,\text{TCY} + 40}{N}$	—	ns	N = prescale value (1, 4 or 16)
53	TCCR	ECCPx Output Fall Time		—	25	ns	
54	TCCF	ECCPx Output Fall Time		—	25	ns	

FIGURE 30-10: PARALLEL MASTER PORT READ TIMING DIAGRAM

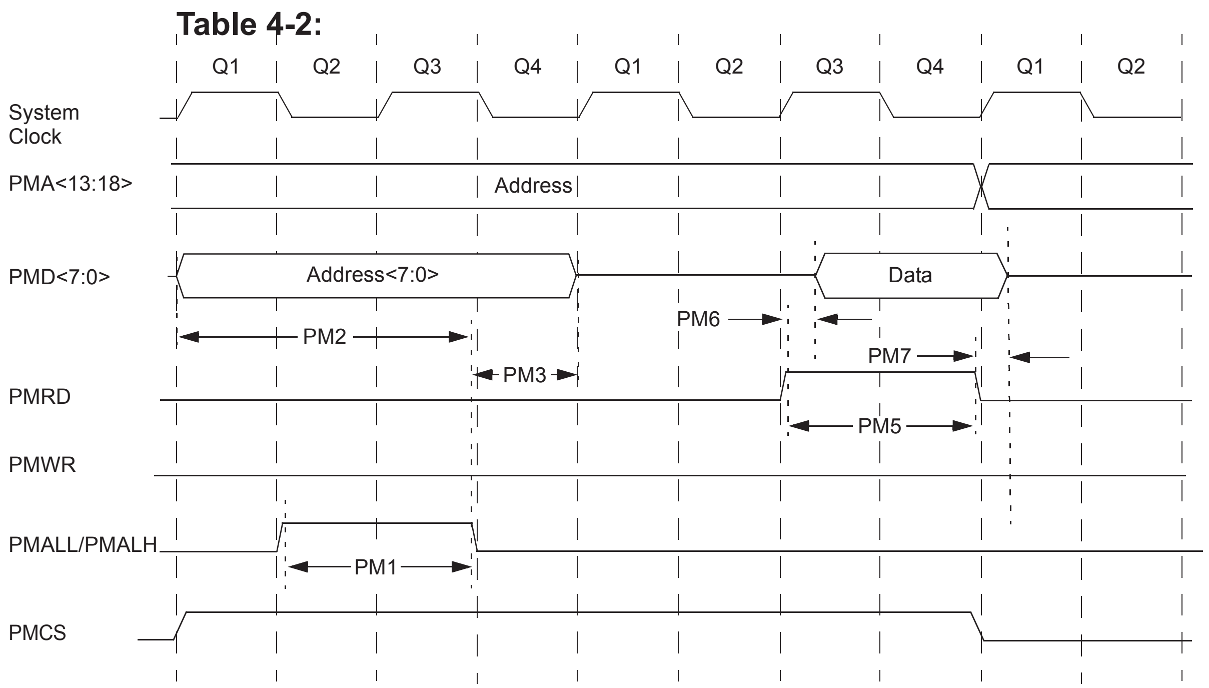


Table 4-2:
Q1
Q2
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
System Clock
PMA<13:18>
Address
PMD<7:0>
Address<7:0>
Data
PM6
PM2
PM7
PM3
PMRD
PM5
PMWR
PMALL/PMALH
PM1
PMCS

Operating Conditions: 2.0V < V_{DD} < 3.6V, -40°C < T_A < +85°C unless otherwise stated.

TABLE 30-18: PARALLEL MASTER PORT READ TIMING REQUIREMENTS

Param. No	Symbol	Characteristics	Min	Typ	Max	Units
PM1		PMALL/PMALH Pulse Width	—	0.5 T_{CY}	—	ns
PM2		Address Out Valid to PMALL/PMALH Invalid (address setup time)	—	0.75 T_{CY}	—	ns
PM3		PMALL/PMALH Invalid to Address Out Invalid (address hold time)	—	0.25 T_{CY}	—	ns
PM5		PMRD Pulse Width	—	0.5 T_{CY}	—	ns
PM6		PMRD or PMENB Active to Data In Valid (data setup time)	—	—	—	ns
PM7		PMRD or PMENB Inactive to Data In Invalid (data hold time)	—	—	—	ns

FIGURE 30-11: PARALLEL MASTER PORT WRITE TIMING DIAGRAM

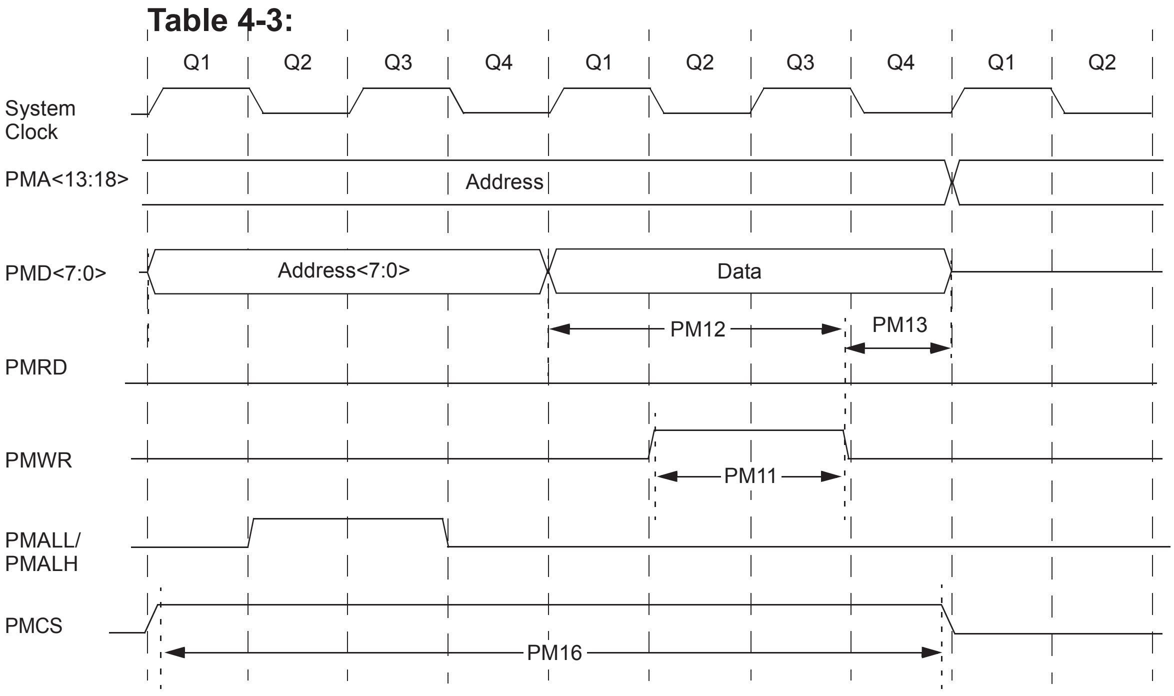


Table 4-3:
Q1
Q2
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
System Clock
PMA<13:18>
Address
PMD<7:0>
Address<7:0>
Data
PM12
PM13
PMRD
PMWR
PM11
PMALL/ PMALH
PMCS
PM16

Note: Operating Conditions: 2.0V < VDD < 3.6V, -40°C < TA < +85°C unless otherwise stated.

TABLE 30-19: PARALLEL MASTER PORT WRITE TIMING REQUIREMENTS

Param. No	Symbol	Characteristics	Min	Typ	Max	Units
PM11		PMWR Pulse Width	—	0.5 TCY	—	ns
PM12		Data Out Valid before PMWR or PMENB goes Inactive (data setup time)	—	—	—	ns
PM13		PMWR or PMEMB Invalid to Data Out Invalid (data hold time)	—	—	—	ns
PM16		PMCS Pulse Width	TCY – 5	—	—	ns

FIGURE 30-12: PARALLEL SLAVE PORT TIMING

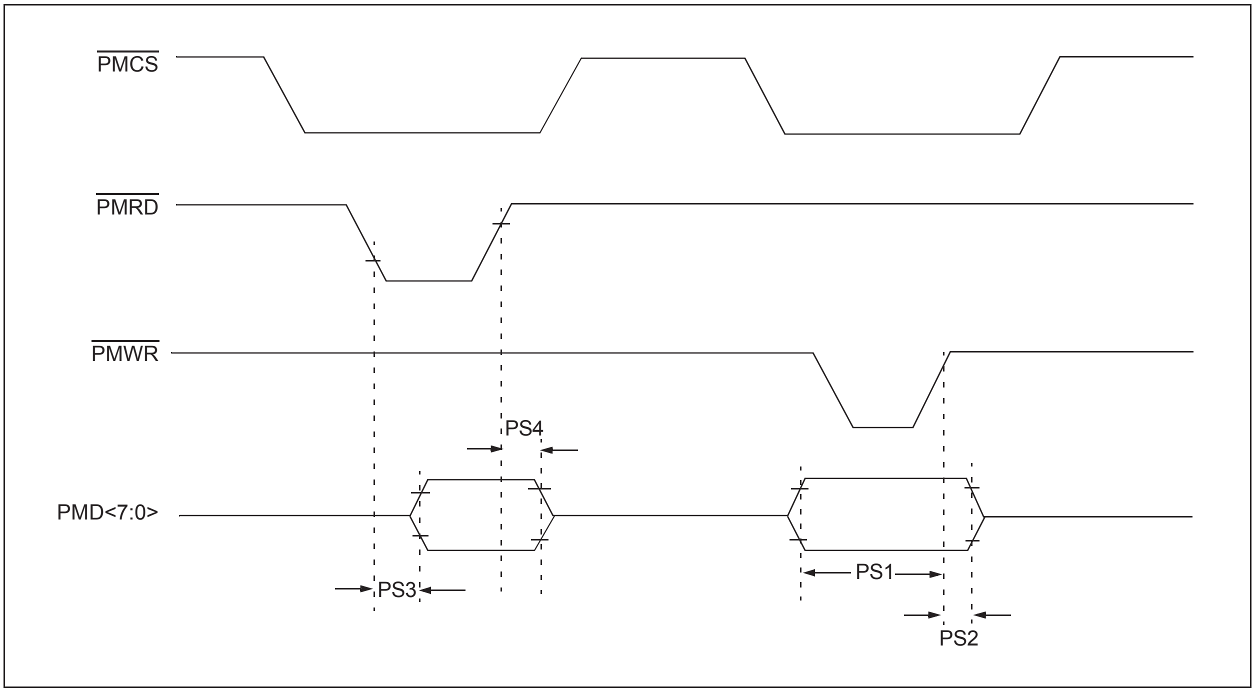
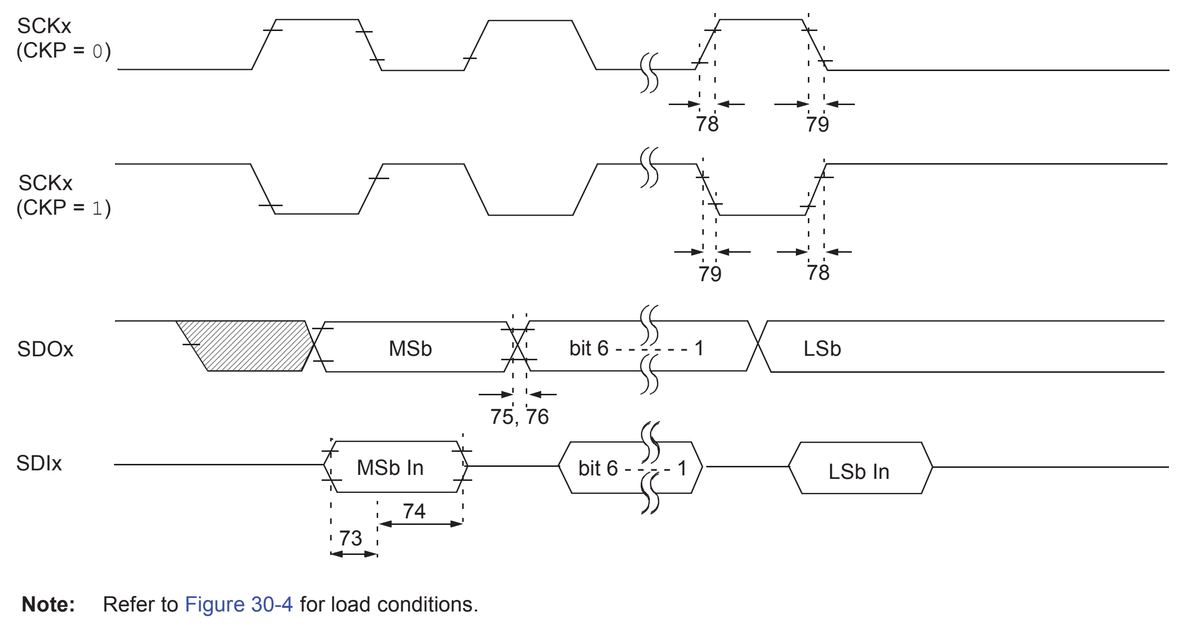


