



dsPIC30F4011/4012

Data Sheet

High-Performance, 16-Bit
Digital Signal Controllers

Note the following details of the code protection feature on Microchip devices:

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

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

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in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates.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

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELoQ, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


AmpLab, FilterLab, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Linear Active Thermistor, Mindi, MiWi, MPASM, MPLIB, MPLINK, PICKit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

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

© 2007, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2002 ==

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. 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.



dsPIC30F4011/4012

dsPIC30F4011/4012 Enhanced Flash 16-Bit Digital Signal Controller

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the "dsPIC30F Family Reference Manual" (DS70046). For more information on the device instruction set and programming, refer to the "dsPIC30F/33F Programmer's Reference Manual" (DS70157).

High-Performance, Modified RISC CPU:

- Modified Harvard architecture
- C compiler optimized instruction set architecture with flexible addressing modes
- 83 base instructions
- 24-bit wide instructions, 16-bit wide data path
- 48 Kbytes on-chip Flash program space (16K instruction words)
- 2 Kbytes of on-chip data RAM
- 1 Kbyte of nonvolatile data EEPROM
- Up to 30 MIPS operation:
 - DC to 40 MHz external clock input
 - 4 MHz-10 MHz oscillator input with PLL active (4x, 8x, 16x)
- 30 interrupt sources:
 - 3 external interrupt sources
 - 8 user-selectable priority levels for each interrupt source
 - 4 processor trap sources
- 16 x 16-bit working register array

DSP Engine Features:

- Dual data fetch
- Accumulator write-back for DSP operations
- Modulo and Bit-Reversed Addressing modes
- Two, 40-bit wide accumulators with optional saturation logic
- 17-bit x 17-bit single-cycle hardware fractional/integer multiplier
- All DSP instructions are single cycle
- ± 16 -bit, single-cycle shift

Peripheral Features:

- High-current sink/source I/O pins: 25 mA/25 mA
- Timer module with programmable prescaler:
 - Five 16-bit timers/counters; optionally pair 16-bit timers into 32-bit timer modules
- 16-bit Capture input functions
- 16-bit Compare/PWM output functions
- 3-wire SPI modules (supports 4 Frame modes)
- I²C™ module supports Multi-Master/Slave mode and 7-bit/10-bit addressing
- 2 UART modules with FIFO Buffers
- 1 CAN module, 2.0B compliant

Motor Control PWM Module Features:

- 6 PWM output channels:
 - Complementary or Independent Output modes
 - Edge and Center-Aligned modes
- 3 duty cycle generators
- Dedicated time base
- Programmable output polarity
- Dead-time control for Complementary mode
- Manual output control
- Trigger for A/D conversions

Quadrature Encoder Interface Module Features:

- Phase A, Phase B and Index Pulse input
- 16-bit up/down position counter
- Count direction status
- Position Measurement (x2 and x4) mode
- Programmable digital noise filters on inputs
- Alternate 16-Bit Timer/Counter mode
- Interrupt on position counter rollover/underflow

dsPIC30F4011/4012

Analog Features:

- 10-Bit Analog-to-Digital Converter (A/D) with 4 S/H inputs:
 - 1 Msps conversion rate
 - 9 input channels
 - Conversion available during Sleep and Idle
- Programmable Brown-out Reset

Special Digital Signal Controller Features:

- Enhanced Flash program memory:
 - 10,000 erase/write cycle (min.) for industrial temperature range, 100K (typical)
- Data EEPROM memory:
 - 100,000 erase/write cycle (min.) for industrial temperature range, 1M (typical)
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)

Special Digital Signal Controller Features (Cont.):

- Flexible Watchdog Timer (WDT) with on-chip, low-power RC oscillator for reliable operation
- Fail-Safe Clock Monitor operation detects clock failure and switches to on-chip, low-power RC oscillator
- Programmable code protection
- In-Circuit Serial Programming™ (ICSP™)
- Selectable Power Management modes:
 - Sleep, Idle and Alternate Clock modes

CMOS Technology:

- Low-power, high-speed Flash technology
- Wide operating voltage range (2.5V to 5.5V)
- Industrial and Extended temperature ranges
- Low-power consumption

dsPIC30F Motor Control and Power Conversion Family*

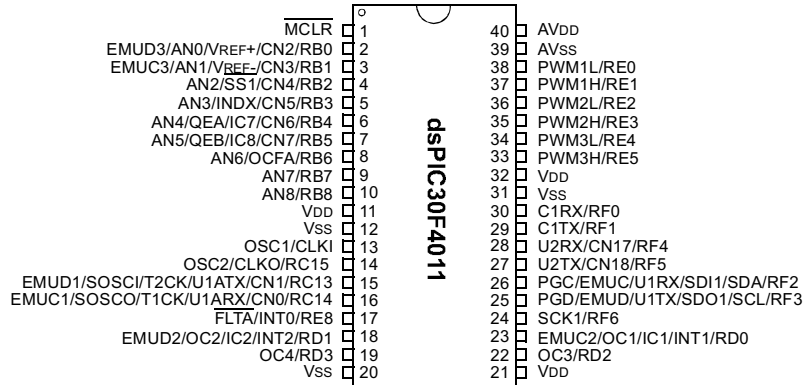
| Device | Pins | Program Mem. Bytes/Instructions | SRAM Bytes | EEPROM Bytes | Timer 16-bit | Input Cap | Output Comp/Std PWM | Motor Control PWM | 10-Bit A/D 1 Msps | Quad Enc | UART | SPI | IC™ | CAN |
|--------------|-------|---------------------------------|------------|--------------|--------------|-----------|---------------------|-------------------|-------------------|----------|------|-----|-----|-----|
| dsPIC30F2010 | 28 | 12K/4K | 512 | 1024 | 3 | 4 | 2 | 6 ch | 6 ch | Yes | 1 | 1 | 1 | - |
| dsPIC30F3010 | 28 | 24K/8K | 1024 | 1024 | 5 | 4 | 2 | 6 ch | 6 ch | Yes | 1 | 1 | 1 | - |
| dsPIC30F4012 | 28 | 48K/16K | 2048 | 1024 | 5 | 4 | 2 | 6 ch | 6 ch | Yes | 1 | 1 | 1 | 1 |
| dsPIC30F3011 | 40/44 | 24K/8K | 1024 | 1024 | 5 | 4 | 4 | 6 ch | 9 ch | Yes | 2 | 1 | 1 | - |
| dsPIC30F4011 | 40/44 | 48K/16K | 2048 | 1024 | 5 | 4 | 4 | 6 ch | 9 ch | Yes | 2 | 1 | 1 | 1 |
| dsPIC30F5015 | 64 | 66K/22K | 2048 | 1024 | 5 | 4 | 4 | 8 ch | 16 ch | Yes | 1 | 2 | 1 | 1 |
| dsPIC30F6010 | 80 | 144K/48K | 8192 | 4096 | 5 | 8 | 8 | 8 ch | 16 ch | Yes | 2 | 2 | 1 | 2 |

* This table provides a summary of the dsPIC30F6010 peripheral features. Other available devices in the dsPIC30F Motor Control and Power Conversion Family are shown for feature comparison.

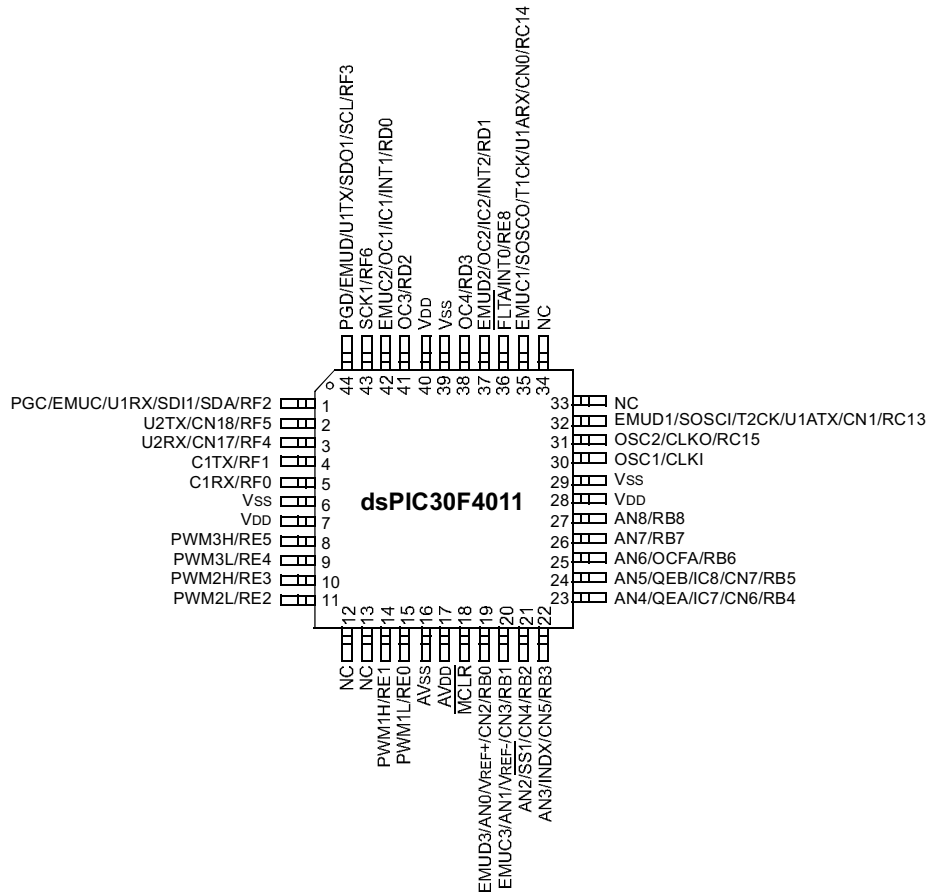
dsPIC30F4011/4012

Pin Diagrams

40-Pin PDIP

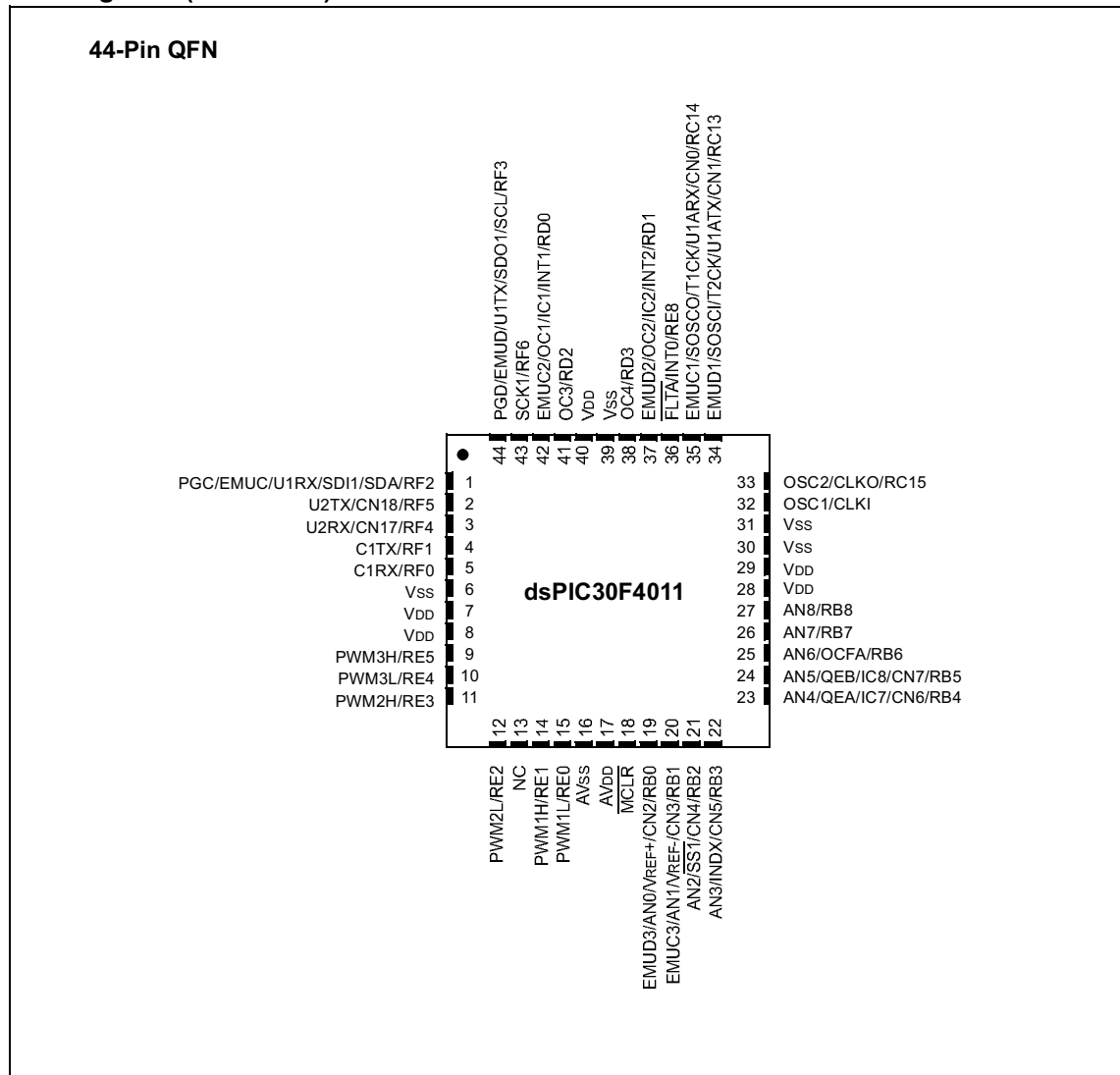


44-Pin TQFP



dsPIC30F4011/4012

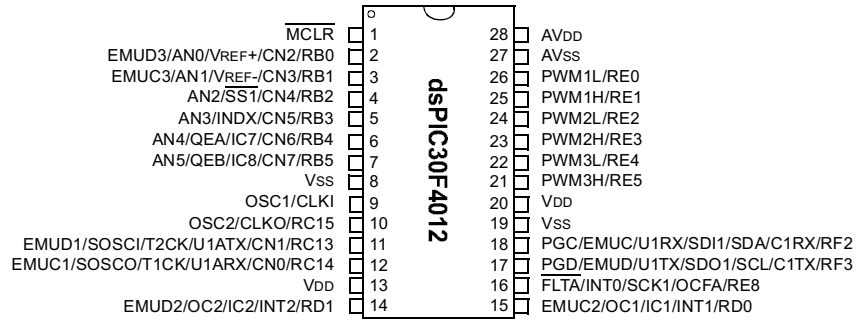
Pin Diagrams (Continued)



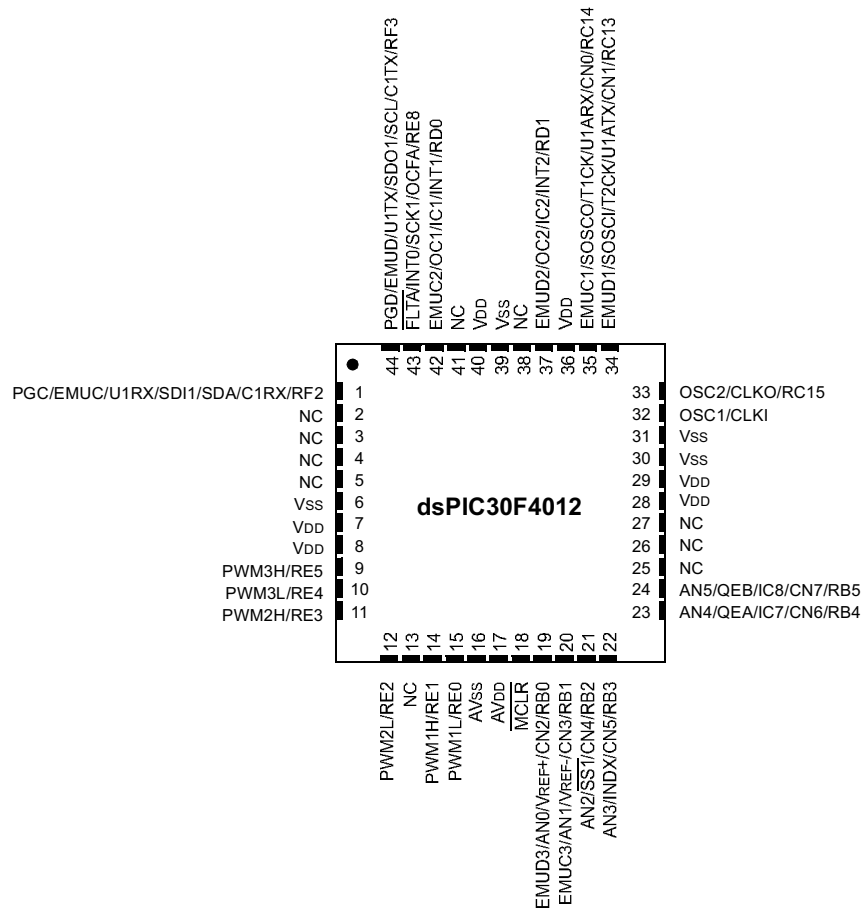
dsPIC30F4011/4012

Pin Diagrams (Continued)