TABLE 30-20: PARALLEL SLAVE PORT REQUIREMENTS

AC CHARACTERISTICS			**Standard Operating Conditions: 2.0V to 3.6V (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for Industrial				
Param. No.	**Symbol**	**Characteristic**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
PS1	TdtV2wrH	Data In Valid before $\overline{\text{PMWR}}$ or $\overline{\text{PMCS}}$ Inactive (setup time)	20	—	—	ns	
PS2	TwrH2dtI	$\overline{\text{PMWR}}$ or $\overline{\text{PMCS}}$ Inactive to Data–In Invalid (hold time)	20	—	—	ns	
PS3	TrdL2dtV	$\overline{\text{PMRD}}$ and $\overline{\text{PMCS}}$ Active to Data–Out Valid	—	—	80	ns	
PS4	TrdH2dtI	$\overline{\text{PMRD}}$ Inactive or $\overline{\text{PMCS}}$ Inactive to Data–Out Invalid	10	—	30	ns	

FIGURE 30-13: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)

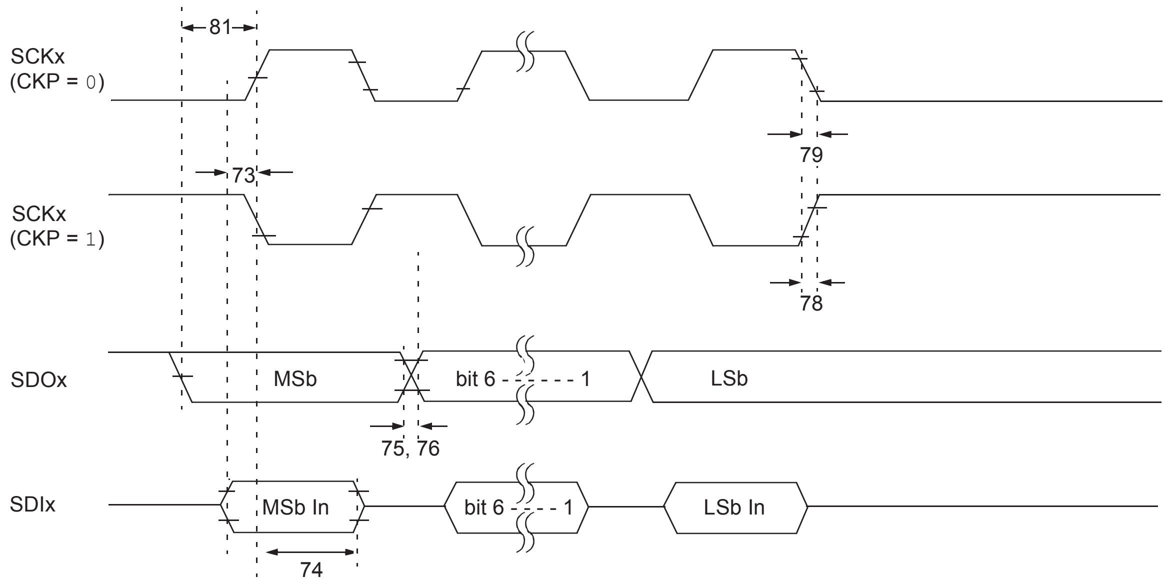


Note: Refer to Figure 30-4 for load conditions.

TABLE 30-21: 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	35	—	ns	VDD = 3.3V, VDDCORE = 2.5V
			100	—	ns	VDD = 2.15V, VDDCORE = 2.15V
74	TSCH2DIL, TSCL2DIL	Hold Time of SDIx Data Input to SCKx Edge	30	—	ns	VDD = 3.3V, VDDCORE = 2.5V
			83	—	ns	VDD = 2.15V
75	TDOR	SDOx Data Output Rise Time	—	25	ns	PORTB or PORTC
76	TDOF	SDOx Data Output Fall Time	—	25	ns	PORTB or PORTC
78	TSCR	SCKx Output Rise Time (Master mode)	—	25	ns	PORTB or PORTC
79	TSCF	SCKx Output Fall Time (Master mode)	—	25	ns	PORTB or PORTC

FIGURE 30-14: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)

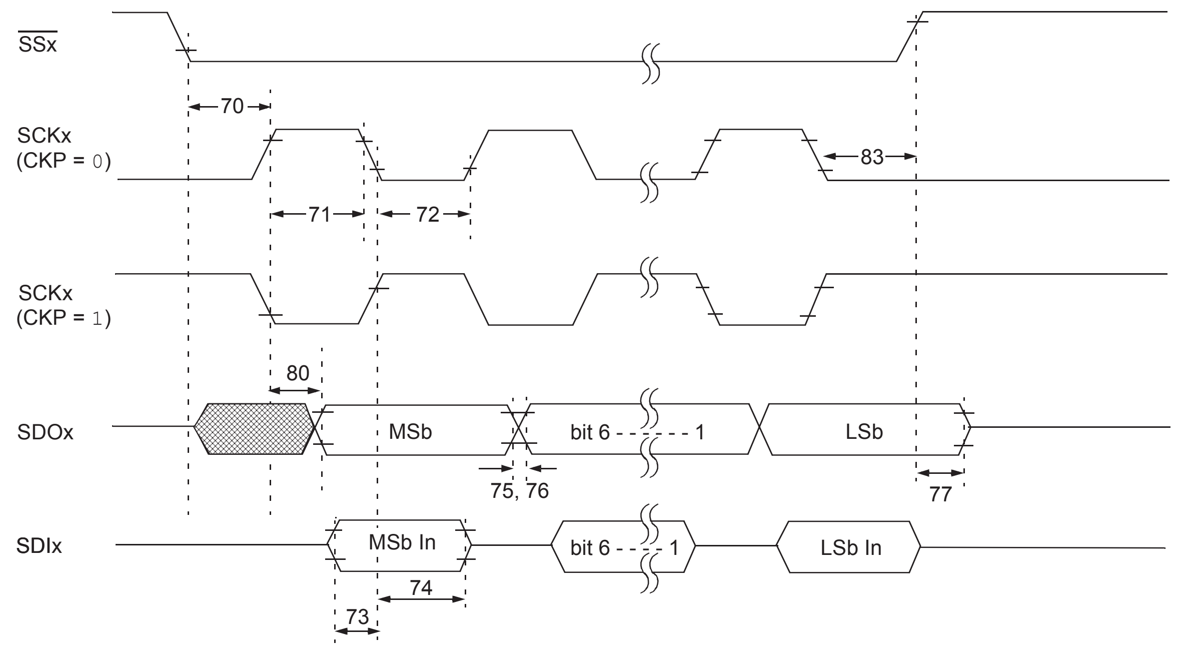


Note: Refer to Figure 30-4 for load conditions.

TABLE 30-22: 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	35	—	ns	VDD = 3.3V, VDDCORE = 2.5V
			100	—	ns	VDD = 2.15V, VDDCORE = 2.15V
74	TSCH2DIL, TSCL2DIL	Hold Time of SDIx Data Input to SCKx Edge	30	—	ns	VDD = 3.3V, VDDCORE = 2.5V
			83	—	ns	VDD = 2.15V
75	TDOR	SDOx Data Output Rise Time	—	25	ns	PORTB or PORTC
76	TDOF	SDOx Data Output Fall Time	—	25	ns	PORTB or PORTC
78	TSCR	SCKx Output Rise Time (Master mode)	—	25	ns	PORTB or PORTC
79	TSCF	SCKx Output Fall Time (Master mode)	—	25	ns	PORTB or PORTC
81	TDOV2SCH, TDOV2SCL	SDOx Data Output Setup to SCKx Edge	TCY	—	ns	

FIGURE 30-15: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)



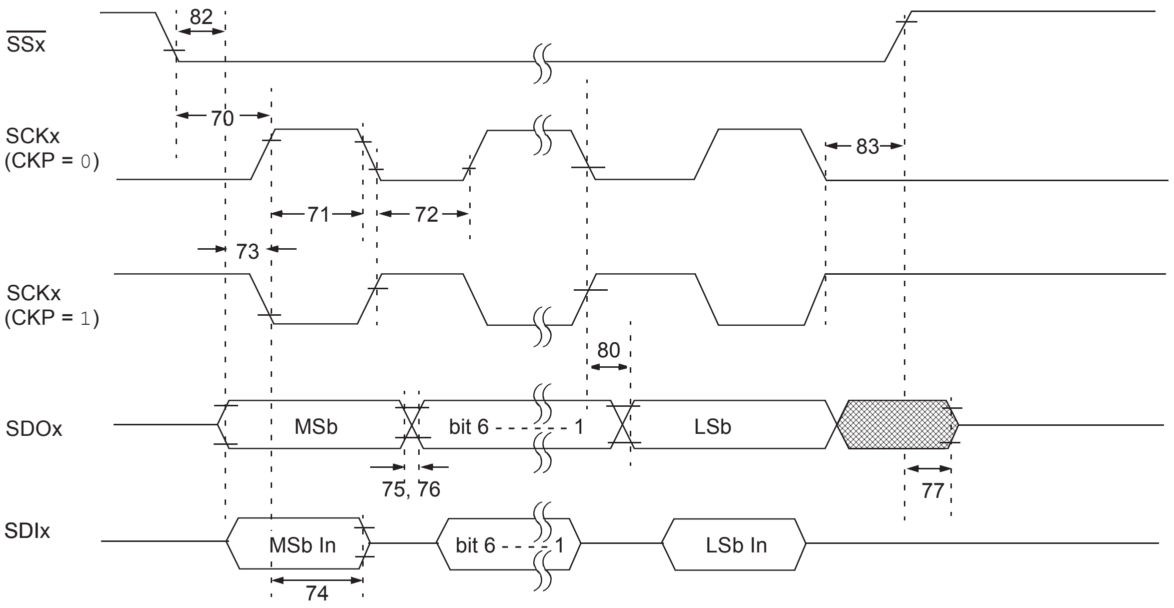
Note: Refer to Figure 30-4 for load conditions.

TABLE 30-23: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{\text{SSx}}$ ↓ to SCKx ↓ or SCKx ↑ Input		3 Tcy	—	ns	
70A	TssL2WB	$\overline{\text{SSx}}$ ↓ to Write to SSPxBUF		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		25	—	ns	
73A	Tb2b	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2		1.5 Tcy + 40	—	ns	**(Note 2)**
74	TscH2diL, TscL2diL	Hold Time of SDIx Data Input to SCKx Edge		35	—	ns	VDD = 3.3V, VDDCORE = 2.5V
				100	—	ns	VDD = 2.15V
75	TdoR	SDOx Data Output Rise Time		—	25	ns	PORTB or PORTC
76	TdoF	SDOx Data Output Fall Time		—	25	ns	PORTB or PORTC
77	TssH2doZ	$\overline{\text{SSx}}$ ↑ to SDOx Output High-Impedance		10	70	ns	
80	TscH2doV, TscL2doV	SDOx Data Output Valid after SCKx Edge		—	50	ns	VDD = 3.3V, VDDCORE = 2.5V
				—	100	ns	VDD = 2.15V
83	TscH2ssH, TscL2ssH	$\overline{\text{SSx}}$ ↑ 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 30-16: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)



Note: Refer to Figure 30-4 for load conditions.

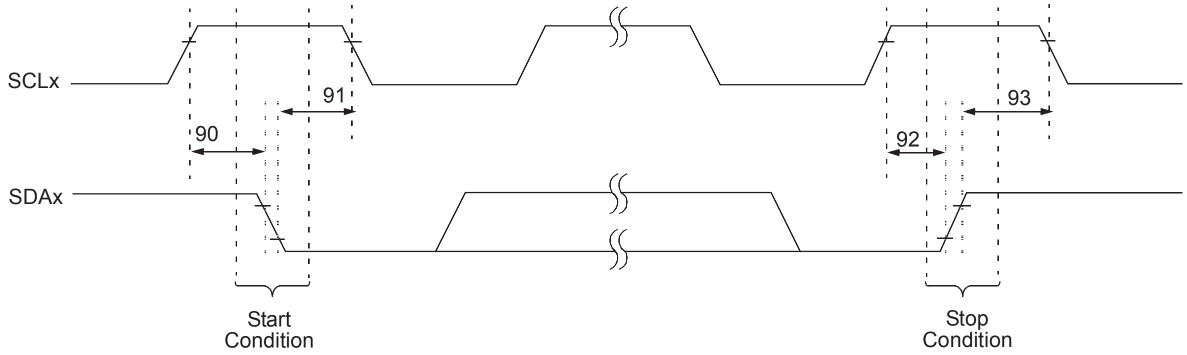
TABLE 30-24: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)

Param No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TSSL2SCH, TSSL2SCL	$\overline{\text{SSx}}$ ↓ to SCKx ↓ or SCKx ↑ Input		3 TCY	—	ns	
70A	TSSL2WB	$\overline{\text{SSx}}$ ↓ to Write to SSPxBUF		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		25	—	ns	
73A	TB2B	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2		1.5 TCY + 40	—	ns	**(Note 2)**
74	TSCH2DIL, TSCL2DIL	Hold Time of SDIx Data Input to SCKx Edge		35	—	ns	VDD = 3.3V, VDDCORE = 2.5V
				100	—	ns	VDD = 2.15V
75	TDOR	SDOx Data Output Rise Time		—	25	ns	
76	TDOF	SDOx Data Output Fall Time		—	25	ns	
77	TSSH2DOZ	$\overline{\text{SSx}}$ ↑ to SDOx Output High-Impedance		10	70	ns	
80	TSCH2DOV, TSCL2DOV	SDOx Data Output Valid after SCKx Edge		—	50	ns	VDD = 3.3V, VDDCORE = 2.5V
				—	100	ns	VDD = 2.15V
81	TDOV2SCH, TDOV2SCL	SDOx Data Output Setup to SCKx Edge		TCY	—	ns	
82	TSSL2DOV	SDOx Data Output Valid after $\overline{\text{SSx}}$ ↓ Edge		—	50	ns	
83	TSCH2SSH, TSCL2SSH	$\overline{\text{SSx}}$ ↑ 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 30-17: I²C BUS START/STOP BITS TIMING

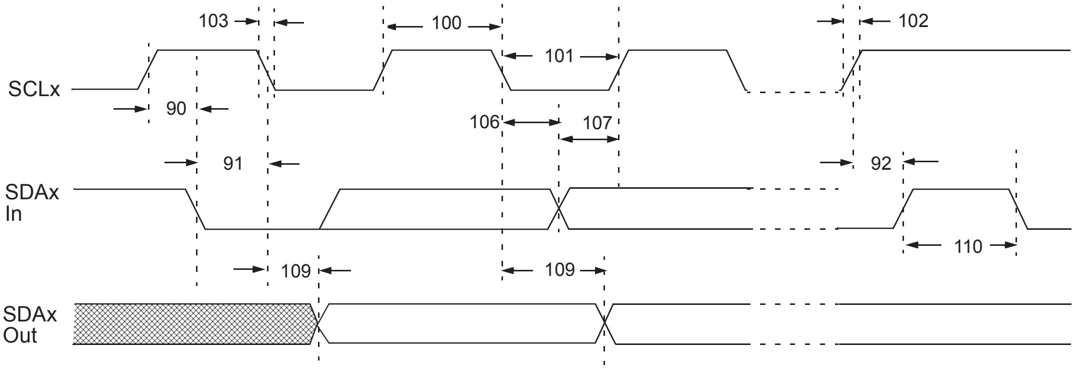

SCLx
SDAx
90
91
92
93
Start Condition
Stop Condition

Note: Refer to Figure 30-4 for load conditions.

TABLE 30-25: I²C BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start Condition	100 kHz mode	4700	—	ns	Only relevant for Repeated Start condition
		Setup Time	400 kHz mode	600	—		
91	THD:STA	Start Condition	100 kHz mode	4000	—	ns	After this period, the first clock pulse is generated
		Hold Time	400 kHz mode	600	—		
92	TSU:STO	Stop Condition	100 kHz mode	4700	—	ns	
		Setup Time	400 kHz mode	600	—		
93	THD:STO	Stop Condition	100 kHz mode	4000	—	ns	
		Hold Time	400 kHz mode	600	—		

FIGURE 30-18: I²C BUS DATA TIMING


103
100
102
101
SCLx
90
106
107
91
92
SDAx
In
110
109
109
SDAx
Out

Note: Refer to Figure 30-4 for load conditions.

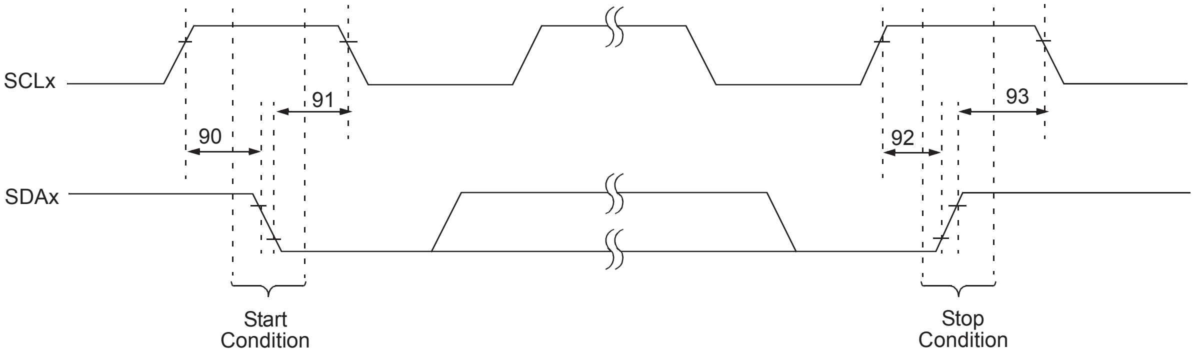
TABLE 30-26: I^{2}C BUS DATA REQUIREMENTS (SLAVE MODE)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	4.0	—	μs	
			400 kHz mode	0.6	—	μs	
			MSSP modules	1.5 TCY	—		
101	TLOW	Clock Low Time	100 kHz mode	4.7	—	μs	
			400 kHz mode	1.3	—	μs	
			MSSP modules	1.5 TCY	—		
102	TR	SDAx and SCLx Rise Time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	CB is specified to be from 10 to 400 pF
103	TF	SDAx and SCLx Fall Time	100 kHz mode	—	300	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	CB is specified to be from 10 to 400 pF
90	TSU:STA	Start Condition Setup Time	100 kHz mode	4.7	—	μs	Only relevant for Repeated Start condition
			400 kHz mode	0.6	—	μs	
91	THD:STA	Start Condition Hold Time	100 kHz mode	4.0	—	μs	After this period, the first clock pulse is generated
			400 kHz mode	0.6	—	μs	
106	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	μs	
107	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	4.7	—	μs	
			400 kHz mode	0.6	—	μs	
109	TAA	Output Valid from Clock	100 kHz mode	—	3500	ns	**(Note 1)**
			400 kHz mode	—	—	ns	
110	TBUF	Bus Free Time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	μs	
D102	CB	Bus Capacitive Loading		—	400	pF	

Note 1: As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCLx to avoid unintended generation of Start or Stop conditions.

2: A Fast mode I^{2}C bus device can be used in a Standard mode I^{2}C 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 I^{2}C bus specification), before the SCLx line is released.

FIGURE 30-19: MSSPx I^{2}C BUS START/STOP BITS TIMING WAVEFORMS

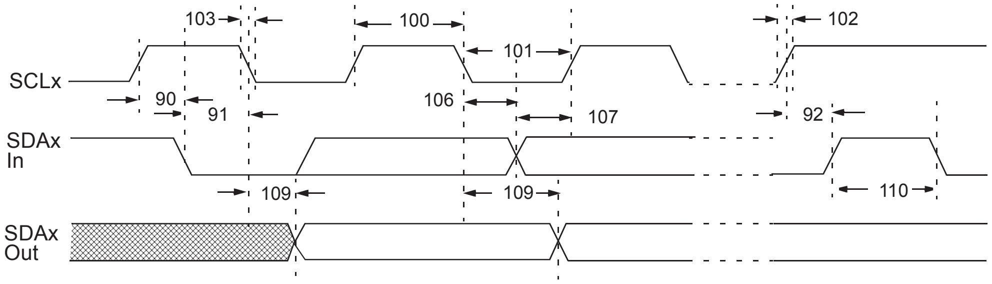

SCLx
SDAx
90
91
92
93
Start Condition
Stop Condition

Note: Refer to Figure 30-4 for load conditions.