28-Pin SPDIP and SOIC



44-Pin QFN



dsPIC30F4011/4012

Table of Contents

| | | |
|------|---|-----|
| 1.0 | Device Overview | 7 |
| 2.0 | CPU Architecture Overview..... | 13 |
| 3.0 | Memory Organization | 21 |
| 4.0 | Address Generator Units..... | 33 |
| 5.0 | Interrupts | 39 |
| 6.0 | Flash Program Memory..... | 45 |
| 7.0 | Data EEPROM Memory | 51 |
| 8.0 | I/O Ports | 57 |
| 9.0 | Timer1 Module | 63 |
| 10.0 | Timer2/3 Module | 67 |
| 11.0 | Timer4/5 Module | 73 |
| 12.0 | Input Capture Module..... | 77 |
| 13.0 | Output Compare Module..... | 81 |
| 14.0 | Quadrature Encoder Interface (QEI) Module | 85 |
| 15.0 | Motor Control PWM Module..... | 91 |
| 16.0 | SPI Module..... | 103 |
| 17.0 | I2C™ Module | 107 |
| 18.0 | Universal Asynchronous Receiver Transmitter (UART) Module | 115 |
| 19.0 | CAN Module | 123 |
| 20.0 | 10-bit, High-Speed Analog-to-Digital Converter (ADC) Module | 133 |
| 21.0 | System Integration | 145 |
| 22.0 | Instruction Set Summary..... | 159 |
| 23.0 | Development Support..... | 167 |
| 24.0 | Electrical Characteristics | 171 |
| 25.0 | Packaging Information..... | 213 |
| | Appendix A: Revision History..... | 221 |
| | Index | 223 |
| | The Microchip Web Site..... | 229 |
| | Customer Change Notification Service | 229 |
| | Customer Support..... | 229 |
| | Reader Response | 230 |
| | Product Identification System..... | 231 |

TO OUR VALUED CUSTOMERS

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

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

Most Current Data Sheet

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

<http://www.microchip.com>

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

Errata

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

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

- Microchip's Worldwide Web site; <http://www.microchip.com>
- Your local Microchip sales office (see last page)

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

Customer Notification System

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

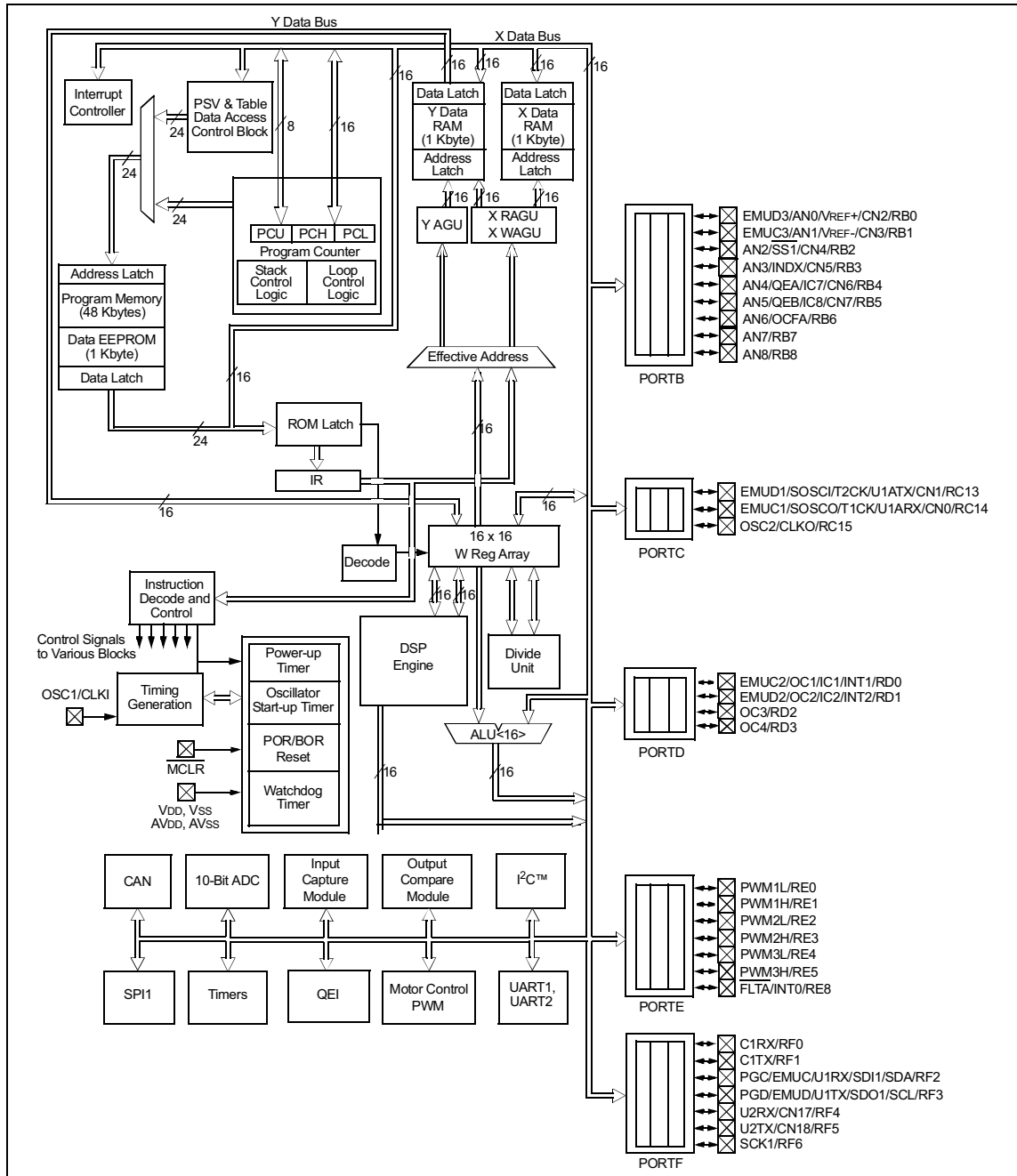
dsPIC30F4011/4012

1.0 DEVICE OVERVIEW

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the "dsPIC30F Family Reference Manual" (DS70046). For more information on the device instruction set and programming, refer to the "dsPIC30F/33F Programmer's Reference Manual" (DS70157).

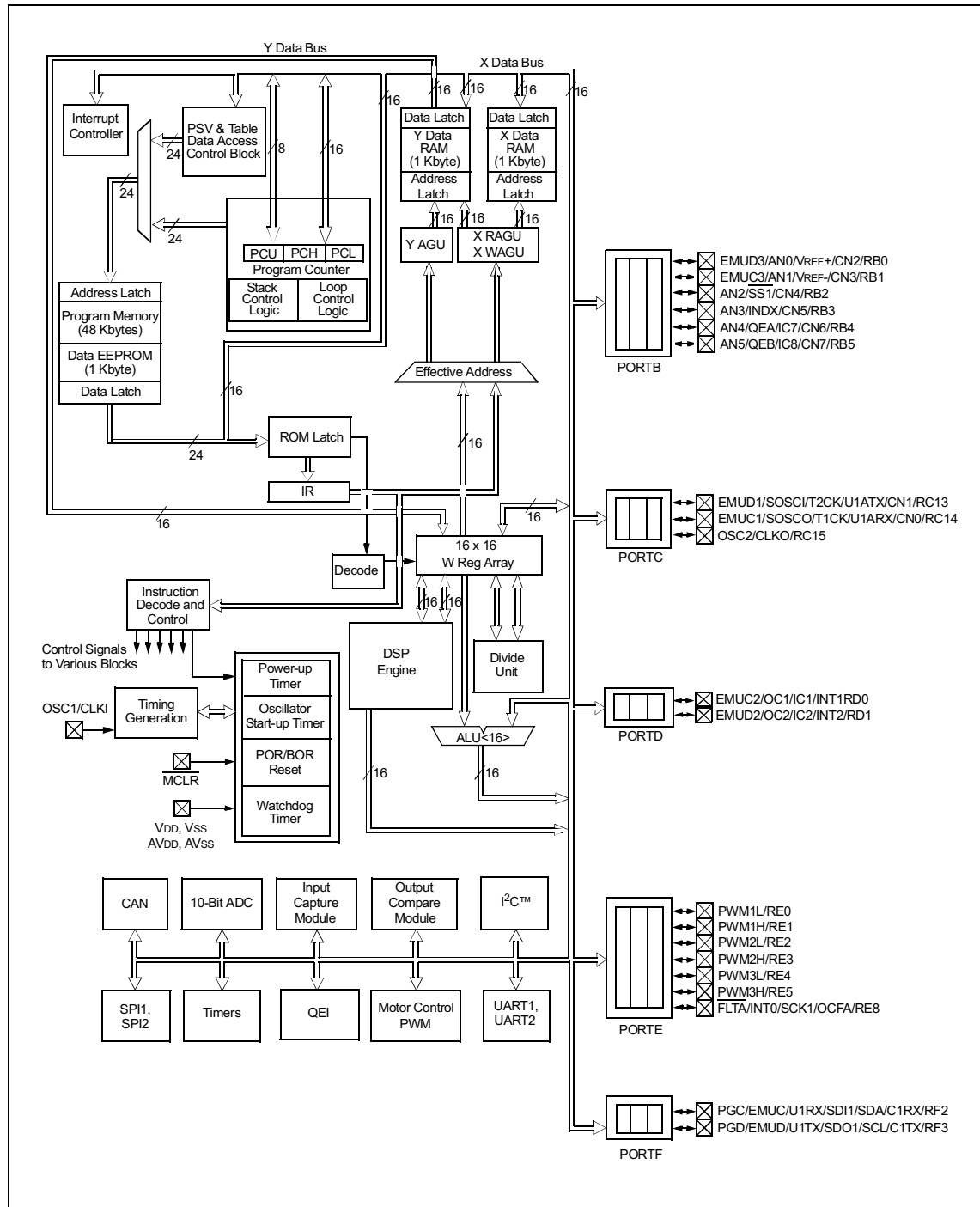
This document contains device-specific information for the dsPIC30F4011/4012 devices. The dsPIC30F devices contain extensive Digital Signal Processor (DSP) functionality within a high-performance, 16-bit microcontroller (MCU) architecture. Figure 1-1 and Figure 1-2 show device block diagrams for the dsPIC30F4011 and dsPIC30F4012 devices.

FIGURE 1-1: dsPIC30F4011 BLOCK DIAGRAM



dsPIC30F4011/4012

FIGURE 1-2: dsPIC30F4012 BLOCK DIAGRAM



dsPIC30F4011/4012

Table 1-1 provides a brief description of the device I/O pinout and the functions that are multiplexed to a port pin. Multiple functions may exist on one port pin. When multiplexing occurs, the peripheral module's functional requirements may force an override of the data direction of the port pin.

TABLE 1-1: dsPIC30F4011 I/O PIN DESCRIPTIONS

| Pin Name | Pin Type | Buffer Type | Description |
|--|--|--|--|
| AN0-AN8 | I | Analog | Analog input channels. AN0 and AN1 are also used for device programming data and clock inputs, respectively. |
| AVDD | P | P | Positive supply for analog module. |
| AVSS | P | P | Ground reference for analog module. |
| CLKI CLKO | I O | ST/CMOS — | External clock source input. Always associated with OSC1 pin function. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes. Always associated with OSC2 pin function. |
| CN0-CN7 CN17-CN18 | I | ST | Input change notification inputs. Can be software programmed for internal weak pull-ups on all inputs. |
| C1RX C1TX | I O | ST — | CAN1 bus receive pin. CAN1 bus transmit pin. |
| EMUD EMUC EMUD1 EMUC1 EMUD2 EMUC2 EMUD3 EMUC3 | I/O I/O I/O I/O I/O I/O I/O I/O | ST ST ST ST ST ST ST ST | ICD Primary Communication Channel data input/output pin. ICD Primary Communication Channel clock input/output pin. ICD Secondary Communication Channel data input/output pin. ICD Secondary Communication Channel clock input/output pin. ICD Tertiary Communication Channel data input/output pin. ICD Tertiary Communication Channel clock input/output pin. ICD Quaternary Communication Channel data input/output pin. ICD Quaternary Communication Channel clock input/output pin. |
| IC1, IC2, IC7, IC8 | I | ST | Capture inputs 1, 2, 7 and 8. |
| INDX QEA QEB | I I I | ST ST ST | Quadrature Encoder Index Pulse input. Quadrature Encoder Phase A input in QE1 mode. Auxiliary Timer External Clock/Gate input in Timer mode. Quadrature Encoder Phase A input in QE1 mode. Auxiliary Timer External Clock/Gate input in Timer mode. |
| INT0 INT1 INT2 | I I I | ST ST ST | External interrupt 0. External interrupt 1. External interrupt 2. |
| FLTA PWM1L PWM1H PWM2L PWM2H PWM3L PWM3H | I O O O O O O | ST — — — — — — | PWM Fault A input. PWM1 low output. PWM1 high output. PWM2 low output. PWM2 high output. PWM3 low output. PWM3 high output. |
| MCLR | I/P | ST | Master Clear (Reset) input or programming voltage input. This pin is an active-low Reset to the device. |
| OCFA OC1-OC4 | I O | ST — | Compare Fault A input (for Compare channels 1, 2, 3 and 4). Compare outputs 1 through 4. |

Legend: CMOS = CMOS compatible input or output Analog = Analog input
 ST = Schmitt Trigger input with CMOS levels O = Output
 I = Input P = Power

dsPIC30F4011/4012

TABLE 1-1: dsPIC30F4011 I/O PIN DESCRIPTIONS (CONTINUED)

| Pin Name | Pin Type | Buffer Type | Description |
|--------------|----------|-------------|--|
| OSC1 | I | ST/CMOS | Oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise. |
| OSC2 | I/O | — | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLK0 in RC and EC modes. |
| PGD | I/O | ST | In-Circuit Serial Programming™ data input/output pin. |
| PGC | I | ST | In-Circuit Serial Programming clock input pin. |
| RB0-RB8 | I/O | ST | PORTB is a bidirectional I/O port. |
| 8RC13-RC15 | 8I/O | 8ST | PORTC is a bidirectional I/O port. |
| RD0-RD3 | I/O | ST | PORTD is a bidirectional I/O port. |
| RE0-RE5, RE8 | I/O | ST | PORTE is a bidirectional I/O port. |
| RF0-RF6 | I/O | ST | PORTF is a bidirectional I/O port. |
| SCK1 | I/O | ST | Synchronous serial clock input/output for SPI1. |
| SDI1 | I | ST | SPI1 data in. |
| SDO1 | O | — | SPI1 data out. |
| SS1 | I | ST | SPI1 slave synchronization. |
| SCL | I/O | ST | Synchronous serial clock input/output for I ² C™. |
| SDA | I/O | ST | Synchronous serial data input/output for I ² C. |
| SOSCO | O | — | 32 kHz low-power oscillator crystal output. |
| SOSCI | I | ST/CMOS | 32 kHz low-power oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise. |
| T1CK | I | ST | Timer1 external clock input. |
| T2CK | I | ST | Timer2 external clock input. |
| U1RX | I | ST | UART1 receive. |
| U1TX | O | — | UART1 transmit. |
| U1ARX | I | ST | UART1 alternate receive. |
| U1ATX | O | — | UART1 alternate transmit. |
| U2RX | I | ST | UART2 receive. |
| U2TX | O | — | UART2 transmit. |
| VDD | P | — | Positive supply for logic and I/O pins. |
| VSS | P | — | Ground reference for logic and I/O pins. |
| VREF+ | I | Analog | Analog voltage reference (high) input. |
| VREF- | I | Analog | Analog voltage reference (low) input. |

Legend: CMOS = CMOS compatible input or output Analog = Analog input
 ST = Schmitt Trigger input with CMOS levels O = Output
 I = Input P = Power

dsPIC30F4011/4012

Table 1-2 provides a brief description of the device I/O pinout and the functions that are multiplexed to a port pin. Multiple functions may exist on one port pin. When multiplexing occurs, the peripheral module's functional requirements may force an override of the data direction of the port pin.

TABLE 1-2: dsPIC30F4012 I/O PIN DESCRIPTIONS

| Pin Name | Pin Type | Buffer Type | Description |
|--|--|--|--|
| AN0-AN5 | I | Analog | Analog input channels. AN0 and AN1 are also used for device programming data and clock inputs, respectively. |
| AVDD | P | P | Positive supply for analog module. |
| AVSS | P | P | Ground reference for analog module. |
| CLKI CLKO | I O | ST/CMOS — | External clock source input. Always associated with OSC1 pin function. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes. Always associated with OSC2 pin function. |
| CN0-CN7 | I | ST | Input change notification inputs. Can be software programmed for internal weak pull-ups on all inputs. |
| C1RX C1TX | I O | ST — | CAN1 bus receive pin. CAN1 bus transmit pin. |
| EMUD EMUC EMUD1 EMUC1 EMUD2 EMUC2 EMUD3 EMUC3 | I/O I/O I/O I/O I/O I/O I/O I/O | ST ST ST ST ST ST ST ST | ICD Primary Communication Channel data input/output pin. ICD Primary Communication Channel clock input/output pin. ICD Secondary Communication Channel data input/output pin. ICD Secondary Communication Channel clock input/output pin. ICD Tertiary Communication Channel data input/output pin. ICD Tertiary Communication Channel clock input/output pin. ICD Quaternary Communication Channel data input/output pin. ICD Quaternary Communication Channel clock input/output pin. |
| IC1, IC2, IC7, IC8 | I | ST | Capture inputs 1, 2, 7 and 8. |
| INDX QEA QEB | I I I | ST ST ST | Quadrature Encoder Index Pulse input. Quadrature Encoder Phase A input in QE1 mode. Auxiliary Timer External Clock/Gate input in Timer mode. Quadrature Encoder Phase A input in QE1 mode. Auxiliary Timer External Clock/Gate input in Timer mode. |
| INT0 INT1 INT2 | I I I | ST ST ST | External interrupt 0. External interrupt 1. External interrupt 2. |
| FLTA PWM1L PWM1H PWM2L PWM2H PWM3L PWM3H | I O O O O O O | ST — — — — — — | PWM Fault A input. PWM1 low output. PWM1 high output. PWM2 low output. PWM2 high output. PWM3 low output. PWM3 high output. |
| MCLR | I/P | ST | Master Clear (Reset) input or programming voltage input. This pin is an active-low Reset to the device. |
| OCFA OC1, OC2 | I O | ST — | Compare Fault A input (for Compare channels 1, 2, 3 and 4). Compare outputs 1 and 2. |

Legend: CMOS = CMOS compatible input or output Analog = Analog input
 ST = Schmitt Trigger input with CMOS levels O = Output
 I = Input P = Power

dsPIC30F4011/4012

TABLE 1-2: dsPIC30F4012 I/O PIN DESCRIPTIONS (CONTINUED)

| Pin Name | Pin Type | Buffer Type | Description |
|------------------|----------|-------------|--|
| OSC1 | I | ST/CMOS | Oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise. |
| OSC2 | I/O | — | Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLK0 in RC and EC modes. |
| PGD | I/O | ST | In-Circuit Serial Programming™ data input/output pin. |
| PGC | I | ST | In-Circuit Serial Programming clock input pin. |
| RB0-RB5 | I/O | ST | PORTB is a bidirectional I/O port. |
| RC13-RC15 | 8I/O | 8ST | PORTC is a bidirectional I/O port. |
| RD0-RD1 | I/O | ST | PORTD is a bidirectional I/O port. |
| RE0-RE5, RE8 | I/O | ST | PORTE is a bidirectional I/O port. |
| RF2-RF3 | I/O | ST | PORTF is a bidirectional I/O port. |
| SCK1 | I/O | ST | Synchronous serial clock input/output for SPI1. |
| SDI1 | I | ST | SPI1 Data In. |
| SDO1 | O | — | SPI1 Data Out. |
| $\overline{SS}1$ | I/O | ST | SPI1 Slave Synchronization |
| SCL | I/O | ST | Synchronous serial clock input/output for I ² C™. |
| SDA | I/O | ST | Synchronous serial data input/output for I ² C. |
| SOSCO | O | — | 32 kHz low-power oscillator crystal output. |
| SOSCI | I | ST/CMOS | 32 kHz low-power oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise. |
| T1CK | I | ST | Timer1 external clock input. |
| T2CK | I | ST | Timer2 external clock input. |
| U1RX | I | ST | UART1 receive. |
| U1TX | O | — | UART1 transmit. |
| U1ARX | I | ST | UART1 alternate receive. |
| U1ATX | O | — | UART1 alternate transmit. |
| VDD | P | — | Positive supply for logic and I/O pins. |
| VSS | P | — | Ground reference for logic and I/O pins. |
| VREF+ | I | Analog | Analog voltage reference (high) input. |
| VREF- | I | Analog | Analog voltage reference (low) input. |

Legend: CMOS = CMOS compatible input or output Analog = Analog input
 ST = Schmitt Trigger input with CMOS levels O = Output
 I = Input P = Power

2.0 CPU ARCHITECTURE OVERVIEW

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the “dsPIC30F Family Reference Manual” (DS70046). For more information on the device instruction set and programming, refer to the “dsPIC30F/33F Programmer’s Reference Manual” (DS70157).

This document provides a summary of the dsPIC30F4011/4012 CPU and peripheral functions. For a complete description of this functionality, please refer to the “dsPIC30F Family Reference Manual” (DS70046).

2.1 Core Overview

The core has a 24-bit instruction word. The Program Counter (PC) is 23 bits wide with the Least Significant bit (LSb) always clear (see **Section 3.1 “Program Address Space”**), and the Most Significant bit (MSb) is ignored during normal program execution, except for certain specialized instructions. Thus, the PC can address up to 4M instruction words of user program space. An instruction prefetch mechanism is used to help maintain throughput. Program loop constructs, free from loop count management overhead, are supported using the `DO` and `REPEAT` instructions, both of which are interruptible at any point.

The working register array consists of 16x16-bit registers, each of which can act as data, address or offset registers. One working register (W15) operates as a software Stack Pointer for interrupts and calls.

The data space is 64 Kbytes (32K words) and is split into two blocks, referred to as X and Y data memory. Each block has its own independent Address Generation Unit (AGU). Most instructions operate solely through the X memory, AGU, which provides the appearance of a single, unified data space. The Multiply-Accumulate (MAC) class of dual source DSP instructions operate through both the X and Y AGUs, splitting the data address space into two parts (see **Section 3.2 “Data Address Space”**). The X and Y data space boundary is device-specific and cannot be altered by the user. Each data word consists of 2 bytes, and most instructions can address data either as words or bytes.

There are two methods of accessing data stored in program memory:

- The upper 32 Kbytes of data space memory can be mapped into the lower half (user space) of program space at any 16K program word boundary, defined by the 8-bit Program Space Visibility Page (PSVPAG) register. This lets any instruction access program space as if it were data space, with a limitation that the access requires an additional cycle. Moreover, only the lower 16 bits of each instruction word can be accessed using this method.

- SWWLinear indirect access of 32K word pages within program space is also possible, using any working register via table read and write instructions. Table read and write instructions can be used to access all 24 bits of an instruction word.

Overhead-free circular buffers (Modulo Addressing) are supported in both X and Y address spaces. This is primarily intended to remove the loop overhead for DSP algorithms.

The X AGU also supports Bit-Reversed Addressing on destination effective addresses, to greatly simplify input or output data reordering for radix-2 FFT algorithms. Refer to **Section 4.0 “Address Generator Units”** for details on Modulo and Bit-Reversed Addressing.

The core supports Inherent (no operand), Relative, Literal, Memory Direct, Register Direct, Register Indirect, Register Offset and Literal Offset Addressing modes. Instructions are associated with predefined addressing modes, depending upon their functional requirements.

For most instructions, the core is capable of executing a data (or program data) memory read, a working register (data) read, a data memory write and a program (instruction) memory read per instruction cycle. As a result, 3-operand instructions are supported, allowing $C = A + B$ operations to be executed in a single cycle.

A DSP engine has been included to significantly enhance the core arithmetic capability and throughput. It features a high-speed, 17-bit by 17-bit multiplier, a 40-bit ALU, two 40-bit saturating accumulators and a 40-bit bidirectional barrel shifter. Data in the accumulator, or any working register, can be shifted up to 16 bits right or 16 bits left in a single cycle. The DSP instructions operate seamlessly with all other instructions and have been designed for optimal real-time performance. The MAC class of instructions can concurrently fetch two data operands from memory, while multiplying two W registers. To enable this concurrent fetching of data operands, the data space has been split for these instructions and is linear for all others. This has been achieved in a transparent and flexible manner by dedicating certain working registers to each address space for the MAC class of instructions.

The core does not support a multi-stage instruction pipeline. However, a single-stage instruction prefetch mechanism is used, which accesses and partially decodes instructions a cycle ahead of execution in order to maximize available execution time. Most instructions execute in a single cycle with certain exceptions.

The core features a vectored exception processing structure for traps and interrupts, with 62 independent vectors. The exceptions consist of up to 8 traps (of which 4 are reserved) and 54 interrupts. Each interrupt is prioritized based on a user-assigned priority between 1 and 7 (1 being the lowest priority and 7 being the highest) in conjunction with a predetermined ‘natural order’. Traps have fixed priorities, ranging from 8 to 15.

dsPIC30F4011/4012

2.2 Programmer's Model

The programmer's model is shown in Figure 2-1 and consists of 16x16-bit working registers (W0 through W15), 2x40-bit accumulators (ACCA and ACCB), STATUS register (SR), Data Table Page register (TBLPAG), Program Space Visibility Page register (PSVPAG), DO and REPEAT registers (DOSTART, DOEND, DCOUNT and RCOUNT) and Program Counter (PC). The working registers can act as data, address or offset registers. All registers are memory mapped. W0 acts as the W register for file register addressing.

Some of these registers have a shadow register associated with each of them, as shown in Figure 2-1. The shadow register is used as a temporary holding register and can transfer its contents to or from its host register upon the occurrence of an event. None of the shadow registers are accessible directly. The following rules apply for transfer of registers into and out of shadows.

- `PUSH.S` and `POP.S`
W0, W1, W2, W3, SR (DC, N, OV, Z and C bits only) are transferred.
- `DO` instruction
DOSTART, DOEND, DCOUNT shadows are pushed on loop start and popped on loop end.

When a byte operation is performed on a working register, only the Least Significant Byte of the target register is affected. However, a benefit of memory mapped working registers is that both the Least and Most Significant Bytes can be manipulated through byte wide data memory space accesses.

2.2.1 SOFTWARE STACK POINTER/ FRAME POINTER

The dsPIC® Digital Signal Controllers contain a software stack. W15 is the dedicated software Stack Pointer (SP) and is automatically modified by exception processing and subroutine calls and returns. However, W15 can be referenced by any instruction in the same manner as all other W registers. This simplifies the reading, writing and manipulation of the Stack Pointer (e.g., creating stack frames).

Note: In order to protect against misaligned stack accesses, W15<0> is always clear.

W15 is initialized to 0x0800 during a Reset. The user may reprogram the SP during initialization to any location within data space.

W14 has been dedicated as a Stack Frame Pointer as defined by the `LNK` and `ULNK` instructions. However, W14 can be referenced by any instruction in the same manner as all other W registers.

2.2.2 STATUS REGISTER

The dsPIC DSC core has a 16-bit STATUS register (SR), the Least Significant Byte of which is referred to as the SR Low Byte (SRL) and the Most Significant Byte as the SR High Byte (SRH). See Figure 2-1 for SR layout.

SRL contains all the DSP ALU operation Status flags (including the Z bit), as well as the CPU Interrupt Priority Level Status bits, IPL<2:0>, and the Repeat Active Status bit, RA. During exception processing, SRL is concatenated with the MSB of the PC to form a complete word value which is then stacked.

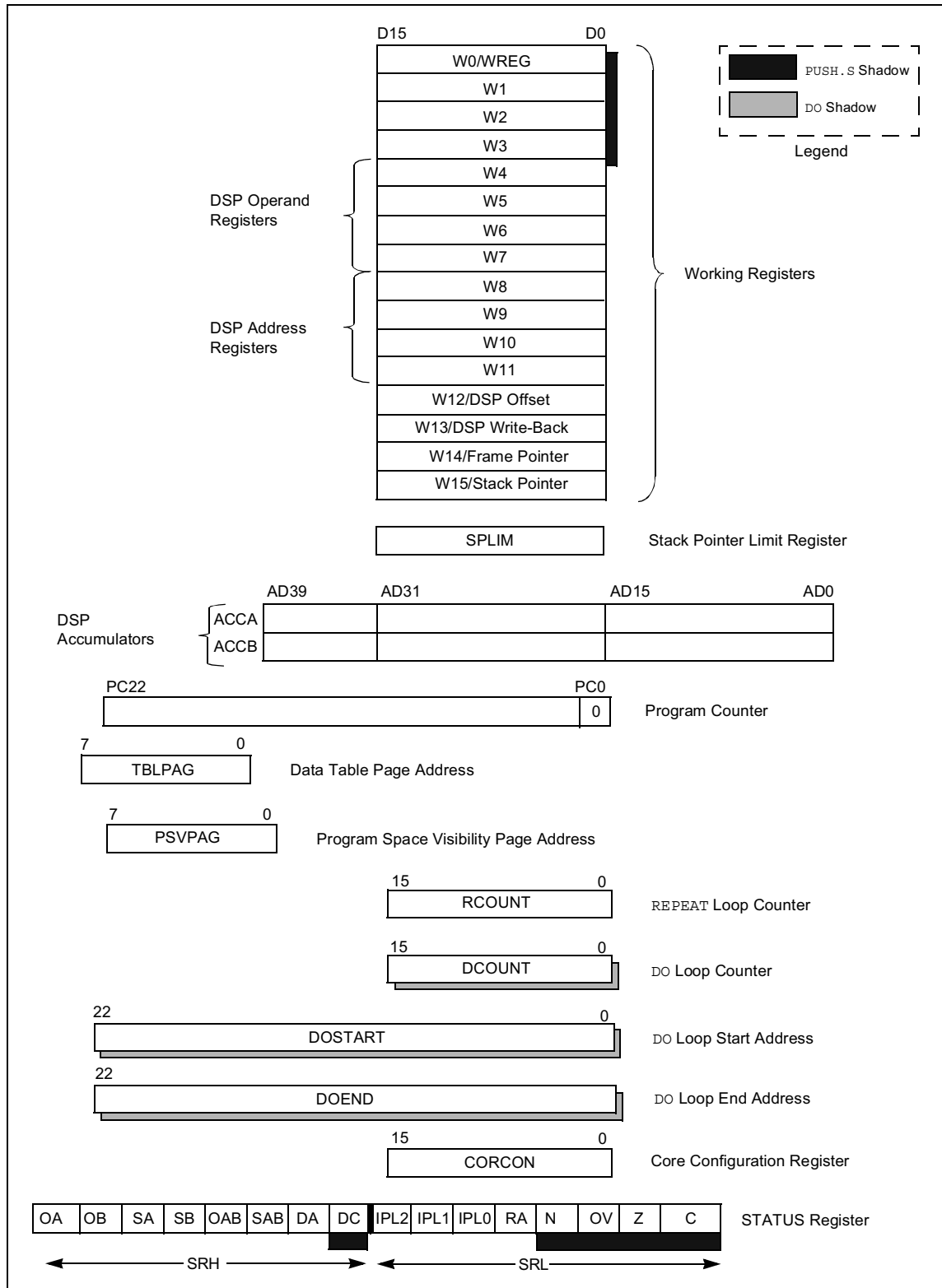
The upper byte of the SR register contains the DSP Adder/Subtractor Status bits, the DO Loop Active bit (DA) and the Digit Carry (DC) Status bit.

2.2.3 PROGRAM COUNTER

The Program Counter is 23 bits wide. Bit 0 is always clear; therefore, the PC can address up to 4M instruction words.

dsPIC30F4011/4012

FIGURE 2-1: dsPIC30F4011/4012 PROGRAMMER'S MODEL



dsPIC30F4011/4012

2.3 Divide Support

The dsPIC DSCs feature a 16/16-bit signed fractional divide operation, as well as 32/16-bit and 16/16-bit signed and unsigned integer divide operations, in the form of single instruction iterative divides. The following instructions and data sizes are supported:

1. `DIVF` – 16/16 signed fractional divide
2. `DIV.sd` – 32/16 signed divide
3. `DIV.ud` – 32/16 unsigned divide
4. `DIV.s` – 16/16 signed divide
5. `DIV.u` – 16/16 unsigned divide

The divide instructions must be executed within a `REPEAT` loop. Any other form of execution (e.g. a series of discrete divide instructions) will not function correctly because the instruction flow depends on `RCOUNT`. The divide instruction does not automatically set up the `RCOUNT` value and it must, therefore, be explicitly and correctly specified in the `REPEAT` instruction, as shown in Table 2-1 (`REPEAT` executes the target instruction {operand value + 1} times). The `REPEAT` loop count must be set up for 18 iterations of the `DIV/DIVF` instruction. Thus, a complete divide operation requires 19 cycles.

Note: The divide flow is interruptible. However, the user needs to save the context as appropriate.

TABLE 2-1: DIVIDE INSTRUCTIONS

| Instruction | Function |
|---------------------|---|
| <code>DIVF</code> | Signed fractional divide: $Wm/Wn \rightarrow W0$; $Rem \rightarrow W1$ |
| <code>DIV.sd</code> | Signed divide: $(Wm + 1:Wm)/Wn \rightarrow W0$; $Rem \rightarrow W1$ |
| <code>DIV.s</code> | Signed divide: $Wm/Wn \rightarrow W0$; $Rem \rightarrow W1$ |
| <code>DIV.ud</code> | Unsigned divide: $(Wm + 1:Wm)/Wn \rightarrow W0$; $Rem \rightarrow W1$ |
| <code>DIV.u</code> | Unsigned divide: $Wm/Wn \rightarrow W0$; $Rem \rightarrow W1$ |

2.4 DSP Engine

The DSP engine consists of a high-speed, 17-bit x 17-bit multiplier, a barrel shifter and a 40-bit adder/subtractor (with two target accumulators, round and saturation logic).

The dsPIC30F devices have a single instruction flow which can execute either DSP or MCU instructions. Many of the hardware resources are shared between the DSP and MCU instructions. For example, the instruction set has both DSP and MCU multiply instructions which use the same hardware multiplier.

The DSP engine also has the capability to perform inherent accumulator-to-accumulator operations which require no additional data. These instructions are `ADD`, `SUB` and `NEG`.

The DS0 engine has various options selected through various bits in the CPU Core Configuration register (`CORCON`), as listed below:

1. Fractional or integer DSP multiply (`IF`).
2. Signed or unsigned DSP multiply (`US`).
3. Conventional or convergent rounding (`RND`).
4. Automatic saturation on/off for `ACCA` (`SATA`).
5. Automatic saturation on/off for `ACCB` (`SATB`).
6. Automatic saturation on/off for writes to data memory (`SATDW`).
7. Accumulator Saturation mode selection (`ACCSAT`).