TABLE 30-27: MSSPx I^{2}C BUS START/STOP BITS REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start Condition	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	Only relevant for Repeated Start condition
		Setup Time	400 kHz mode	2(TOSC)(BRG + 1)	—		
91	THD:STA	Start Condition	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	After this period, the first clock pulse is generated
		Hold Time	400 kHz mode	2(TOSC)(BRG + 1)	—		
92	TSU:STO	Stop Condition	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	
		Setup Time	400 kHz mode	2(TOSC)(BRG + 1)	—		
93	THD:STO	Stop Condition	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	
		Hold Time	400 kHz mode	2(TOSC)(BRG + 1)	—		

FIGURE 30-20: MSSPx I^{2}C BUS DATA TIMING


103
100
102
101
SCLx
90
91
106
107
92
SDAx
In
109
109
110
SDAx
Out

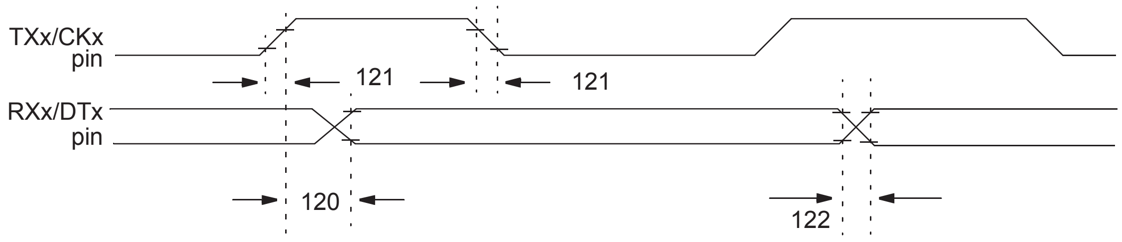
Note: Refer to Figure 30-4 for load conditions.

TABLE 30-28: MSSPx I^{2}C BUS DATA REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	2(TOSC)(BRG + 1)	—	μs	
			400 kHz mode	2(TOSC)(BRG + 1)	—	μs	
101	TLOW	Clock Low Time	100 kHz mode	2(TOSC)(BRG + 1)	—	μs	
			400 kHz mode	2(TOSC)(BRG + 1)	—	μs	
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	
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	
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	μs	Only relevant for Repeated Start condition
			400 kHz mode	2(TOSC)(BRG + 1)	—	μs	
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(TOSC)(BRG + 1)	—	μs	After this period, the first clock pulse is generated
			400 kHz mode	2(TOSC)(BRG + 1)	—	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 1)**
			400 kHz mode	100	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	μs	
			400 kHz mode	2(TOSC)(BRG + 1)	—	μs	
109	TAA	Output Valid from Clock	100 kHz mode	—	3450	ns	
			400 kHz mode	—	900	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: A Fast mode I^{2}C bus device can be used in a Standard mode I^{2}C bus system, but parameter #107 ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCLx signal. If such a device does stretch the LOW period of the SCLx signal, it must output the next data bit to the SDAx line, parameter #102 + parameter #107 = 1000 + 250 = 1250 ns (for 100 kHz mode), before the SCLx line is released.

FIGURE 30-21: EUSARTx SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

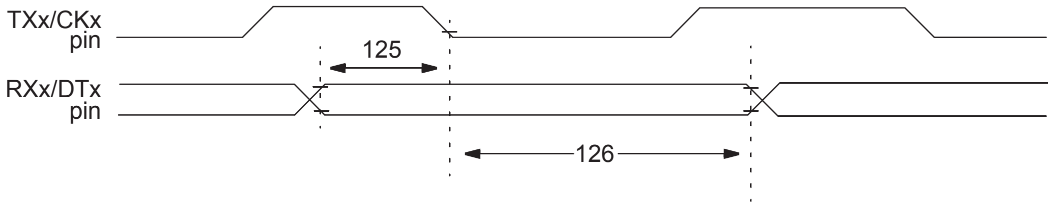


Note: Refer to Figure 30-4 for load conditions.

TABLE 30-29: EUSARTx SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
120	TCKH2DTV	Sync XMIT (Master and Slave) 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 30-22: EUSARTx SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 30-4 for load conditions.

TABLE 30-30: EUSARTx SYNCHRONOUS RECEIVE REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TDTV2CKL	Sync RCV (Master and Slave) Data Hold before CKx ↓ (DTx hold time)	10	—	ns	
126	TCKL2DTL	Data Hold after CKx ↓ (DTx hold time)	15	—	ns	

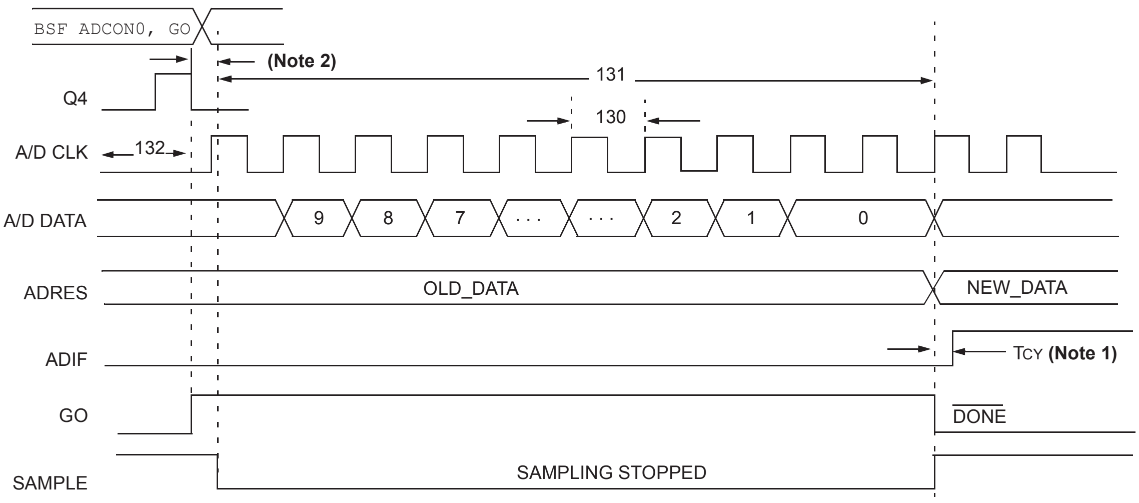
TABLE 30-31: A/D CONVERTER CHARACTERISTICS: PIC18F47J13 FAMILY (INDUSTRIAL)

Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
A01	NR	Resolution	—	—	12	bit	ΔVREF ≥ 3.0V
A03	EIL	Integral Linearity Error	—	<±1	±2	LSb	ΔVREF ≥ 3.0V
A04	EDL	Differential Linearity Error	—	<±1	1.5	LSb	ΔVREF ≥ 3.0V
A06	EOFF	Offset Error	—	<±1	5	LSb	ΔVREF ≥ 3.0V
A07	EGN	Gain Error	—	—	<±3.5	LSb	ΔVREF ≥ 3.0V
A10		Monotonicity	Guaranteed(1)			—	VSS ≤ VAIN ≤ VREF
A20	ΔVREF	Reference Voltage Range (VREFH – VREFL)	2.0 3	— —	— —	V V	VDD < 3.0V VDD ≥ 3.0V
A21	VREFH	Reference Voltage High For 10-bit resolution For 12-bit resolution	 VREFL VSS + 3V	 — —	 VDD + 0.3V VDD + 0.3V	 V V	
A22	VREFL	Reference Voltage Low For 10-bit resolution For 12-bit resolution	 VSS – 0.3V VSS – 0.3V	 — —	 VREFH VDD - 3V	 V V	
A25	VAIN	Analog Input Voltage	VREFL	—	VREFH	V	
A30	ZAIN	Recommended Impedance of Analog Voltage Source For 10-bit resolution For 12-bit resolution	 — —	 — —	 2.5 1	 kΩ kΩ	
A50	IREF	VREF Input Current(2)	— —	— —	5 150	μA μA	During VAIN acquisition. During A/D conversion cycle.

Note 1: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

2: VREFH current is from RA3/AN3/C1INBVREF+ pin or VDD, whichever is selected as the VREFH source. VREFL current is from RA2/AN2/C2INB/C1IND/C3INB/VREF-/CVREF pin or VSS, whichever is selected as the VREFL source.

FIGURE 30-23: A/D CONVERSION TIMING



Note 1: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the `SLEEP` instruction to be executed.

2: This is a minimal RC delay (typically 100 ns), which also disconnects the holding capacitor from the analog input.

TABLE 30-32: 10-BIT A/D CONVERSION REQUIREMENTS

Param No.	Symbol	Characteristic	Min	Max	Units	Conditions
130	TAD	A/D Clock Period	0.7	25.0[1]	μs	TOSC based, VREF ≥ 3.0V
131	TCNV	Conversion Time (not including acquisition time)[2]	11	12	TAD	
132	TACQ	Acquisition Time[3]	1.4	—	μs	-40°C to +85°C
135	TSWC	Switching Time from Convert → Sample	—	**(Note 4)**		
137	TDIS	Discharge Time	0.2	—	μs	

Note 1: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

2: 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 50W.

4: On the following cycle of the device clock.

TABLE 30-33: 12-BIT 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
131	TCNV	Conversion Time (not including acquisition time)[2]	13	14	TAD	
132	TACQ	Acquisition Time[3]	1.4	—	μs	
135	TSWC	Switching Time from Convert → Sample	—	**(Note 4)**		
137	TDIS	Discharge Time	0.2	—	μs	

Note 1: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

2: ADRES registers may be read on the following TCY cycle.

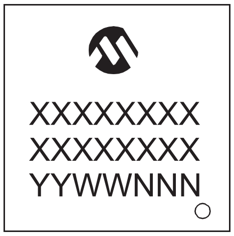
3: The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (VDD to VSS or VSS to VDD). The source impedance (RS) on the input channels is 50Ω.

4: On the following cycle of the device clock.

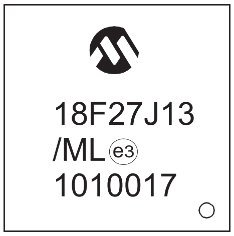
31.0 PACKAGING INFORMATION

31.1 Package Marking Information

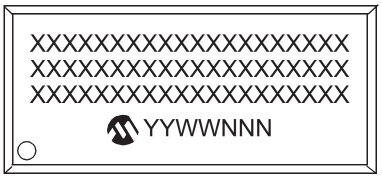
28-Lead QFN


XXXXXXXX
XXXXXXXX
YYWWNNN

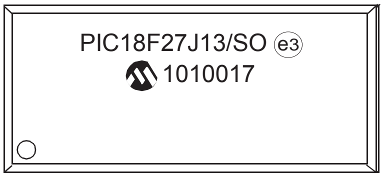
Example


18F27J13
/ML e3
1010017

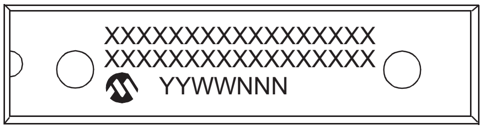
28-Lead SOIC (.300”)


XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
YYWWNNN

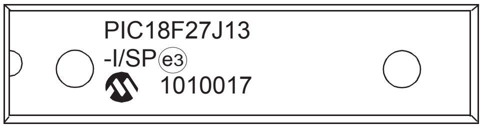
Example


PIC18F27J13/SO e3
1010017

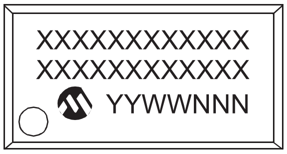
28-Lead SPDIP


XXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXX
YYWWNNN

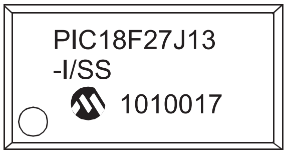
Example


PIC18F27J13
-I/SP e3
1010017

28-Lead SSOP


XXXXXXXXXXXX
XXXXXXXXXXXX
YYWWNNN

Example

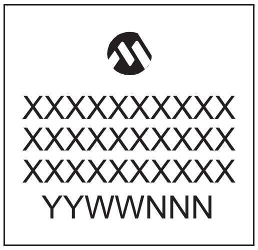

PIC18F27J13
-I/SS
1010017

Legend:		
	XX...X	Customer-specific information
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
		Pb-free JEDEC designator for Matte Tin (Sn)
	*	This package is Pb-free. The Pb-free JEDEC designator (e3) can be found on the outer packaging for this package.