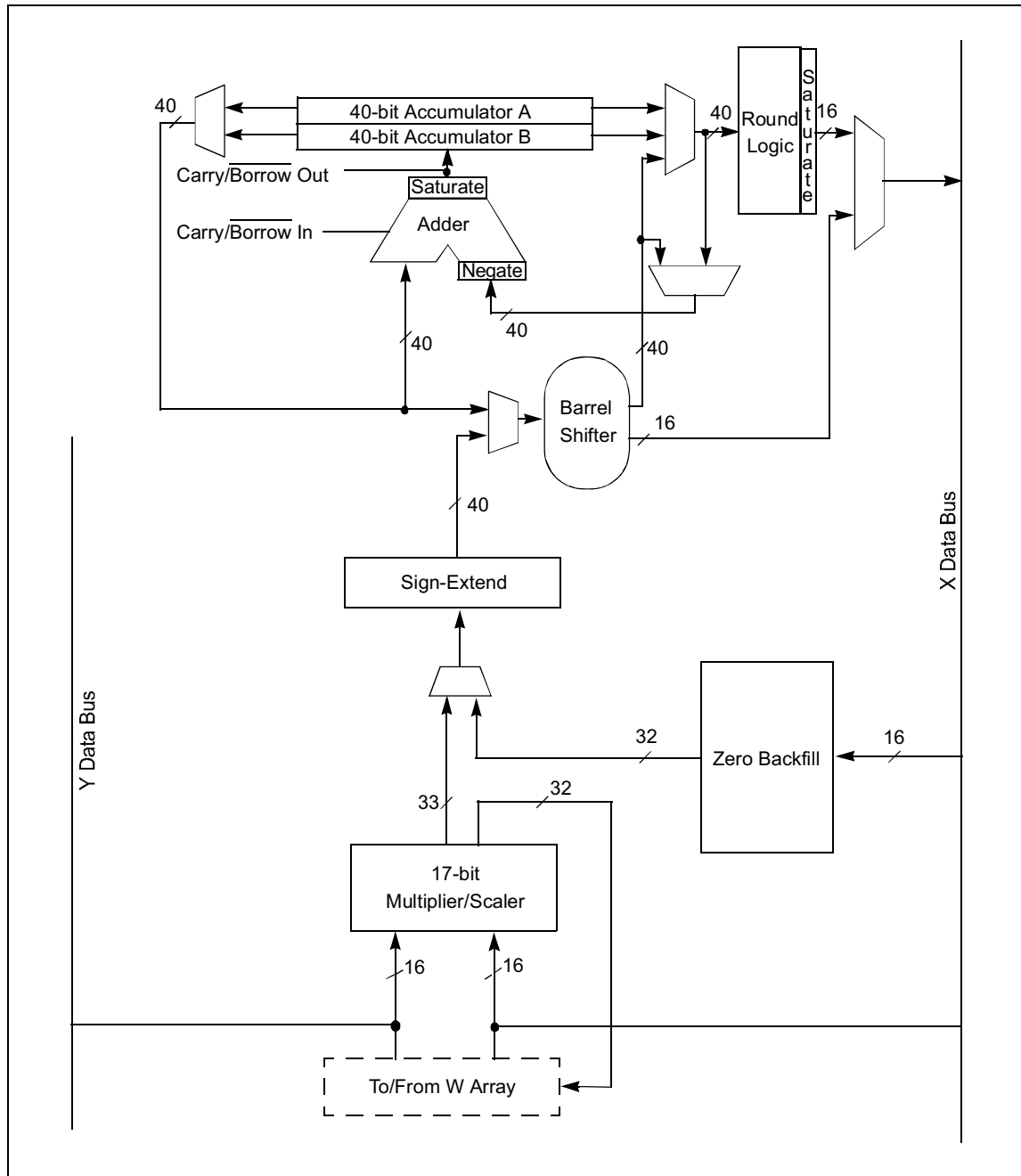
Note: For `CORCON` layout, see Table 3-3.

A block diagram of the DSP engine is shown in Figure 2-2.

TABLE 2-2: DSP INSTRUCTION SUMMARY

| Instruction | Algebraic Operation |
|---------------------|---------------------|
| <code>CLR</code> | $A = 0$ |
| <code>ED</code> | $A = (x - y)^2$ |
| <code>EDAC</code> | $A = A + (x - y)^2$ |
| <code>MAC</code> | $A = A + (x * y)$ |
| <code>MOVSAC</code> | No change in A |
| <code>MPY</code> | $A = x * y$ |
| <code>MPY.N</code> | $A = -x * y$ |
| <code>MSC</code> | $A = A - x * y$ |

FIGURE 2-2: DSP ENGINE BLOCK DIAGRAM



dsPIC30F4011/4012

2.4.1 MULTIPLIER

The 17x17-bit multiplier is capable of signed or unsigned operation and can multiplex its output using a scaler to support either 1.31 fractional (Q31) or 32-bit integer results. Unsigned operands are zero-extended into the 17th bit of the multiplier input value. Signed operands are sign-extended into the 17th bit of the multiplier input value. The output of the 17x17-bit multiplier/scaler is a 33-bit value, which is sign-extended to 40 bits. Integer data is inherently represented as a signed two's complement value, where the MSB is defined as a sign bit. Generally speaking, the range of an N-bit two's complement integer is -2^{N-1} to $2^{N-1} - 1$. For a 16-bit integer, the data range is -32768 (0x8000) to 32767 (0x7FFF), including 0. For a 32-bit integer, the data range is -2,147,483,648 (0x8000 0000) to 2,147,483,645 (0x7FFF FFFF).

When the multiplier is configured for fractional multiplication, the data is represented as a two's complement fraction, where the MSB is defined as a sign bit and the radix point is implied to lie just after the sign bit (QX format). The range of an N-bit two's complement fraction with this implied radix point is -1.0 to $(1-2^{1-N})$. For a 16-bit fraction, the Q15 data range is -1.0 (0x8000) to 0.999969482 (0x7FFF), including 0, and has a precision of 3.01518×10^{-5} . In Fractional mode, a 16x16 multiply operation generates a 1.31 product, which has a precision of 4.65661×10^{-10} .

The same multiplier is used to support the DSC multiply instructions, which include integer 16-bit signed, unsigned and mixed sign multiplies.

The MUL instruction may be directed to use byte or word-sized operands. Byte operands direct a 16-bit result, and word operands direct a 32-bit result to the specified register(s) in the W array.

2.4.2 DATA ACCUMULATORS AND ADDER/SUBTRACTER

The data accumulator consists of a 40-bit adder/subtractor with automatic sign extension logic. It can select one of two accumulators (A or B) as its pre-accumulation source and post-accumulation destination. For the ADD and LAC instructions, the data to be accumulated or loaded can be optionally scaled via the barrel shifter, prior to accumulation.

2.4.2.1 Adder/Subtractor, Overflow and Saturation

The adder/subtractor is a 40-bit adder with an optional zero input into one side and either true or complement data into the other input. In the case of addition, the carry/borrow input is active-high and the other input is true data (not complemented), whereas in the case of subtraction, the carry/borrow input is active-low and the other input is complemented. The adder/subtractor generates Overflow Status bits, SA/SB and OA/OB, which are latched and reflected in the STATUS register.

- Overflow from bit 39: this is a catastrophic overflow in which the sign of the accumulator is destroyed.
- Overflow into guard bits 32 through 39: this is a recoverable overflow. This bit is set whenever all the guard bits are not identical to each other.

The adder has an additional saturation block which controls accumulator data saturation, if selected. It uses the result of the adder, the Overflow Status bits described above, and the SATA/B (CORCON<7:6>) and ACCSAT (CORCON<4>) mode control bits to determine when and to what value to saturate.

Six STATUS register bits have been provided to support saturation and overflow; they are:

1. OA:
ACCA overflowed into guard bits
2. OB:
ACCB overflowed into guard bits
3. SA:
ACCA saturated (bit 31 overflow and saturation)
or
ACCA overflowed into guard bits and saturated (bit 39 overflow and saturation)
4. SB:
ACCB saturated (bit 31 overflow and saturation)
or
ACCB overflowed into guard bits and saturated (bit 39 overflow and saturation)
5. OAB:
Logical OR of OA and OB
6. SAB:
Logical OR of SA and SB

The OA and OB bits are modified each time data passes through the adder/subtractor. When set, they indicate that the most recent operation has overflowed into the accumulator guard bits (bits 32 through 39). Also, the OA and OB bits can optionally generate an arithmetic warning trap when set and the corresponding Overflow Trap Flag Enable bit (OVATE, OVBTE) in the INTCON1 register (refer to **Section 5.0 "Interrupts"**) is set. This allows the user to take immediate action, for example, to correct system gain.

The SA and SB bits are modified each time data passes through the adder/subtractor but can only be cleared by the user. When set, they indicate that the accumulator has overflowed its maximum range (bit 31 for 32-bit saturation or bit 39 for 40-bit saturation) and is saturated (if saturation is enabled). When saturation is not enabled, SA and SB default to bit 39 overflow and thus indicate that a catastrophic overflow has occurred. If the COVTE bit in the INTCON1 register is set, SA and SB bits generate an arithmetic warning trap when saturation is disabled.

The overflow and saturation Status bits can optionally be viewed in the STATUS register (SR) as the logical OR of OA and OB (in bit OAB) and the logical OR of SA and SB (in bit SAB). This allows programmers to check one bit in the STATUS register to determine if either accumulator has overflowed, or one bit to determine if either accumulator has saturated. This would be useful for complex number arithmetic which typically uses both the accumulators.

The device supports three Saturation and Overflow modes:

1. **Bit 39 Overflow and Saturation:**
When bit 39 overflow and saturation occurs, the saturation logic loads the maximally positive 9.31 (0x7FFFFFFF) or maximally negative 9.31 value (0x80000000) into the target accumulator. The SA or SB bit is set and remains set until cleared by the user. This is referred to as 'super saturation' and provides protection against erroneous data or unexpected algorithm problems (e.g., gain calculations).
2. **Bit 31 Overflow and Saturation:**
When bit 31 overflow and saturation occurs, the saturation logic then loads the maximally positive 1.31 value (0x007FFFFFFF) or maximally negative 1.31 value (0x00800000) into the target accumulator. The SA or SB bit is set and remains set until cleared by the user. When this Saturation mode is in effect, the guard bits are not used (so the OA, OB or OAB bits are never set).
3. **Bit 39 Catastrophic Overflow**
The bit 39 overflow Status bit from the adder is used to set the SA or SB bit, which remain set until cleared by the user. No saturation operation is performed and the accumulator is allowed to overflow (destroying its sign). If the COVTE bit in the INTCON1 register is set, a catastrophic overflow can initiate a trap exception.

2.4.2.2 Accumulator 'Write-Back'

The MAC class of instructions (with the exception of MPY, MPY.N, ED and EDAC) can optionally write a rounded version of the high word (bits 31 through 16) of the accumulator that is not targeted by the instruction into data space memory. The write is performed across the X bus into combined X and Y address space. The following addressing modes are supported:

1. **W13, Register Direct:**
The rounded contents of the non-target accumulator are written into W13 as a 1.15 fraction.
2. **[W13]+ = 2, Register Indirect with Post-Increment:**
The rounded contents of the non-target accumulator are written into the address pointed to by W13 as a 1.15 fraction. W13 is then incremented by 2 (for a word write).

2.4.2.3 Round Logic

The round logic is a combinational block, which performs a conventional (biased) or convergent (unbiased) round function during an accumulator write (store). The Round mode is determined by the state of the RND bit in the CORCON register. It generates a 16-bit, 1.15 data value which is passed to the data space write saturation logic. If rounding is not indicated by the instruction, a truncated 1.15 data value is stored and the least significant word is simply discarded.

Conventional rounding takes bit 15 of the accumulator, zero-extends it and adds it to the ACCxH word (bits 16 through 31 of the accumulator). If the ACCxL word (bits 0 through 15 of the accumulator) is between 0x8000 and 0xFFFF (0x8000 included), ACCxH is incremented. If ACCxL is between 0x0000 and 0x7FFF, ACCxH is left unchanged. A consequence of this algorithm is that over a succession of random rounding operations, the value tends to be biased slightly positive.

Convergent (or unbiased) rounding operates in the same manner as conventional rounding, except when ACCxL equals 0x8000. If this is the case, the LSb (bit 16 of the accumulator) of ACCxH is examined. If it is '1', ACCxH is incremented. If it is '0', ACCxH is not modified. Assuming that bit 16 is effectively random in nature, this scheme removes any rounding bias that may accumulate.

The SAC and SAC.R instructions store either a truncated (SAC) or rounded (SAC.R) version of the contents of the target accumulator to data memory, via the X bus (subject to data saturation, see **Section 2.4.2.4 "Data Space Write Saturation"**). Note that for the MAC class of instructions, the accumulator write-back operation functions in the same manner, addressing combined DSC (X and Y) data space though the X bus. For this class of instructions, the data is always subject to rounding.

dsPIC30F4011/4012

2.4.2.4 Data Space Write Saturation

In addition to adder/subtractor saturation, writes to data space may also be saturated, but without affecting the contents of the source accumulator. The data space write saturation logic block accepts a 16-bit, 1.15 fractional value from the round logic block as its input, together with overflow status from the original source (accumulator) and the 16-bit round adder. These are combined and used to select the appropriate 1.15 fractional value as output to write to data space memory.

If the SATDW bit in the CORCON register is set, data (after rounding or truncation) is tested for overflow and adjusted accordingly. For input data greater than 0x007FFF, data written to memory is forced to the maximum positive 1.15 value, 0x7FFF. For input data less than 0xFF8000, data written to memory is forced to the maximum negative 1.15 value, 0x8000. The MSb of the source (bit 39) is used to determine the sign of the operand being tested.

If the SATDW bit in the CORCON register is not set, the input data is always passed through unmodified under all conditions.

2.4.3 BARREL SHIFTER

The barrel shifter is capable of performing up to 16-bit arithmetic or logic right shifts, or up to 16-bit left shifts in a single cycle. The source can be either of the two DSP accumulators or the X bus (to support multi-bit shifts of register or memory data).

The shifter requires a signed binary value to determine both the magnitude (number of bits) and direction of the shift operation. A positive value shifts the operand right. A negative value shifts the operand left. A value of '0' does not modify the operand.

The barrel shifter is 40 bits wide, thereby obtaining a 40-bit result for DSP shift operations and a 16-bit result for MCU shift operations. Data from the X bus is presented to the barrel shifter between bit positions 16 to 31 for right shifts, and bit positions 0 to 15 for left shifts.

dsPIC30F4011/4012

3.0 MEMORY ORGANIZATION

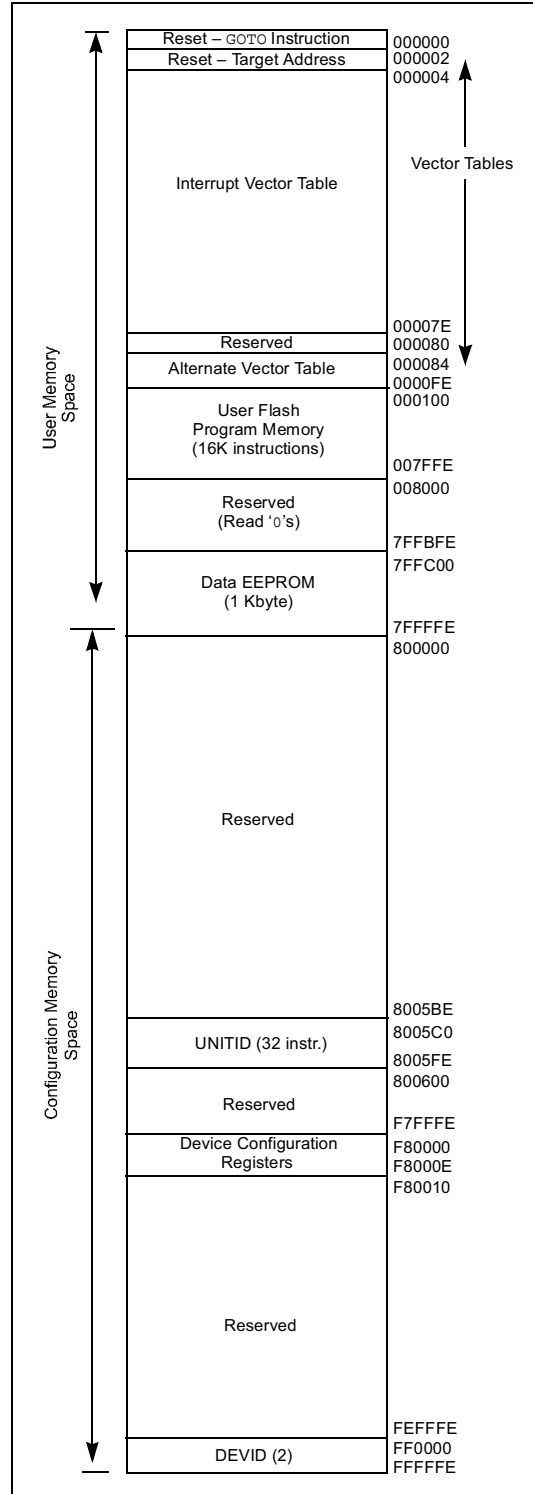
Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the "dsPIC30F Family Reference Manual" (DS70046). For more information on the device instruction set and programming, refer to the "dsPIC30F/33F Programmer's Reference Manual" (DS70157).

3.1 Program Address Space

The program address space is 4M instruction words. It is addressable by the 23-bit PC, table instruction Effective Address (EA) or data space EA, when program space is mapped into data space as defined by Table 3-1. Note that the program space address is incremented by two between successive program words in order to provide compatibility with data space addressing.