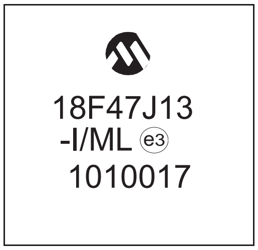
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

Package Marking Information (Continued)

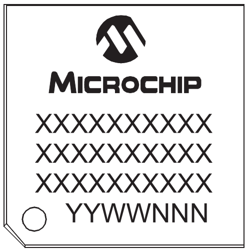
44-Lead QFN


XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
YYWWNNN

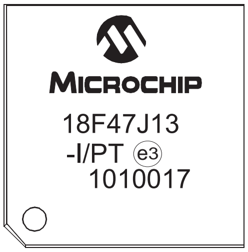
Example


18F47J13
-I/ML e3
1010017

44-Lead TQFP


MICROCHIP
XXXXXXXXXX
XXXXXXXXXX
XXXXXXXXXX
YYWWNNN

Example

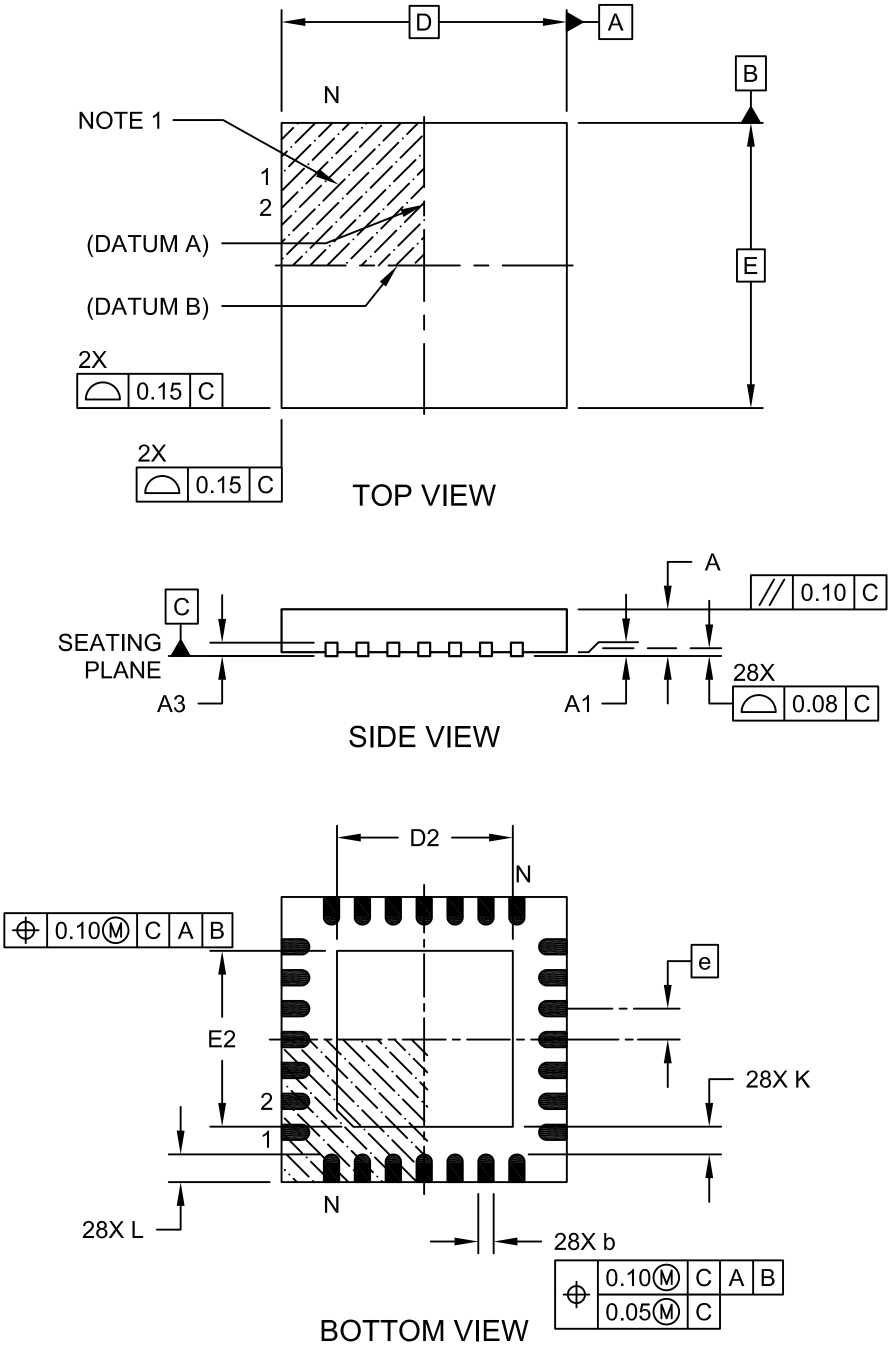

MICROCHIP
18F47J13
-I/PT e3
1010017

31.2 Package Details

The following sections give the technical details of the packages.

28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN] With 0.55 mm Terminal Length

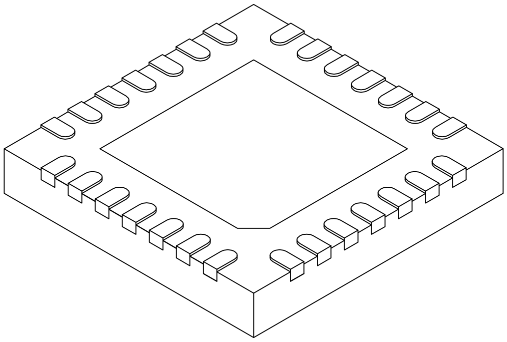
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging


D
A
B
N
NOTE 1
1
2
(DATUM A)
(DATUM B)
E
2X
0.15 C
2X
0.15 C
TOP VIEW
A
0.10 C
C
SEATING PLANE
A3
A1
28X
0.08 C
SIDE VIEW
D2
N
0.10Ⓜ C A B
e
E2
28X K
2
1
N
28X L
28X b
0.10Ⓜ C A B
0.05Ⓜ C
BOTTOM VIEW

Microchip Technology Drawing C04-105C Sheet 1 of 2

28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN] With 0.55 mm Terminal Length

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		28	
Pitch	e		0.65 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.20 REF	
Overall Width	E		6.00 BSC	
Exposed Pad Width	E2	3.65	3.70	4.20
Overall Length	D		6.00 BSC	
Exposed Pad Length	D2	3.65	3.70	4.20
Terminal Width	b	0.23	0.30	0.35
Terminal Length	L	0.50	0.55	0.70
Terminal-to-Exposed Pad	K	0.20	-	-

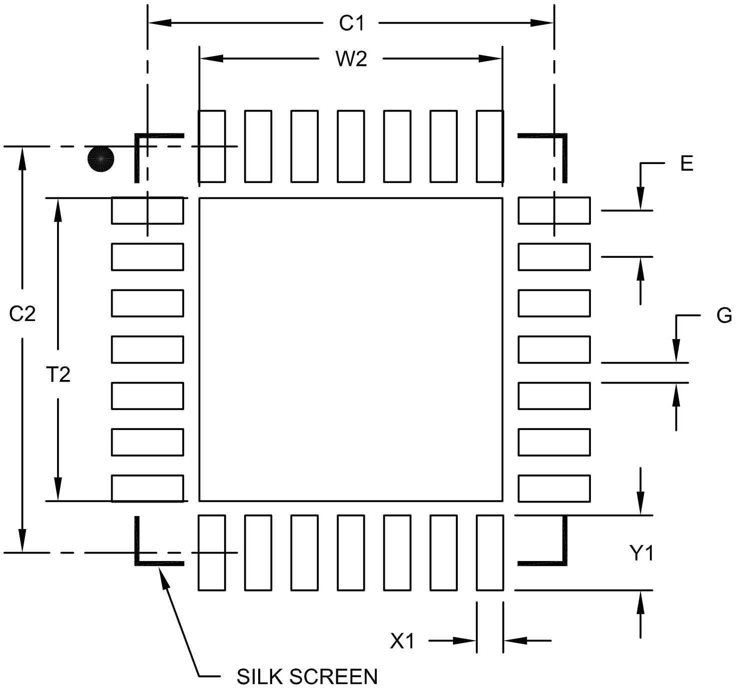
Notes:

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 C04-105C Sheet 2 of 2

28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Optional Center Pad Width	W2			4.25
Optional Center Pad Length	T2			4.25
Contact Pad Spacing	C1		5.70	
Contact Pad Spacing	C2		5.70	
Contact Pad Width (X28)	X1			0.37
Contact Pad Length (X28)	Y1			1.00
Distance Between Pads	G	0.20		

Notes:

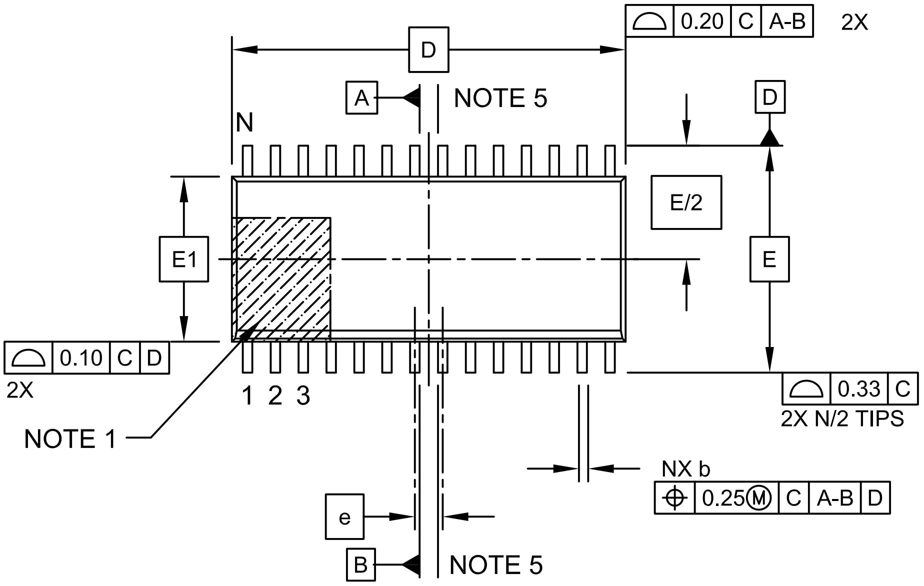
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

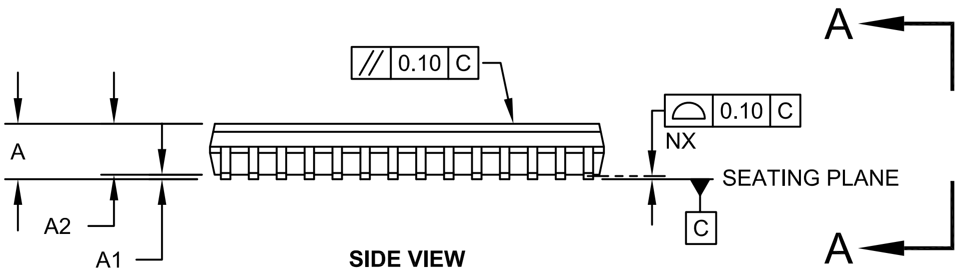
Microchip Technology Drawing No. C04-2105A