User program space access is restricted to the lower 4M instruction word address range (0x000000 to 0x7FFFFE) for all accesses other than TBLRD/TBLWT, which use TBLPAG<7> to determine user or configuration space access. In Table 3-1, read/write instructions, bit 23 allows access to the Device ID, the User ID and the Configuration bits; otherwise, bit 23 is always clear.

FIGURE 3-1: PROGRAM SPACE MEMORY MAP FOR dsPIC30F4011/4012

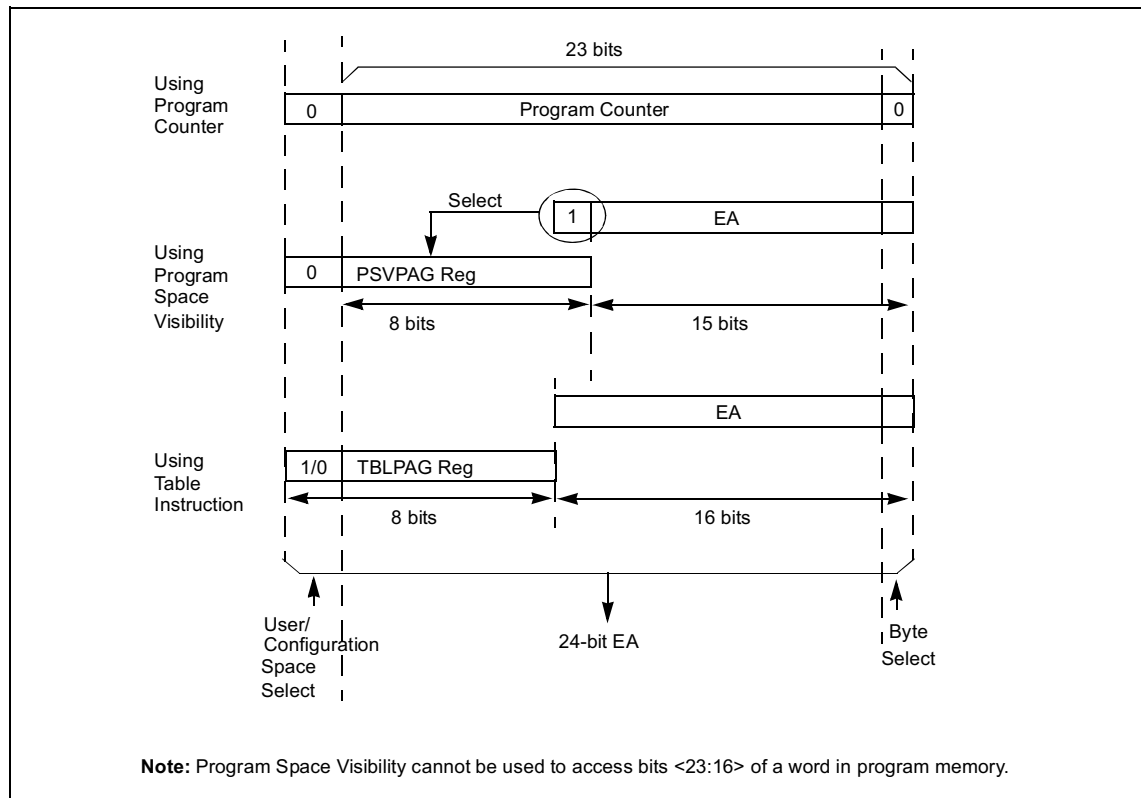


dsPIC30F4011/4012

TABLE 3-1: PROGRAM SPACE ADDRESS CONSTRUCTION

| Access Type | Access Space | Program Space Address | | | | |
|--------------------------|----------------------------------|-----------------------|-------------|---------------|---------------|-----|
| | | <23> | <22:16> | <15> | <14:1> | <0> |
| Instruction Access | User | 0 | PC<22:1> | | | 0 |
| TBLRD/TBLWT | User (TBLPAG<7> = 0) | TBLPAG<7:0> | | | Data EA<15:0> | |
| TBLRD/TBLWT | Configuration (TBLPAG<7> = 1) | TBLPAG<7:0> | | | Data EA<15:0> | |
| Program Space Visibility | User | 0 | PSVPAG<7:0> | Data EA<14:0> | | |

FIGURE 3-2: DATA ACCESS FROM PROGRAM SPACE ADDRESS GENERATION



3.1.1 DATA ACCESS FROM PROGRAM MEMORY USING TABLE INSTRUCTIONS

This architecture fetches 24-bit wide program memory. Consequently, instructions are always aligned. However, as the architecture is modified Harvard, data can also be present in program space.

There are two methods by which program space can be accessed; via special table instructions, or through the remapping of a 16K word program space page into the upper half of data space (see **Section 3.1.2 “Data Access From Program Memory Using Program Space Visibility”**). The **TBLRDL** and **TBLWTL** instructions offer a direct method of reading or writing the least significant word (lsw) of any address within program space, without going through data space. The **TBLRDH** and **TBLWTH** instructions are the only method whereby the upper 8 bits of a program space word can be accessed as data.

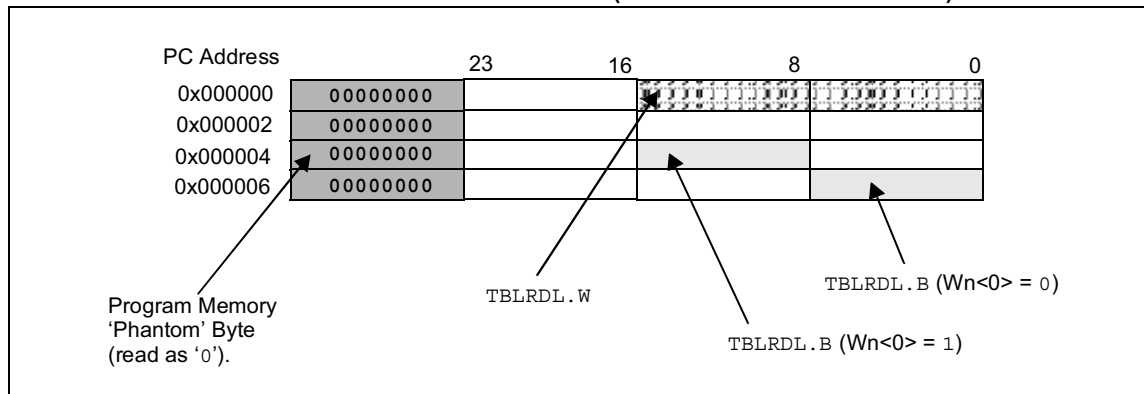
The PC is incremented by two for each successive 24-bit program word. This allows program memory addresses to directly map to data space addresses. Program memory can thus be regarded as two, 16-bit word-wide address spaces, residing side by side, each with the same address range. **TBLRDL** and **TBLWTL** access the space which contains the least significant data word, and **TBLRDH** and **TBLWTH** access the space which contains the Most Significant Byte of data.

Figure 3-2 shows how the EA is created for table operations and data space accesses (PSV = 1). Here, P<23:0> refers to a program space word, whereas D<15:0> refers to a data space word.

A set of table instructions is provided to move byte or word-sized data to and from program space (see Figure 3-3 and Figure 3-4).

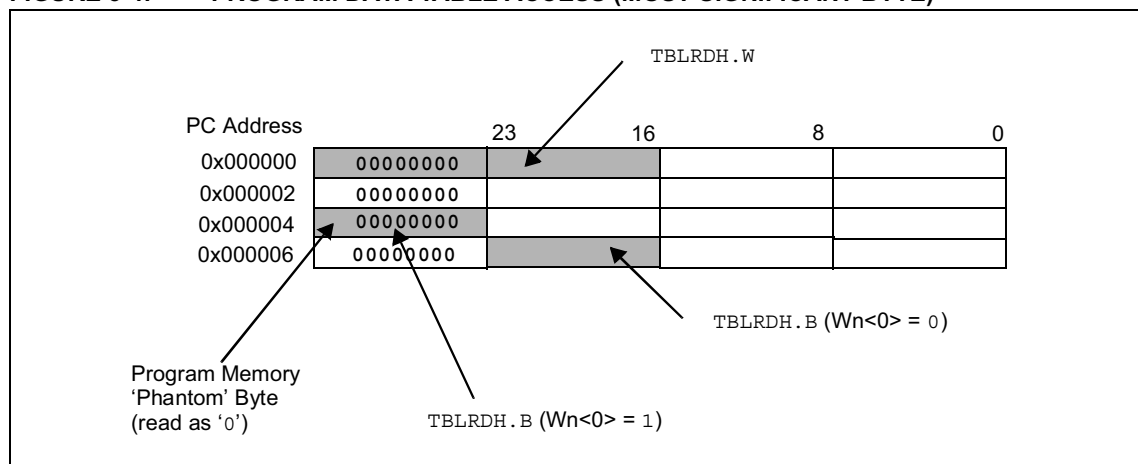
1. **TBLRDL**: Table Read Low
Word: Read the lsw of the program address; P<15:0> maps to D<15:0>.
Byte: Read one of the LSBs of the program address; P<7:0> maps to the destination byte when byte select = 0; P<15:8> maps to the destination byte when byte select = 1.
2. **TBLWTL**: Table Write Low (refer to **Section 6.0 “Flash Program Memory”** for details on Flash programming).
3. **TBLRDH**: Table Read High
Word: Read the msw of the program address; P<23:16> maps to D<7:0>; D<15:8> will always be 0.
Byte: Read one of the MSBs of the program address; P<23:16> maps to the destination byte when byte select = 0; The destination byte will always be ‘0’ when byte select = 1.
4. **TBLWTH**: Table Write High (refer to **Section 6.0 “Flash Program Memory”** for details on Flash programming).

FIGURE 3-3: PROGRAM DATA TABLE ACCESS (LEAST SIGNIFICANT WORD)



dsPIC30F4011/4012

FIGURE 3-4: PROGRAM DATA TABLE ACCESS (MOST SIGNIFICANT BYTE)



3.1.2 DATA ACCESS FROM PROGRAM MEMORY USING PROGRAM SPACE VISIBILITY

The upper 32 Kbytes of data space may optionally be mapped into any 16K word program space page. This provides transparent access of stored constant data from X data space without the need to use special instructions (i.e., TBLRDL/H, TBLWTL/H instructions).

Program space access through the data space occurs if the MSb of the data space EA is set and program space visibility is enabled by setting the PSV bit in the Core Control register (CORCON). The functions of CORCON are discussed in **Section 2.4 "DSP Engine"**.

Data accesses to this area add an additional cycle to the instruction being executed, since two program memory fetches are required.

Note that the upper half of addressable data space is always part of the X data space. Therefore, when a DSP operation uses program space mapping to access this memory region, Y data space should typically contain state (variable) data for DSP operations, whereas X data space should typically contain coefficient (constant) data.

Although each data space address, 0x8000 and higher, maps directly into a corresponding program memory address (see Figure 3-5), only the lower 16 bits of the 24-bit program word are used to contain the data. The upper 8 bits should be programmed to force an illegal instruction to maintain machine robustness. Refer to the "*dsPIC30F Programmer's Reference Manual*" (DS70030) for details on instruction encoding.

Note that by incrementing the PC by 2 for each program memory word, the Least Significant 15 bits of data space addresses directly map to the Least Significant 15 bits in the corresponding program space addresses. The remaining bits are provided by the Program Space Visibility Page register, PSVPAG<7:0>, as shown in Figure 3-5.

Note: PSV access is temporarily disabled during table reads/writes.

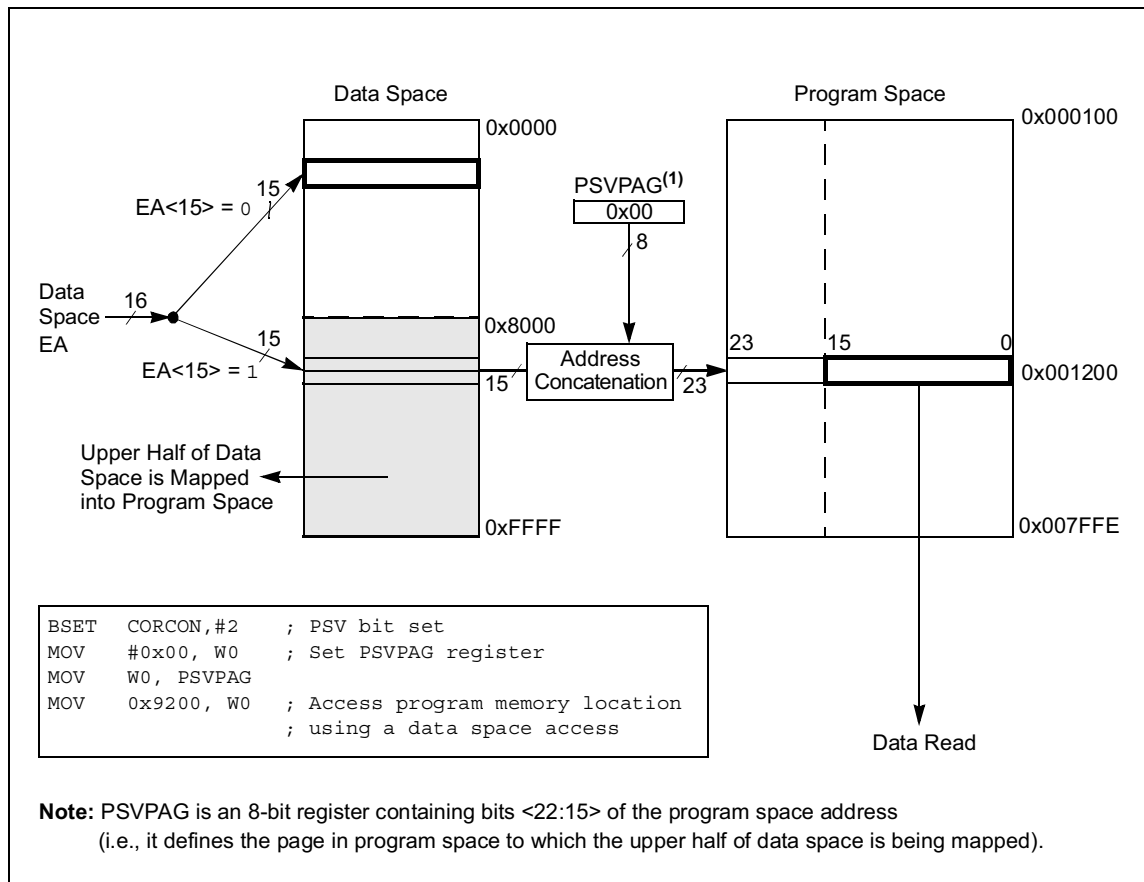
For instructions that use PSV which are executed outside a REPEAT loop:

- The following instructions require one instruction cycle in addition to the specified execution time:
 - MAC class of instructions with data operand prefetch
 - MOV instructions
 - MOV.D instructions
- All other instructions require two instruction cycles in addition to the specified execution time of the instruction.

For instructions that use PSV which are executed inside a REPEAT loop:

- The following instances require two instruction cycles in addition to the specified execution time of the instruction:
 - Execution in the first iteration
 - Execution in the last iteration
 - Execution prior to exiting the loop due to an interrupt
 - Execution upon re-entering the loop after an interrupt is serviced
- Any other iteration of the REPEAT loop allows the instruction, accessing data using PSV, to execute in a single cycle.

FIGURE 3-5: DATA SPACE WINDOW INTO PROGRAM SPACE OPERATION



3.2 Data Address Space

The core has two data spaces. The data spaces can be considered either separate (for some DSP instructions), or as one unified linear address range (for MCU instructions). The data spaces are accessed using two Address Generation Units (AGUs) and separate data paths.

3.2.1 DATA SPACE MEMORY MAP

The data space memory is split into two blocks, X and Y data space. A key element of this architecture is that Y space is a subset of X space, and is fully contained within X space. In order to provide an apparent linear addressing space, X and Y spaces have contiguous addresses.

When executing any instruction other than one of the MAC class of instructions, the X block consists of the 64-Kbyte data address space (including all Y addresses). When executing one of the MAC class of instructions, the X block consists of the 64-Kbyte data address space excluding the Y address block (for data reads only). In other words, all other instructions regard the entire data memory as one composite address space. The MAC class instructions extract the Y address space from data space and address it using EAs sourced from W10 and W11. The remaining X data space is addressed using W8 and W9. Both address spaces are concurrently accessed only with the MAC class instructions.

A data space memory map is shown in Figure 3-6.