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

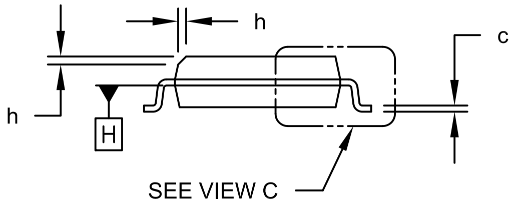
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging


0.20 C A-B
2X
D
A
NOTE 5
N
D
E/2
E1
E
0.10 C D
2X
1 2 3
NOTE 1
0.33 C
2X N/2 TIPS
NX b
0.25 M C A-B D
e
B
NOTE 5

TOP VIEW


A
0.10 C
0.10 C
NX
A
SEATING PLANE
C
A2
A1
A

SIDE VIEW

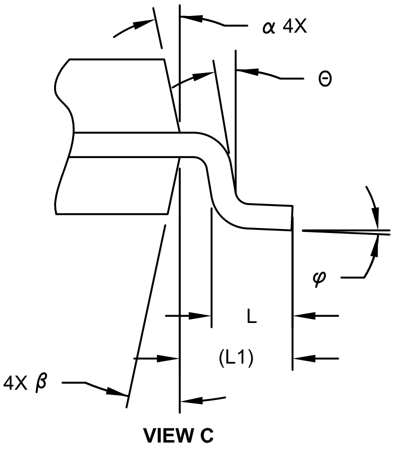

h
c
h
H
SEE VIEW C

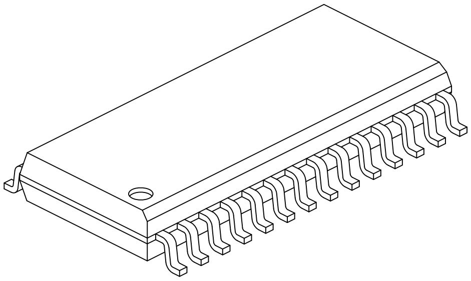
VIEW A-A

Microchip Technology Drawing C04-052C Sheet 1 of 2

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging





Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	28		
Pitch	e	1.27 BSC		
Overall Height	A	-	-	2.65
Molded Package Thickness	A2	2.05	-	-
Standoff §	A1	0.10	-	0.30
Overall Width	E	10.30 BSC		
Molded Package Width	E1	7.50 BSC		
Overall Length	D	17.90 BSC		
Chamfer (Optional)	h	0.25	-	0.75
Foot Length	L	0.40	-	1.27
Footprint	L1	1.40 REF		
Lead Angle	Θ	0°	-	-
Foot Angle	φ	0°	-	8°
Lead Thickness	c	0.18	-	0.33
Lead Width	b	0.31	-	0.51
Mold Draft Angle Top	α	5°	-	15°
Mold Draft Angle Bottom	β	5°	-	15°

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

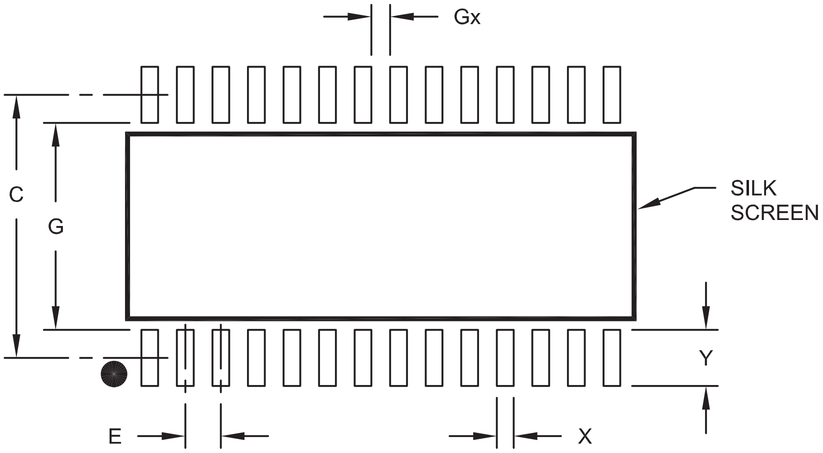
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing C04-052C Sheet 2 of 2

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

	Units	MILLIMETERS		
	Dimension Limits	MIN	NOM	MAX
Contact Pitch	E	1.27 BSC		
Contact Pad Spacing	C		9.40	
Contact Pad Width (X28)	X			0.60
Contact Pad Length (X28)	Y			2.00
Distance Between Pads	Gx	0.67		
Distance Between Pads	G	7.40		

Notes:

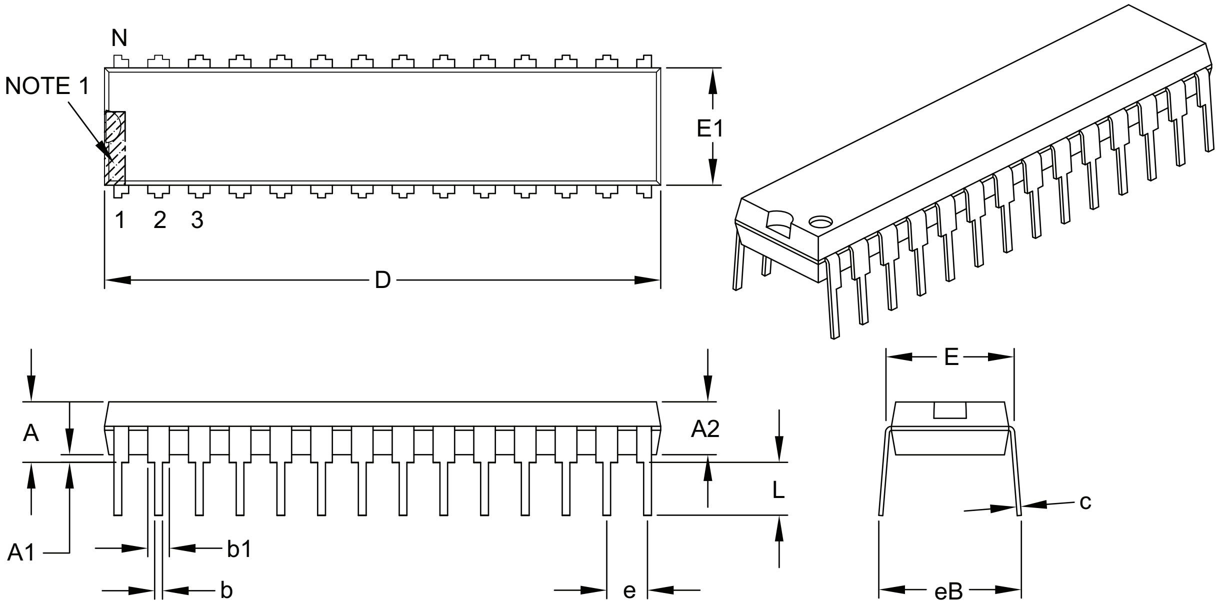
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2052A

28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N	28		
Pitch	e	.100 BSC		
Top to Seating Plane	A	–	–	.200
Molded Package Thickness	A2	.120	.135	.150
Base to Seating Plane	A1	.015	–	–
Shoulder to Shoulder Width	E	.290	.310	.335
Molded Package Width	E1	.240	.285	.295
Overall Length	D	1.345	1.365	1.400
Tip to Seating Plane	L	.110	.130	.150
Lead Thickness	c	.008	.010	.015
Upper Lead Width	b1	.040	.050	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	–	–	.430

Notes:

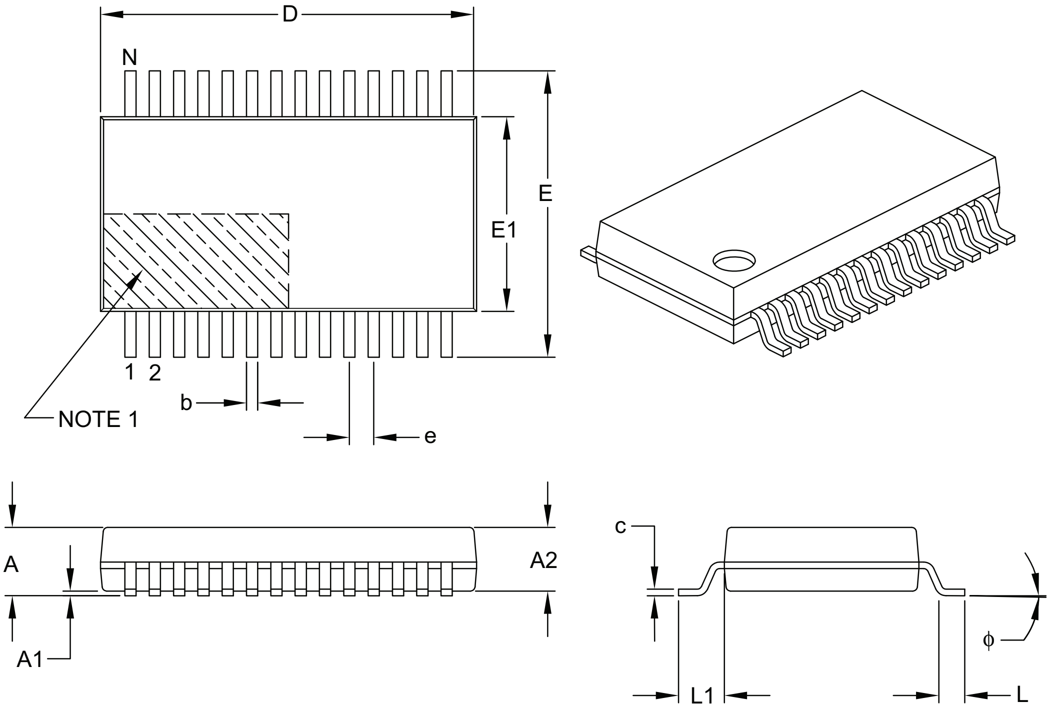
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-070B

28-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		0.65 BSC	
Overall Height	A	–	–	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	–	–
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	9.90	10.20	10.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1		1.25 REF	
Lead Thickness	c	0.09	–	0.25
Foot Angle	ϕ	0°	4°	8°
Lead Width	b	0.22	–	0.38

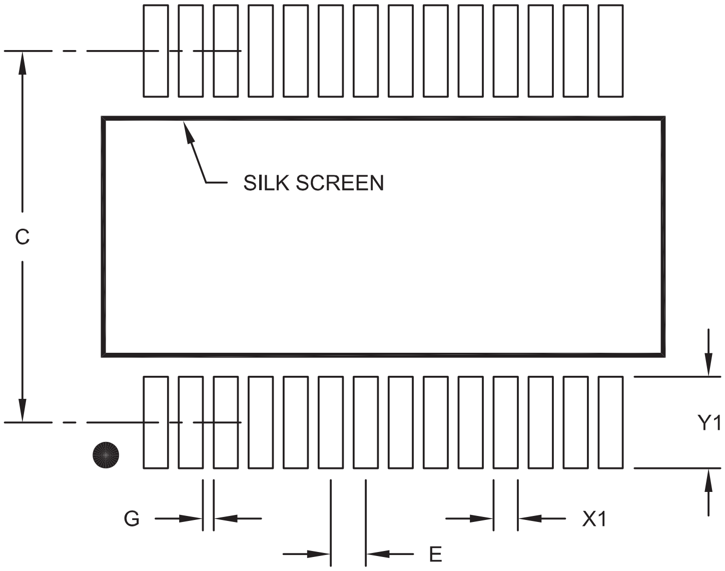
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073B