Figure 3-7 shows a graphical summary of how X and Y data spaces are accessed for MCU and DSP instructions.

dsPIC30F4011/4012

FIGURE 3-6: dsPIC30F4011/4012 DATA SPACE MEMORY MAP

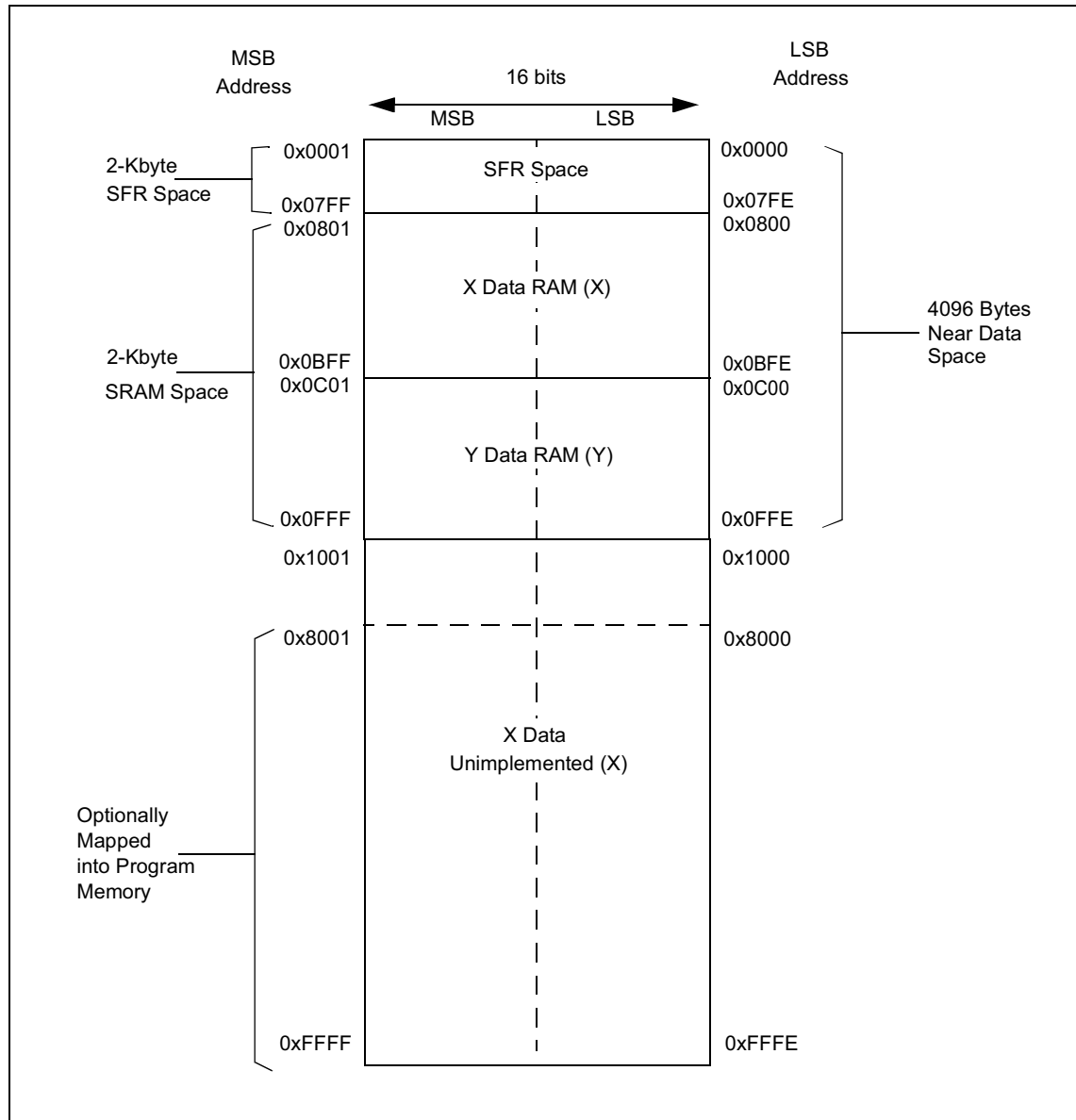
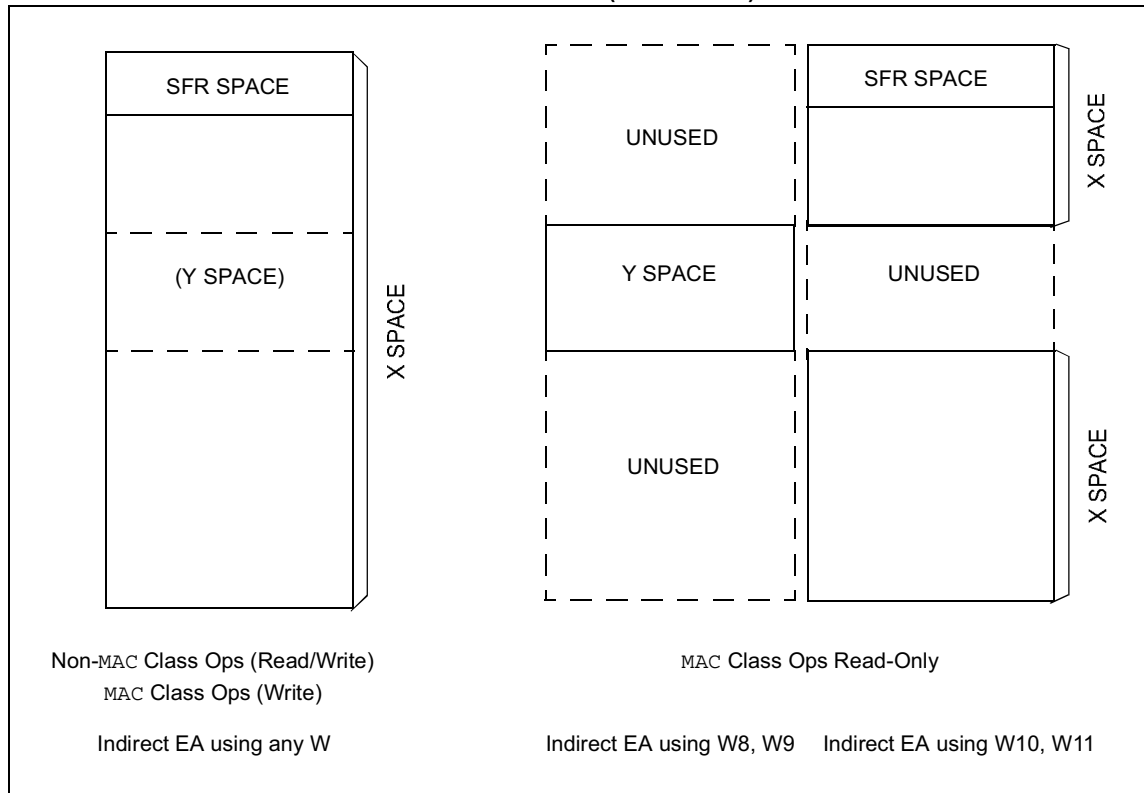


FIGURE 3-7: DATA SPACE FOR MCU AND DSP (MAC CLASS) INSTRUCTIONS EXAMPLE



dsPIC30F4011/4012

3.2.2 DATA SPACES

The X data space is used by all instructions and supports all addressing modes. There are separate read and write data buses. The X read data bus is the return data path for all instructions that view data space as combined X and Y address space. It is also the X address space data path for the dual operand read instructions (MAC class). The X write data bus is the only write path to data space for all instructions.

The X data space also supports Modulo Addressing for all instructions, subject to addressing mode restrictions. Bit-Reversed Addressing is only supported for writes to X data space.

The Y data space is used in concert with the X data space by the MAC class of instructions (CLR, ED, EDAC, MAC, MOVSAC, MPY, MPY.N and MSC) to provide two concurrent data read paths. No writes occur across the Y bus. This class of instructions dedicates two W register pointers, W10 and W11, to always address Y data space, independent of X data space, whereas W8 and W9 always address X data space. Note that during accumulator write-back, the data address space is considered a combination of X and Y data spaces, so the write occurs across the X bus. Consequently, the write can be to any address in the entire data space.

The Y data space can only be used for the data prefetch operation associated with the MAC class of instructions. It also supports Modulo Addressing for automated circular buffers. Of course, all other instructions can access the Y data address space through the X data path, as part of the composite linear space.

The boundary between the X and Y data spaces is defined as shown in Figure 3-6 and is not user-programmable. Should an EA point to data outside its own assigned address space, or to a location outside physical memory, an all-zero word/byte is returned. For example, although Y address space is visible by all non-MAC instructions using any addressing mode, an attempt by a MAC instruction to fetch data from that space, using W8 or W9 (X Space Pointers), returns 0x0000.

TABLE 3-2: EFFECT OF INVALID MEMORY ACCESSES

| Attempted Operation | Data Returned |
|---|---------------|
| EA = an unimplemented address | 0x0000 |
| W8 or W9 used to access Y data space in a MAC instruction | 0x0000 |
| W10 or W11 used to access X data space in a MAC instruction | 0x0000 |

All Effective Addresses (EA) are 16 bits wide and point to bytes within the data space. Therefore, the data space address range is 64 Kbytes or 32K words.

3.2.3 DATA SPACE WIDTH

The core data width is 16-bits. All internal registers are organized as 16-bit wide words. Data space memory is organized in byte addressable, 16-bit wide blocks.

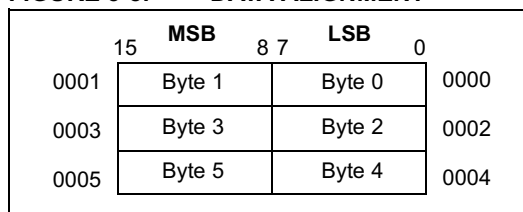
3.2.4 DATA ALIGNMENT

To help maintain backward compatibility with PIC® devices and improve data space memory usage efficiency, the dsPIC30F instruction set supports both word and byte operations. Data is aligned in data memory and registers as words, but all data space EAs resolve to bytes. Data byte reads read the complete word, which contains the byte, using the LSB of any EA to determine which byte to select. The selected byte is placed onto the LSB of the X data path (no byte accesses are possible from the Y data path as the MAC class of instructions can only fetch words). That is, data memory and registers are organized as two parallel byte-wide entities, with shared (word) address decode, but separate write lines. Data byte writes only write to the corresponding side of the array or register which matches the byte address.

As a consequence of this byte accessibility, all Effective Address calculations (including those generated by the DSP operations, which are restricted to word-sized data) are internally scaled to step through word-aligned memory. For example, the core would recognize that Post-Modified Register Indirect Addressing mode, [Ws++], will result in a value of Ws + 1 for byte operations and Ws + 2 for word operations.

All word accesses must be aligned to an even address. Misaligned word data fetches are not supported, so care must be taken when mixing byte and word operations, or translating from 8-bit MCU code. Should a misaligned read or write be attempted, an address error trap is generated. If the error occurred on a read, the instruction underway is completed, whereas if it occurred on a write, the instruction will be executed but the write does not occur. In either case, a trap is then executed, allowing the system and/or user to examine the machine state prior to execution of the address Fault.

FIGURE 3-8: DATA ALIGNMENT



dsPIC30F4011/4012

All byte loads into any W register are loaded into the LSB; the MSB is not modified.

A sign-extend (SE) instruction is provided to allow users to translate 8-bit signed data to 16-bit signed values. Alternatively, for 16-bit unsigned data, users can clear the MSB of any W register by executing a zero-extend (ZE) instruction on the appropriate address.

Although most instructions are capable of operating on word or byte data sizes, it should be noted that some instructions, including the DSP instructions, operate only on words.

3.2.5 NEAR DATA SPACE

An 8-Kbyte 'near' data space is reserved in X address memory space between 0x0000 and 0x1FFF, which is directly addressable via a 13-bit absolute address field within all memory direct instructions. The remaining X address space and all of the Y address space is addressable indirectly. Additionally, the whole of X data space is addressable using MOV instructions, which support Memory Direct Addressing with a 16-bit address field.

3.2.6 SOFTWARE STACK

The dsPIC DSC contains a software stack. W15 is used as the Stack Pointer.

The Stack Pointer always points to the first available free word and grows from lower addresses towards higher addresses. It pre-decrements for stack pops and post-increments for stack pushes, as shown in Figure 3-9. Note that for a PC push during any CALL instruction, the MSB of the PC is zero-extended before the push, ensuring that the MSB is always clear.

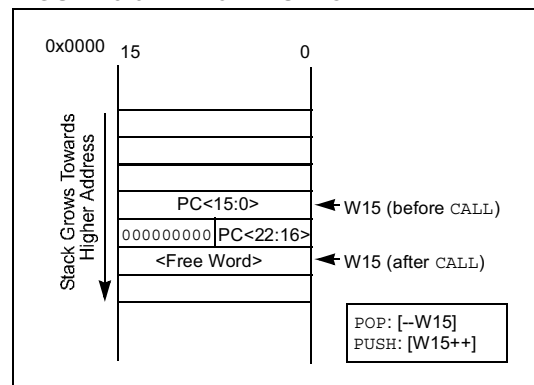
Note: A PC push during exception processing concatenates the SRL register to the MSB of the PC prior to the push.

There is a Stack Pointer Limit register (SPLIM) associated with the Stack Pointer. SPLIM is uninitialized at Reset. As is the case for the Stack Pointer, SPLIM<0> is forced to '0', because all stack operations must be word-aligned. Whenever an Effective Address (EA) is generated, using W15 as a source or destination pointer, the address thus generated is compared with the value in SPLIM. If the contents of the Stack Pointer (W15) and the SPLIM register are equal, and a push operation is performed, a stack error trap will not occur. The stack error trap will occur on a subsequent push operation. Thus, for example, if it is desirable to cause a stack error trap when the stack grows beyond address 0x2000 in RAM, initialize the SPLIM with the value, 0x1FFE.

Similarly, a Stack Pointer underflow (stack error) trap is generated when the Stack Pointer address is found to be less than 0x0800, thus preventing the stack from interfering with the Special Function Register (SFR) space.

A write to the SPLIM register should not be immediately followed by an indirect read operation using W15.

FIGURE 3-9: CALL STACK FRAME



dsPIC30F4011/4012

TABLE 3-3: CORE REGISTER MAP

| SFR Name | Address (Home) | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|----------------|---------------------------|--------|--------|--------|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|---------------------|
| W0 | 0000 | W0WREG | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W1 | 0002 | W1 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W2 | 0004 | W2 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W3 | 0006 | W3 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W4 | 0008 | W4 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W5 | 000A | W5 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W6 | 000C | W6 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W7 | 000E | W7 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W8 | 0010 | W8 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W9 | 0012 | W9 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W10 | 0014 | W10 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W11 | 0016 | W11 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W12 | 0018 | W12 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W13 | 001A | W13 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W14 | 001C | W14 | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| W15 | 001E | W15 | | | | | | | | | | | | | | | | 0000 1000 0000 0000 |
| SPLIM | 0020 | SPLIM | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| ACCAL | 0022 | ACCAL | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| ACCAH | 0024 | ACCAH | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| ACCAU | 0026 | Sign Extension (ACCA<39>) | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| ACCBH | 0028 | ACCBH | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| ACCBH | 002A | ACCBH | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| ACCBH | 002C | Sign Extension (ACCB<39>) | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| PCL | 002E | PCL | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| PCH | 0030 | PCH | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| TBLPAG | 0032 | TBLPAG | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| PSVPAG | 0034 | PSVPAG | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| RCOUNT | 0036 | RCOUNT | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| DCOUNT | 0038 | DCOUNT | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| DOSTARTL | 003A | DOSTARTL | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| DOSTARTH | 003C | DOSTARTH | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| DOENDL | 003E | DOENDL | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| DOENDH | 0040 | DOENDH | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| SR | 0042 | OA | OB | SA | SB | OAB | SAB | DA | DC | IPL2 | IPL1 | IPL0 | RA | N | OV | Z | C | 0000 0000 0000 0000 |
| CORCON | 0044 | CORCON | | | | | | | | | | | | | | | | 0000 0000 0010 0000 |
| MODCON | 0046 | MODCON | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| XMODSRT | 0048 | XMODSRT | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

TABLE 3-3: CORE REGISTER MAP (CONTINUED)

| SFR Name | Address (Home) | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|----------------|--------|--------|--------|--------|--------|--------|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|---------------------|
| XMODEND | 004A | | | | | | | XE<15:1> | | | | | | | | | 1 | uuuu uuuu uuuu uuuu |
| YMODSRT | 004C | | | | | | | YS<15:1> | | | | | | | | | 0 | uuuu uuuu uuuu uuuu |
| YMODEND | 004E | | | | | | | YE<15:1> | | | | | | | | | 1 | uuuu uuuu uuuu uuuu |
| XBREV | 0050 | BREN | | | | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| DISICNT | 0052 | — | — | | | | | | | | | | | | | | | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the 'dsPIC30F Family Reference Manual' (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

4.0 ADDRESS GENERATOR UNITS

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the “dsPIC30F Family Reference Manual” (DS70046). For more information on the device instruction set and programming, refer to the “dsPIC30F/33F Programmer’s Reference Manual” (DS70157).

The dsPIC DSC core contains two independent address generator units: the X AGU and Y AGU. The Y AGU supports word-sized data reads for the DSP MAC class of instructions only. The dsPIC Digital Signal Controller AGUs support three types of data addressing:

- Linear Addressing
- Modulo (Circular) Addressing
- Bit-Reversed Addressing

Linear and Modulo Data Addressing modes can be applied to data space or program space. Bit-Reversed Addressing is only applicable to data space addresses.

4.1 Instruction Addressing Modes

The addressing modes in Table 4-1 form the basis of the addressing modes optimized to support the specific features of individual instructions. The addressing modes provided in the MAC class of instructions are somewhat different from those in the other instruction types.

4.1.1 FILE REGISTER INSTRUCTIONS

Most file register instructions use a 13-bit address field (f) to directly address data present in the first 8192 bytes of data memory (near data space). Most file register instructions employ a working register, W0, which is denoted as WREG in these instructions. The destination is typically either the same file register, or WREG (with the exception of the MUL instruction), which writes the result to a register or register pair. The MOV instruction allows additional flexibility and can access the entire data space during file register operation.

4.1.2 MCU INSTRUCTIONS

The three-operand MCU instructions are of the form:

Operand 3 = Operand 1 <function> Operand 2

where Operand 1 is always a working register (i.e., the addressing mode can only be Register Direct), which is referred to as Wb. Operand 2 can be a W register, fetched from data memory or a 5-bit literal. The result location can either be a W register or an address location. The following addressing modes are supported by MCU instructions:

- Register Direct
- Register Indirect
- Register Indirect Post-Modified
- Register Indirect Pre-Modified
- 5-bit or 10-bit Literal

Note: Not all instructions support all the addressing modes given above. Individual instructions may support different subsets of these addressing modes.

TABLE 4-1: FUNDAMENTAL ADDRESSING MODES SUPPORTED

| Addressing Mode | Description |
|--|--|
| File Register Direct | The address of the file register is specified explicitly. |
| Register Direct | The contents of a register are accessed directly. |
| Register Indirect | The contents of Wn forms the EA. |
| Register Indirect Post-Modified | The contents of Wn forms the EA. Wn is post-modified (incremented or decremented) by a constant value. |
| Register Indirect Pre-Modified | Wn is pre-modified (incremented or decremented) by a signed constant value to form the EA. |
| Register Indirect with Register Offset | The sum of Wn and Wb forms the EA. |
| Register Indirect with Literal Offset | The sum of Wn and a literal forms the EA. |

dsPIC30F4011/4012

4.1.3 MOVE AND ACCUMULATOR INSTRUCTIONS

Move instructions and the DSP accumulator class of instructions provide a greater degree of addressing flexibility than other instructions. In addition to the addressing modes supported by most MCU instructions, move and accumulator instructions also support Register Indirect with Register Offset Addressing mode, also referred to as Register Indexed mode.

Note: For the `MOV` instructions, the addressing mode specified in the instruction can differ for the source and destination EA. However, the 4-bit `Wb` (register offset) field is shared between both source and destination (but typically only used by one).

In summary, the following addressing modes are supported by move and accumulator instructions:

- Register Direct
- Register Indirect
- Register Indirect Post-Modified
- Register Indirect Pre-Modified
- Register Indirect with Register Offset (Indexed)
- Register Indirect with Literal Offset
- 8-bit Literal
- 16-bit Literal

Note: Not all instructions support all the addressing modes given above. Individual instructions may support different subsets of these addressing modes.

4.1.4 MAC INSTRUCTIONS

The dual source operand DSP instructions (`CLR`, `ED`, `EDAC`, `MAC`, `MPY`, `MPY.N`, `MOVSAC` and `MSC`), also referred to as `MAC` instructions, utilize a simplified set of addressing modes to allow the user to effectively manipulate the Data Pointers through register indirect tables.

The two source operand prefetch registers must be members of the set {`W8`, `W9`, `W10`, `W11`}. For data reads, `W8` and `W9` will always be directed to the X AGU, and `W10` and `W11` will always be directed to the Y AGU. The Effective Addresses (EA) generated (before and after modification) must, therefore, be valid addresses within X data space for `W8` and `W9`, and Y data space for `W10` and `W11`.

Note: Register Indirect with Register Offset Addressing is only available for `W9` (in X data space) and `W11` (in Y data space).

In summary, the following addressing modes are supported by the `MAC` class of instructions:

- Register Indirect
- Register Indirect Post-Modified by 2
- Register Indirect Post-Modified by 4
- Register Indirect Post-Modified by 6
- Register Indirect with Register Offset (Indexed)

4.1.5 OTHER INSTRUCTIONS

Besides the various addressing modes outlined above, some instructions use literal constants of various sizes. For example, `BRA` (branch) instructions use 16-bit signed literals to specify the branch destination directly, whereas the `DISI` instruction uses a 14-bit unsigned literal field. In some instructions, such as `ADD ACC`, the source of an operand or result is implied by the opcode itself. Certain operations, such as `NOB`, do not have any operands.

4.2 Modulo Addressing

Modulo Addressing is a method of providing an automated means to support circular data buffers using hardware. The objective is to remove the need for software to perform data address boundary checks when executing tightly looped code, as is typical in many DSP algorithms.

Modulo Addressing can operate in either data or program space (since the Data Pointer mechanism is essentially the same for both). One circular buffer can be supported in each of the X (which also provides the pointers into program space) and Y data spaces. Modulo Addressing can operate on any W register pointer. However, it is not advisable to use `W14` or `W15` for Modulo Addressing, since these two registers are used as the Stack Frame Pointer and Stack Pointer, respectively.

In general, any particular circular buffer can only be configured to operate in one direction, as there are certain restrictions on the buffer start address (for incrementing buffers) or end address (for decrementing buffers) based upon the direction of the buffer.

The only exception to the usage restrictions is for buffers which have a power-of-2 length. As these buffers satisfy the start and end address criteria, they may operate in a Bidirectional mode (i.e., address boundary checks will be performed on both the lower and upper address boundaries).

4.2.1 START AND END ADDRESS

The Modulo Addressing scheme requires that a start and end address be specified and loaded into the 16-bit Modulo Buffer Address registers: XMODSRT, XMODEND, YMODSRT and YMODEND (see Table 3-3).

Note: Y space Modulo Addressing EA calculations assume word-sized data (LSb of every EA is always clear).

The length of a circular buffer is not directly specified. It is determined by the difference between the corresponding start and end addresses. The maximum possible length of the circular buffer is 32K words (64 Kbytes).

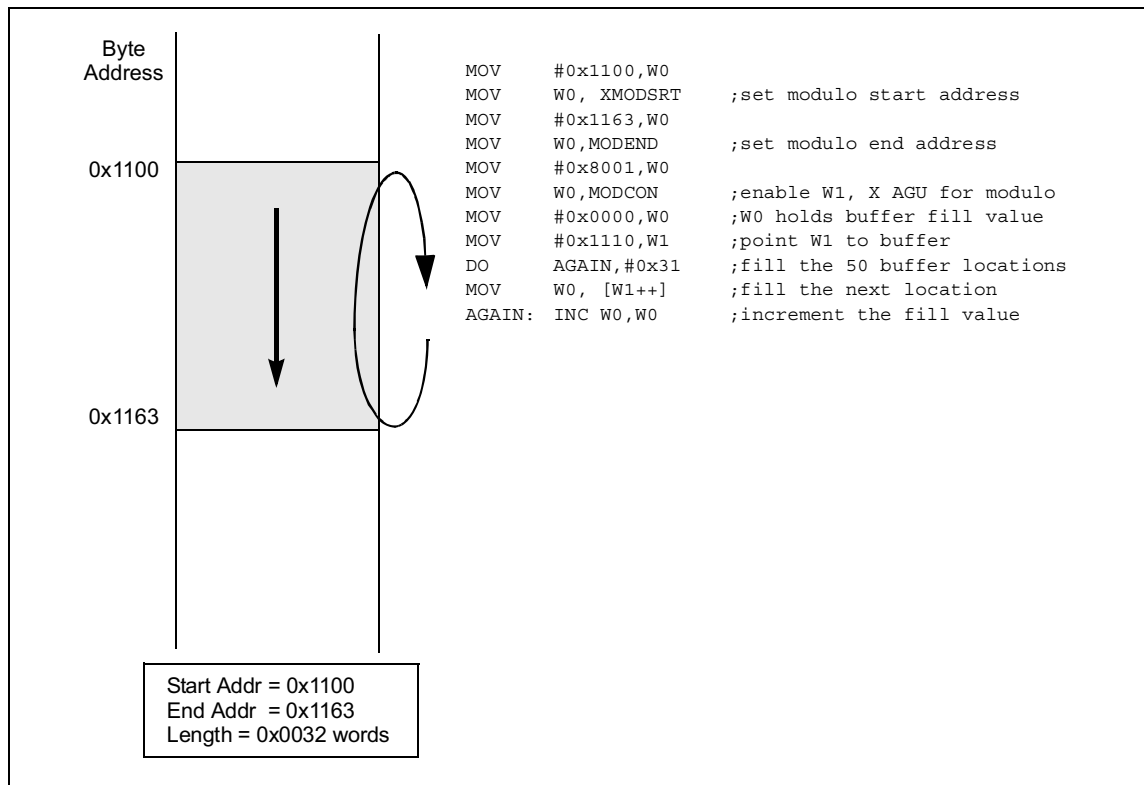
4.2.2 W ADDRESS REGISTER SELECTION

The Modulo and Bit-Reversed Addressing Control register, MODCON<15:0>, contains enable flags as well as a W register field to specify the W Address registers. The XWM and YWM fields select which registers operate with Modulo Addressing. If XWM = 15, X RAGU and X WAGU Modulo Addressing are disabled. Similarly, if YWM = 15, Y AGU Modulo Addressing is disabled.

The X Address Space Pointer W register (XWM), to which Modulo Addressing is to be applied, is stored in MODCON<3:0> (see Table 3-3). Modulo Addressing is enabled for X data space when XWM is set to any value other than 15 and the XMODEN bit is set at MODCON<15>.

The Y Address Space Pointer W register (YWM), to which Modulo Addressing is to be applied, is stored in MODCON<7:4>. Modulo Addressing is enabled for Y data space when YWM is set to any value other than 15 and the YMODEN bit is set at MODCON<14>.

FIGURE 4-1: MODULO ADDRESSING OPERATION EXAMPLE



dsPIC30F4011/4012

4.2.3 MODULO ADDRESSING APPLICABILITY

Modulo Addressing can be applied to the Effective Address (EA) calculation associated with any W register. It is important to realize that the address boundaries check for addresses less than or greater than the upper (for incrementing buffers) and lower (for decrementing buffers) boundary addresses (not just equal to). Address changes may, therefore, jump beyond boundaries and still be adjusted correctly.

Note: The modulo corrected Effective Address is written back to the register only when Pre-Modify or Post-Modify Addressing mode is used to compute the Effective Address. When an address offset (e.g., [W7+W2]) is used, Modulo Addressing correction is performed, but the contents of the register remain unchanged.

4.3 Bit-Reversed Addressing

Bit-Reversed Addressing is intended to simplify data reordering for radix-2 FFT algorithms. It is supported by the X AGU for data writes only.

The modifier, which may be a constant value or register contents, is regarded as having its bit order reversed. The address source and destination are kept in normal order. Thus, the only operand requiring reversal is the modifier.

4.3.1 BIT-REVERSED ADDRESSING IMPLEMENTATION

Bit-Reversed Addressing is enabled when:

1. BWM (W register selection) in the MODCON register is any value other than 15 (the stack can not be accessed using Bit-Reversed Addressing) **and**
2. The BREN bit is set in the XBREV register **and**
3. The addressing mode used is Register Indirect with Pre-Increment or Post-Increment.

If the length of a bit-reversed buffer is $M = 2^N$ bytes, then the last 'N' bits of the data buffer start address must be zeros.

$XB<14:0>$ is the Bit-Reversed Addressing modifier, or 'pivot point', which is typically a constant. In the case of an FFT computation, its value is equal to half of the FFT data buffer size.

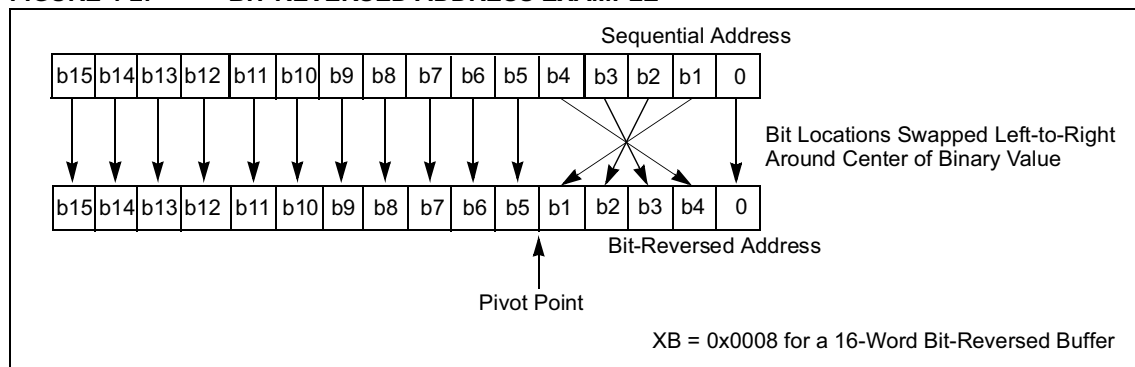
Note: All bit-reversed EA calculations assume word-sized data (LSb of every EA is always clear). The XB value is scaled accordingly to generate compatible (byte) addresses.

When enabled, Bit-Reversed Addressing is only executed for Register Indirect with Pre-Increment or Post-Increment Addressing mode and word-sized data writes. It will not function for any other addressing mode or for byte-sized data, and normal addresses will be generated instead. When Bit-Reversed Addressing is active, the W Address Pointer will always be added to the address modifier (XB) and the offset associated with the Register Indirect Addressing mode will be ignored. In addition, as word-sized data is a requirement, the LSb of the EA is ignored (and always clear).

Note: Modulo Addressing and Bit-Reversed Addressing should not be enabled together. In the event that the user attempts to do this, Bit-Reversed Addressing will assume priority when active for the X WAGU, and X WAGU Modulo Addressing will be disabled. However, Modulo Addressing will continue to function in the X RAGU.

If Bit-Reversed Addressing has already been enabled by setting the BREN (XBREV<15>) bit, then a write to the XBREV register should not be immediately followed by an indirect read operation using the W register that has been designated as the Bit-Reversed Pointer.

FIGURE 4-2: BIT-REVERSED ADDRESS EXAMPLE



dsPIC30F4011/4012

TABLE 4-2: BIT-REVERSED ADDRESS SEQUENCE (16-ENTRY)

| Normal Address | | | | | Bit-Reversed Address | | | | |
|----------------|----|----|----|---------|----------------------|----|----|----|---------|
| A3 | A2 | A1 | A0 | Decimal | A3 | A2 | A1 | A0 | Decimal |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 8 |
| 0 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 0 | 4 |
| 0 | 0 | 1 | 1 | 3 | 1 | 1 | 0 | 0 | 12 |
| 0 | 1 | 0 | 0 | 4 | 0 | 0 | 1 | 0 | 2 |
| 0 | 1 | 0 | 1 | 5 | 1 | 0 | 1 | 0 | 10 |
| 0 | 1 | 1 | 0 | 6 | 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 | 1 | 1 | 1 | 0 | 14 |
| 1 | 0 | 0 | 0 | 8 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 9 | 1 | 0 | 0 | 1 | 9 |
| 1 | 0 | 1 | 0 | 10 | 0 | 1 | 0 | 1 | 5 |
| 1 | 0 | 1 | 1 | 11 | 1 | 1 | 0 | 1 | 13 |
| 1 | 1 | 0 | 0 | 12 | 0 | 0 | 1 | 1 | 3 |
| 1 | 1 | 0 | 1 | 13 | 1 | 0 | 1 | 1 | 11 |
| 1 | 1 | 1 | 0 | 14 | 0 | 1 | 1 | 1 | 7 |
| 1 | 1 | 1 | 1 | 15 | 1 | 1 | 1 | 1 | 15 |

TABLE 4-3: BIT-REVERSED ADDRESS MODIFIER VALUES FOR XBREV REGISTER

| Buffer Size (Words) | XB<14:0> Bit-Reversed Address Modifier Value* |
|---------------------|---|
| 32768 | 0x4000 |
| 16384 | 0x2000 |
| 8192 | 0x1000 |
| 4096 | 0x0800 |
| 2048 | 0x0400 |
| 1024 | 0x0200 |
| 512 | 0x0100 |
| 256 | 0x0080 |
| 128 | 0x0040 |
| 64 | 0x0020 |
| 32 | 0x0010 |
| 16 | 0x0008 |
| 8 | 0x0004 |
| 4 | 0x0002 |
| 2 | 0x0001 |

*Modifier values for buffer sizes greater than 1024 words will exceed the available data memory on the dsPIC30F4011/4012 devices.

dsPIC30F4011/4012

NOTES:

5.0 INTERRUPTS

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the "dsPIC30F Family Reference Manual" (DS70046). For more information on the device instruction set and programming, refer to the "dsPIC30F/33F Programmer's Reference Manual" (DS70157).

The dsPIC30F4011/4012 has 30 interrupt sources and 4 processor exceptions (traps), which must be arbitrated based on a priority scheme.

The CPU is responsible for reading the Interrupt Vector Table (IVT) and transferring the address contained in the interrupt vector to the program counter. The interrupt vector is transferred from the program data bus into the program counter via a 24-bit wide multiplexer on the input of the program counter.

The Interrupt Vector Table (IVT) and Alternate Interrupt Vector Table (AIVT) are placed near the beginning of program memory (0x000004). The IVT and AIVT are shown in Figure 5-1.

The interrupt controller is responsible for pre-processing the interrupts and processor exceptions, prior to their being presented to the processor core. The peripheral interrupts and traps are enabled, prioritized and controlled using centralized Special Function Registers:

- IFS0<15:0>, IFS1<15:0>, IFS2<15:0>
All interrupt request flags are maintained in these three registers. The flags are set by their respective peripherals or external signals, and they are cleared via software.
- IEC0<15:0>, IEC1<15:0>, IEC2<15:0>
All interrupt enable control bits are maintained in these three registers. These control bits are used to individually enable interrupts from the peripherals or external signals.
- IPC0<15:0>... IPC11<7:0>
The user-assignable priority level associated with each of these interrupts is held centrally in these twelve registers.
- IPL<3:0>
The current CPU priority level is explicitly stored in the IPL bits. IPL<3> is present in the CORCON register, whereas IPL<2:0> are present in the STATUS register (SR) in the processor core.
- INTCON1<15:0>, INTCON2<15:0>
Global interrupt control functions are derived from these two registers. INTCON1 contains the control and status flags for the processor exceptions. The INTCON2 register controls the external interrupt request signal behavior and the use of the AIVT.