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

	Units	MILLIMETERS		
	Dimension Limits	MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Contact Pad Spacing	C		7.20	
Contact Pad Width (X28)	X1			0.45
Contact Pad Length (X28)	Y1			1.75
Distance Between Pads	G	0.20		

Notes:

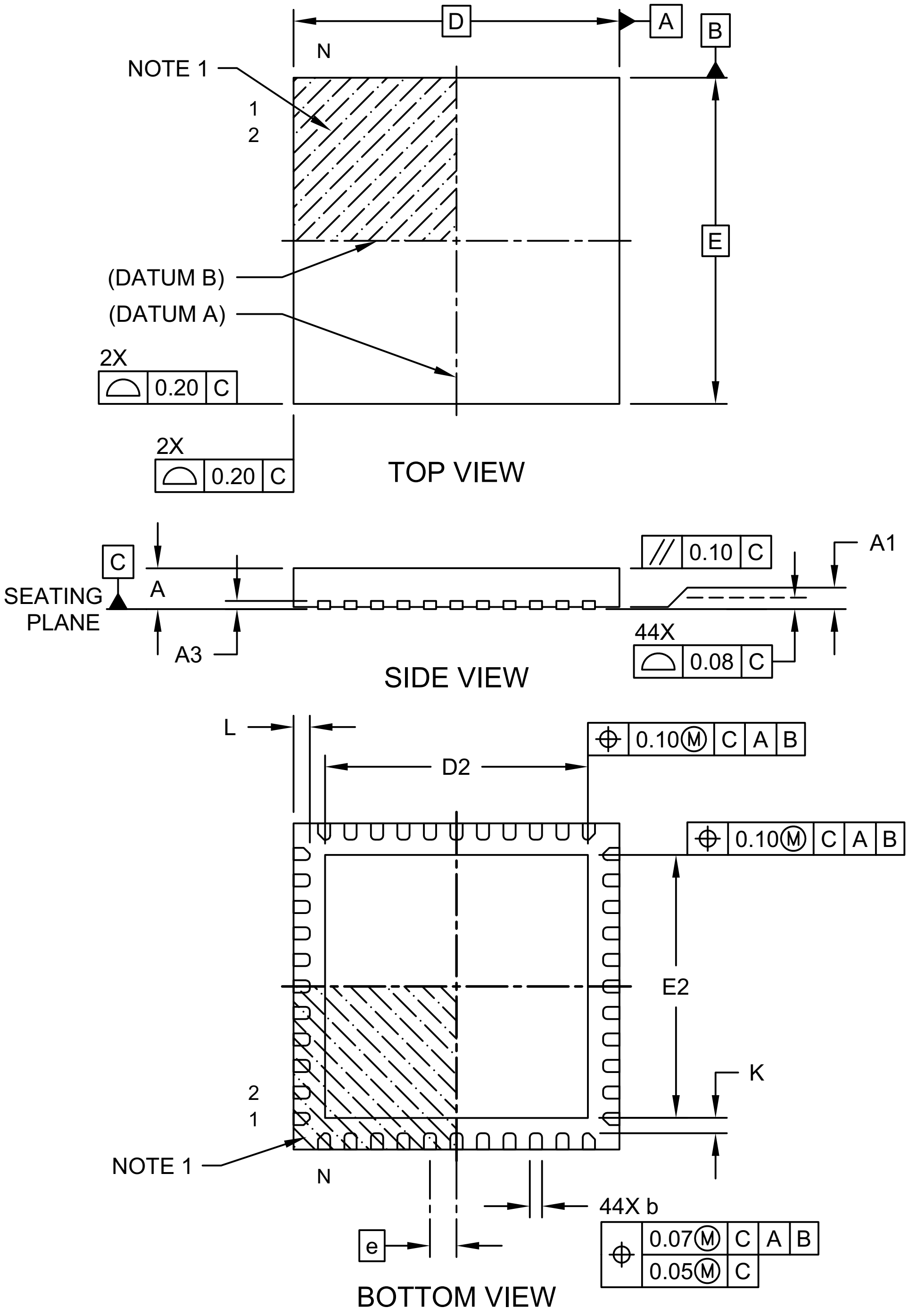
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

44-Lead Plastic Quad Flat, No Lead Package (ML) - 8x8 mm Body [QFN or VQFN]

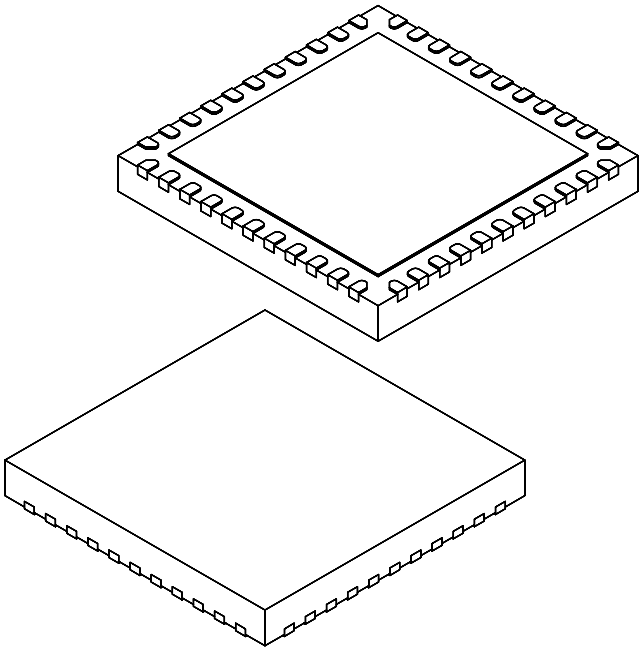
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging


D
A
B
N
NOTE 1
1
2
E
(DATUM B)
(DATUM A)
2X
0.20 C
2X
0.20 C
TOP VIEW
C
SEATING PLANE
A
A3
// 0.10 C
A1
44X
0.08 C
SIDE VIEW
L
0.10Ⓜ C A B
D2
0.10Ⓜ C A B
E2
K
2
1
NOTE 1
N
44X b
0.07Ⓜ C A B
0.05Ⓜ C
e
BOTTOM VIEW

Microchip Technology Drawing C04-103D Sheet 1 of 2

44-Lead Plastic Quad Flat, No Lead Package (ML) - 8x8 mm Body [QFN or VQFN]

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	44		
Pitch	e	0.65 BSC		
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3	0.20 REF		
Overall Width	E	8.00 BSC		
Exposed Pad Width	E2	6.25	6.45	6.60
Overall Length	D	8.00 BSC		
Exposed Pad Length	D2	6.25	6.45	6.60
Terminal Width	b	0.20	0.30	0.35
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K	0.20	-	-

Notes:

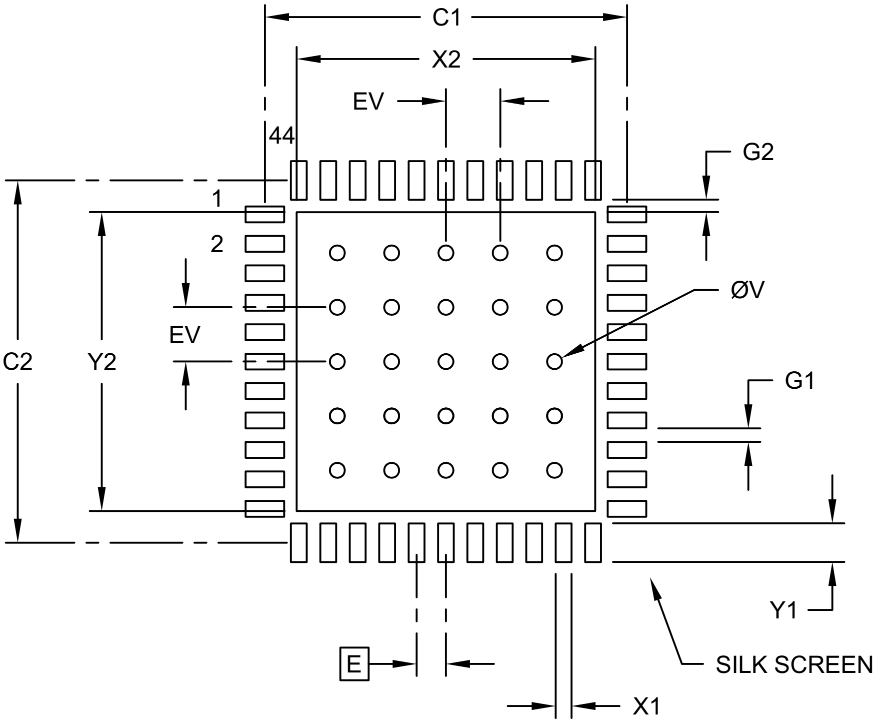
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-103D Sheet 2 of 2

44-Lead Plastic Quad Flat, No Lead Package (ML) - 8x8 mm Body [QFN or VQFN]

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	0.65 BSC		
Optional Center Pad Width	X2			6.60
Optional Center Pad Length	Y2			6.60
Contact Pad Spacing	C1		8.00	
Contact Pad Spacing	C2		8.00	
Contact Pad Width (X44)	X1			0.35
Contact Pad Length (X44)	Y1			0.85
Contact Pad to Contact Pad (X40)	G1	0.30		
Contact Pad to Center Pad (X44)	G2	0.28		
Thermal Via Diameter	V		0.33	
Thermal Via Pitch	EV		1.20	

Notes:

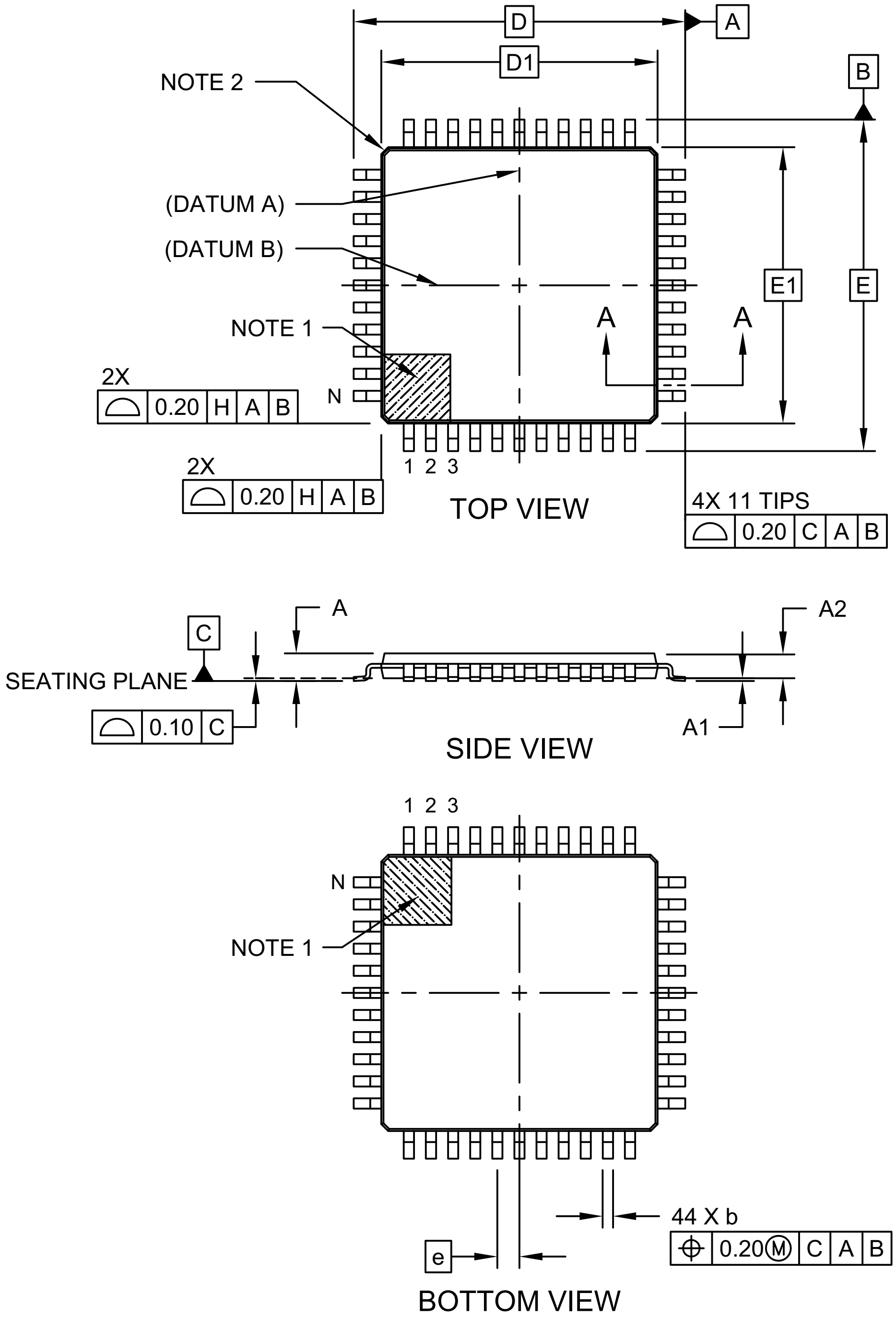
1. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing No. C04-2103C

44-Lead Plastic Thin Quad Flatpack (PT) - 10x10x1.0 mm Body [TQFP]

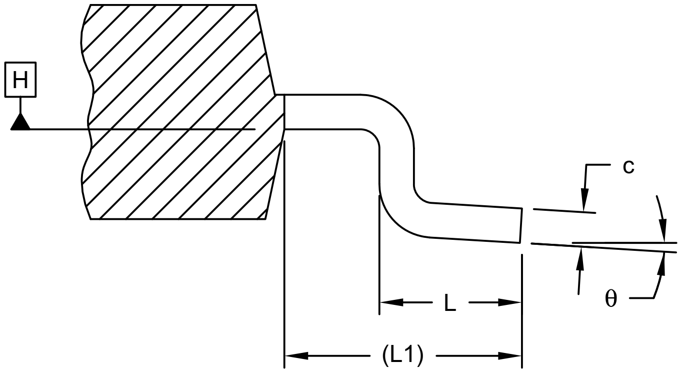
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging


D
A
D1
B
NOTE 2
(DATUM A)
(DATUM B)
E1
E
A
A
NOTE 1
2X
0.20 H A B
N
2X
0.20 H A B
1 2 3
TOP VIEW
4X 11 TIPS
0.20 C A B
A
A2
C
SEATING PLANE
0.10 C
A1
SIDE VIEW
1 2 3
N
NOTE 1
44 X b
0.20Ⓜ C A B
e
BOTTOM VIEW

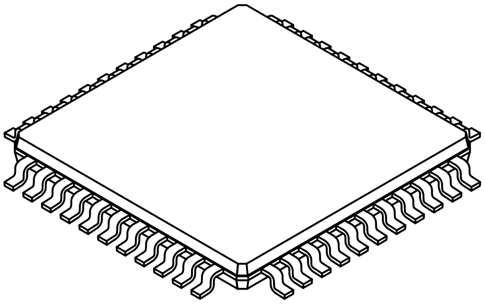
Microchip Technology Drawing C04-076C Sheet 1 of 2

44-Lead Plastic Thin Quad Flatpack (PT) - 10x10x1.0 mm Body [TQFP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



SECTION A-A



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Leads	N		44	
Lead Pitch	e		0.80 BSC	
Overall Height	A	-	-	1.20
Standoff	A1	0.05	-	0.15
Molded Package Thickness	A2	0.95	1.00	1.05
Overall Width	E		12.00 BSC	
Molded Package Width	E1		10.00 BSC	
Overall Length	D		12.00 BSC	
Molded Package Length	D1		10.00 BSC	
Lead Width	b	0.30	0.37	0.45
Lead Thickness	c	0.09	-	0.20
Lead Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Foot Angle	θ	0°	3.5°	7°

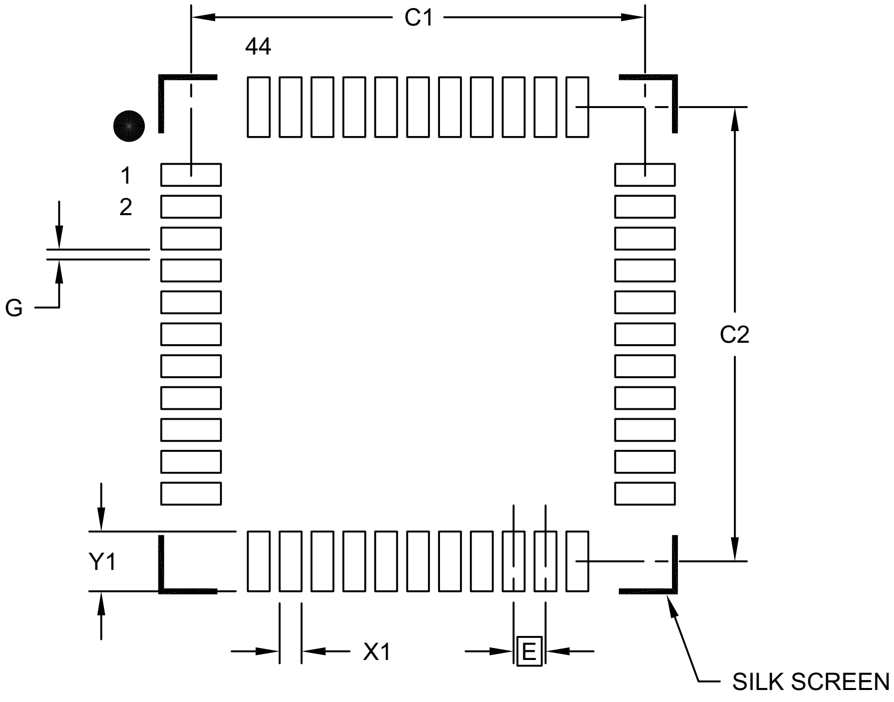
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Exact shape of each corner is optional.
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 C04-076C Sheet 2 of 2

44-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		0.80 BSC		
Contact Pad Spacing	C1			11.40	
Contact Pad Spacing	C2			11.40	
Contact Pad Width (X44)	X1				0.55
Contact Pad Length (X44)	Y1				1.50
Distance Between Pads	G		0.25		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2076B

APPENDIX A: REVISION HISTORY

Revision A (March 2010)

Original data sheet for PIC18F47J13 Family devices.

Revision B (9/2016)

Removed Preliminary from the data sheet; Updated Packages; Other minor corrections.

Revision C (3/2017)

Replaced ADC chapter with version from Revision A of the document; Minor changes to the entire document.

APPENDIX B: MIGRATION FROM PIC18F46J11 TO PIC18F47J13

Code for the devices in the PIC18F46J11 family can be migrated to the PIC18F47J13 without many changes. The differences between the two device families are listed in Table B-1.

TABLE B-1: NOTABLE DIFFERENCES BETWEEN PIC18F47J13 AND PIC18F46J11 FAMILIES

Characteristic	PIC18F47J13 Family	18F46J11 Family
Maximum Program Memory	128 Kbytes	64 Kbytes
Oscillator Options	PLL can be enabled at start-up with the Configuration bit option. 96 MHz PLL circuit is available via the Configuration bit setting, allowing 48 MHz operation from INTOSC. Default 4x PLL circuit is still available.	Requires firmware to set the PLLEN bit at run time. 4x PLL circuit only. Maximum operating frequency from INTOSC is 32 MHz.
SOSC Oscillator Options	Low-power oscillator option for SOSC, with run-time switch.	Low-power oscillator option for SOSC, only via the Configuration bit setting.
T1CKI Clock Input	T1CKI can be used as a clock input without enabling the Timer1 oscillator.	No
Timers	8	5
ECCP	3	2
CCP	7	0
SPI FOSC/8 Master Clock Option	Yes	No
Second I^{2}C Port	Yes, all packages.	Yes, but only on 44-pin devices.
ADC	13 Channel, 10/12-Bit Conversion modes with Special Event Trigger option.	13 Channel, 10-bit only.
Peripheral Module Disable Bits	Yes, allowing further power reduction.	No
Band Gap Voltage Reference Output	Yes, enabled on pin, RA1, by setting the VBGOE bit (WDTCON<4>).	No
$\overline{\text{REPU}}$/$\overline{\text{RDPU}}$ Pull-Up Enable Bits	Moved to TRISE register (avoids read, modify, write issues).	Pull-up bits configured in PORTE register
Comparators	Three, each with four input pin selections.	Two, each with two input pin selections.
Increased Output Drive Strength	RA0 through RA5, RDx and REx.	No
PWRT Period	F Devices – 500 μs LF Devices – 46 ms	F Devices – 1 ms LF Devices – 64 ms

THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at: http://microchip.com/support

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	[X](1)	-	X	/XX	XXX
Device	Tape and Reel Option		Temperature Range	Package	Pattern

Device(3,4): PIC18F26J13, PIC18F27J13, PIC18F46J13 and PIC18F47J13
VDD range 2.15V to 3.6V
PIC18LF26J13, PIC18LF27J13, PIC18LF46J13 and PIC18LF47J13
VDD range 2.0V to 3.6V

Temperature Range:
I = -40°C to +85°C (Industrial)
E = -40°C to +125°C (Extended)

Package(2):
ML = QFN
PT = TQFP (Thin Quad Flatpack)
SO = SOIC
SP = Skinny Plastic DIP
SS = SSOP

Pattern: QTP, SQTP, Code or Special Requirements (blank otherwise)

Examples:

a) PIC18F47J13-I/PT 301 = Industrial temp., TQFP package, QTP pattern #301.
b) PIC18F47J13T-I/PT = Tape and reel, Industrial temp., 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 not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.

2: Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

3: F = Standard Voltage Range with internal 2.5V core voltage regulator
LF= Wide Voltage Range with no internal core voltage regulator

4: T = In tape and reel TQFP packages only

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, BeaconThings, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, 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, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, 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.

© 2010-2017, Microchip Technology Incorporated, All Rights Reserved.

ISBN: 978-1-5224-1560-2

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

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-3326-8000
Fax: 86-21-3326-8021

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880- 3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

France - Saint Cloud
Tel: 33-1-30-60-70-00

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: 47-7289-7561

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

11/07/16