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

All interrupt sources can be user-assigned to one of seven priority levels, 1 through 7, via the IPCx registers. Each interrupt source is associated with an interrupt vector, as shown in Table 5-1. Levels 7 and 1 represent the highest and lowest maskable priorities, respectively.

Note: Assigning a priority level of 0 to an interrupt source is equivalent to disabling that interrupt.

If the NSTDIS bit (INTCON1<15>) is set, nesting of interrupts is prevented. Thus, if an interrupt is currently being serviced, processing of a new interrupt is prevented, even if the new interrupt is of higher priority than the one currently being serviced.

Note: The IPL bits become read-only whenever the NSTDIS bit has been set to '1'.

Certain interrupts have specialized control bits for features like edge or level triggered interrupts, interrupt-on-change, etc. Control of these features remains within the peripheral module which generates the interrupt.

The DISI instruction can be used to disable the processing of interrupts of priorities 6 and lower for a certain number of instructions, during which the DISI bit (INTCON2<14>) remains set.

When an interrupt is serviced, the PC is loaded with the address stored in the vector location in program memory that corresponds to the interrupt. There are 63 different vectors within the IVT (refer to Figure 5-2). These vectors are contained in locations 0x000004 through 0x0000FE of program memory (refer to Figure 5-2). These locations contain 24-bit addresses, and in order to preserve robustness, an address error trap will take place should the PC attempt to fetch any of these words during normal execution. This prevents execution of random data as a result of accidentally decrementing a PC into vector space, accidentally mapping a data space address into vector space or the PC rolling over to 0x000000 after reaching the end of implemented program memory space. Execution of a GOTO instruction to this vector space will also generate an address error trap.

dsPIC30F4011/4012

5.1 Interrupt Priority

The user-assignable Interrupt Priority (IP<2:0>) bits for each individual interrupt source are located in the Least Significant 3 bits of each nibble within the IPCx register(s). Bit 3 of each nibble is not used and is read as a '0'. These bits define the priority level assigned to a particular interrupt by the user.

Note: The user-selectable priority levels start at 0 as the lowest priority, and level 7 as the highest priority.

Since more than one interrupt request source may be assigned to a specific user-specified priority level, a means is provided to assign priority within a given level. This method is called "Natural Order Priority".

Natural order priority is determined by the position of an interrupt in the vector table, and only affects interrupt operation when multiple interrupts with the same user-assigned priority become pending at the same time.

Table 5-1 lists the interrupt numbers and interrupt sources for the dsPIC DSCs and their associated vector numbers.

Note 1: The natural order priority scheme has 0 as the highest priority and 53 as the lowest priority.

2: The natural order priority number is the same as the INT number.

The ability for the user to assign every interrupt to one of seven priority levels implies that the user can assign a very high overall priority level to an interrupt with a low natural order priority. For example, the PLVD (Low-Voltage Detect) can be given a priority of 7. The INT0 (External Interrupt 0) may be assigned to priority level 1, thus giving it a very low effective priority.

TABLE 5-1: INTERRUPT VECTOR TABLE

| INT Number | Vector Number | Interrupt Source |
|--------------------------------|---------------|--|
| Highest Natural Order Priority | | |
| 0 | 8 | INT0 – External Interrupt 0 |
| 1 | 9 | IC1 – Input Capture 1 |
| 2 | 10 | OC1 – Output Compare 1 |
| 3 | 11 | T1 – Timer 1 |
| 4 | 12 | IC2 – Input Capture 2 |
| 5 | 13 | OC2 – Output Compare 2 |
| 6 | 14 | T2 – Timer 2 |
| 7 | 15 | T3 – Timer 3 |
| 8 | 16 | SPI1 |
| 9 | 17 | U1RX – UART1 Receiver |
| 10 | 18 | U1TX – UART1 Transmitter |
| 11 | 19 | ADC – ADC Convert Done |
| 12 | 20 | NVM – NVM Write Complete |
| 13 | 21 | SI2C – I ² C™ Slave Interrupt |
| 14 | 22 | MI2C – I ² C Master Interrupt |
| 15 | 23 | Input Change Interrupt |
| 16 | 24 | INT1 – External Interrupt 1 |
| 17 | 25 | IC7 – Input Capture 7 |
| 18 | 26 | IC8 – Input Capture 8 |
| 19 | 27 | OC3 – Output Compare 3 |
| 20 | 28 | OC4 – Output Compare 4 |
| 21 | 29 | T4 – Timer4 |
| 22 | 30 | T5 – Timer5 |
| 23 | 31 | INT2 – External Interrupt 2 |
| 24 | 32 | U2RX – UART2 Receiver |
| 25 | 33 | U2TX – UART2 Transmitter |
| 26 | 34 | Reserved |
| 27 | 35 | C1 – Combined IRQ for CAN1 |
| 28 | 36 | Reserved |
| 29 | 37 | Reserved |
| 30 | 38 | Reserved |
| 31 | 39 | Reserved |
| 32 | 40 | Reserved |
| 33 | 41 | Reserved |
| 34 | 42 | Reserved |
| 35 | 43 | Reserved |
| 36 | 44 | Reserved |
| 37 | 45 | Reserved |
| 38 | 46 | Reserved |
| 39 | 47 | PWM – PWM Period Match |
| 40 | 48 | QE1 – QE1 Interrupt |
| 41 | 49 | Reserved |
| 42 | 50 | Reserved |
| 43 | 51 | FLTA – PWM Fault A |
| 44 | 52 | Reserved |
| 45-53 | 53-61 | Reserved |
| Lowest Natural Order Priority | | |

5.2 Reset Sequence

A Reset is not a true exception, because the interrupt controller is not involved in the Reset process. The processor initializes its registers in response to a Reset, which forces the PC to zero. The processor then begins program execution at location 0x000000. A GOTO instruction is stored in the first program memory location, immediately followed by the address target for the GOTO instruction. The processor executes the GOTO to the specified address and then begins operation at the specified target (start) address.

5.2.1 RESET SOURCES

There are 5 sources of error which will cause a device reset.

- **Watchdog Time-out:**
The watchdog has timed out, indicating that the processor is no longer executing the correct flow of code.
- **Uninitialized W Register Trap:**
An attempt to use an uninitialized W register as an Address Pointer will cause a Reset.
- **Illegal Instruction Trap:**
Attempted execution of any unused opcodes will result in an illegal instruction trap. Note that a fetch of an illegal instruction does not result in an illegal instruction trap if that instruction is flushed prior to execution due to a flow change.
- **Brown-out Reset (BOR):**
A momentary dip in the power supply to the device has been detected which may result in malfunction.
- **Trap Lockout:**
Occurrence of multiple trap conditions simultaneously will cause a Reset.

5.3 Traps

Traps can be considered as non-maskable interrupts, indicating a software or hardware error which adhere to a predefined priority, as shown in Figure 5-1. They are intended to provide the user a means to correct erroneous operation during debug and when operating within the application.

Note: If the user does not intend to take corrective action in the event of a trap error condition, these vectors must be loaded with the address of a default handler that simply contains the RESET instruction. If, on the other hand, one of the vectors containing an invalid address is called, an address error trap is generated.

Note that many of these trap conditions can only be detected when they occur. Consequently, the questionable instruction is allowed to complete prior to trap exception processing. If the user chooses to recover from the error, the result of the erroneous action that caused the trap may have to be corrected.

There are 8 fixed priority levels for traps, Level 8 through Level 15, which means that the IPL3 is always set during processing of a trap.

If the user is not currently executing a trap and he sets the IPL<3:0> bits to a value of '0111' (Level 7), then all interrupts are disabled, but traps can still be processed.

5.3.1 TRAP SOURCES

The following traps are provided with increasing priority. However, since all traps can be nested, priority has little effect.

5.3.1.1 Math Error Trap

The math error trap executes under the following four circumstances:

1. Should an attempt be made to divide by zero, the divide operation will be aborted on a cycle boundary and the trap taken.
2. If enabled, a math error trap will be taken when an arithmetic operation on either accumulator A or B causes an overflow from bit 31 and the accumulator guard bits are not utilized.
3. If enabled, a math error trap will be taken when an arithmetic operation on either accumulator A or B causes a catastrophic overflow from bit 39 and all saturation is disabled.
4. If the shift amount specified in a shift instruction is greater than the maximum allowed shift amount, a trap will occur.

dsPIC30F4011/4012

5.3.1.2 Address Error Trap

This trap is initiated when any of the following circumstances occurs:

1. A misaligned data word access is attempted.
2. A data fetch from an unimplemented data memory location is attempted.
3. A data access of an unimplemented program memory location is attempted.
4. An instruction fetch from vector space is attempted.

Note: In the MAC class of instructions, wherein the data space is split into X and Y data space, unimplemented X space includes all of Y space and unimplemented Y space includes all of X space.

5. Execution of a "BRA #literal" instruction, or a "GOTO #literal" instruction, where literal is an unimplemented program memory address.
6. Executing instructions after modifying the PC to point to unimplemented program memory addresses. The PC may be modified by loading a value into the stack and executing a RETURN instruction.

5.3.1.3 Stack Error Trap

This trap is initiated under the following conditions:

1. The Stack Pointer is loaded with a value which is greater than the (user-programmable) limit value written into the SPLIM register (stack overflow).
2. The Stack Pointer is loaded with a value which is less than 0x0800 (simple stack underflow).

5.3.1.4 Oscillator Fail Trap

This trap is initiated if the external oscillator fails and operation becomes reliant on an internal RC backup.

5.3.2 HARD AND SOFT TRAPS

It is possible that multiple traps can become active within the same cycle (e.g., a misaligned word stack write to an overflowed address). In such a case, the fixed priority shown in Figure 5-2 is implemented, which may require the user to check if other traps are pending in order to completely correct the Fault.

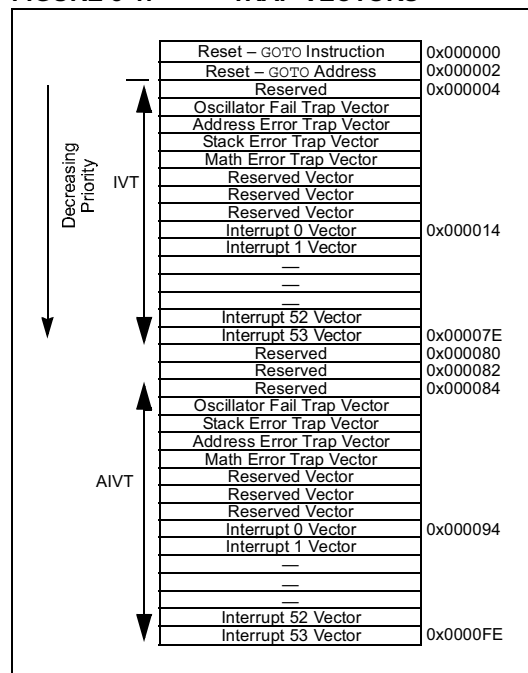
'Soft' traps include exceptions of priority level 8 through level 11, inclusive. The arithmetic error trap (level 11) falls into this category of traps.

'Hard' traps include exceptions of priority level 12 through level 15, inclusive. The address error (level 12), stack error (level 13) and oscillator error (level 14) traps fall into this category.

Each hard trap that occurs must be acknowledged before code execution of any type may continue. If a lower priority hard trap occurs while a higher priority trap is pending, acknowledged or is being processed, a hard trap conflict will occur.

The device is automatically Reset in a hard trap conflict condition. The TRAPR status bit (RCON<15>) is set when the Reset occurs, so that the condition may be detected in software.

FIGURE 5-1: TRAP VECTORS



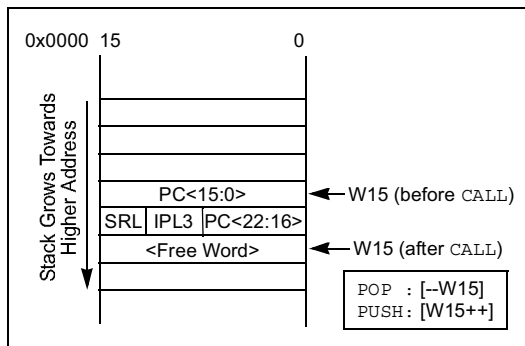
5.4 Interrupt Sequence

All interrupt event flags are sampled in the beginning of each instruction cycle by the IFSx registers. A pending interrupt request (IRQ) is indicated by the flag bit being equal to a '1' in an IFSx register. The IRQ will cause an interrupt to occur if the corresponding bit in the interrupt enable (IECx) register is set. For the remainder of the instruction cycle, the priorities of all pending interrupt requests are evaluated.

If there is a pending IRQ with a priority level greater than the current processor priority level in the IPL bits, the processor will be interrupted.

The processor then stacks the current program counter and the low byte of the processor STATUS register (SRL), as shown in Figure 5-2. The low byte of the STATUS register contains the processor priority level at the time prior to the beginning of the interrupt cycle. The processor then loads the priority level for this interrupt into the STATUS register. This action will disable all lower priority interrupts until the completion of the Interrupt Service Routine.

FIGURE 5-2: INTERRUPT STACK FRAME



- Note 1:** The user can always lower the priority level by writing a new value into SR. The Interrupt Service Routine must clear the interrupt flag bits in the IFSx register before lowering the processor interrupt priority in order to avoid recursive interrupts.
- 2:** The IPL3 bit (CORCON<3>) is always clear when interrupts are being processed. It is set only during execution of traps.

The RETFIE (return from interrupt) instruction will unstack the program counter and STATUS registers to return the processor to its state prior to the interrupt sequence.

5.5 Alternate Interrupt Vector Table

In program memory, the Interrupt Vector Table (IVT) is followed by the Alternate Interrupt Vector Table (AIVT), as shown in Figure 5-1. Access to the Alternate Interrupt Vector Table is provided by the ALTIPT bit in the INTCON2 register. If the ALTIPT bit is set, all interrupt and exception processes will use the alternate vectors instead of the default vectors. The alternate vectors are organized in the same manner as the default vectors. The AIVT supports emulation and debugging efforts by providing a means to switch between an application and a support environment, without requiring the interrupt vectors to be reprogrammed. This feature also enables switching between applications for evaluation of different software algorithms at run time.

If the AIVT is not required, the program memory allocated to the AIVT may be used for other purposes. AIVT is not a protected section and may be freely programmed by the user.

5.6 Fast Context Saving

A context saving option is available using shadow registers. Shadow registers are provided for the DC, N, OV, Z and C bits in SR, and the registers W0 through W3. The shadows are only one-level deep. The shadow registers are accessible using the PUSH.S and POP.S instructions only.

When the processor vectors to an interrupt, the PUSH.S instruction can be used to store the current value of the aforementioned registers into their respective shadow registers.

If an ISR of a certain priority uses the PUSH.S and POP.S instructions for fast context saving, then a higher priority ISR should not include the same instructions. Users must save the key registers in software during a lower priority interrupt if the higher priority ISR uses fast context saving.

5.7 External Interrupt Requests

The interrupt controller supports three external interrupt request signals, INT0-INT2. These inputs are edge sensitive; they require a low-to-high, or a high-to-low transition, to generate an interrupt request. The INTCON2 register has three bits, INT0EP-INT2EP, that select the polarity of the edge detection circuitry.

5.8 Wake-up from Sleep and Idle

The interrupt controller may be used to wake-up the processor from either Sleep or Idle modes if Sleep or Idle modes are active when the interrupt is generated.

If an enabled interrupt request of sufficient priority is received by the interrupt controller, then the standard interrupt request is presented to the processor. At the same time, the processor will wake-up from Sleep or Idle and begin execution of the Interrupt Service Routine (ISR) needed to process the interrupt request.

dsPIC30F4011/4012

TABLE 5-2: INTERRUPT CONTROLLER REGISTER MAP

| SFR Name | ADR | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|------|--------|--------|-------------|--------|--------|-------------|--------|--------|--------|-------|-------------|---------|---------|--------|-------------|--------|---------------------|
| INTCON1 | 0080 | NSTDIS | — | — | — | — | OVATE | OVATE | COVTE | — | — | — | MATHERR | ADDRERR | STKERR | OSCFALL | — | 0000 0000 0000 0000 |
| INTCON2 | 0082 | ALTVT | DISI | — | — | — | — | — | — | — | — | — | — | — | INTZEP | INT1EP | INT0EP | 0000 0000 0000 0000 |
| IFS0 | 0084 | CNIF | M2CIF | S2CIF | NVMIF | ADIF | U1TXIF | U1RXIF | SPI1IF | T3IF | T2IF | OC2IF | IC2IF | T1IF | OC1IF | IC1IF | INT0IF | 0000 0000 0000 0000 |
| IFS1 | 0086 | — | — | — | — | C1IF | — | — | QE1IF | INT2IF | T5IF | T4IF | OC4IF | OC3IF | IC8IF | IC7IF | INT1IF | 0000 0000 0000 0000 |
| IFS2 | 0088 | — | — | — | — | FLTAIF | — | — | — | PWMIF | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| IEC0 | 008C | CNIE | M2CIE | S2CIE | NVMIE | ADIE | U1TXIE | U1RXIE | SPI1IE | T3IE | T2IE | OC2IE | IC2IE | T1IE | OC1IE | IC1IE | INT0IE | 0000 0000 0000 0000 |
| IEC1 | 008E | — | — | — | — | C1IE | — | — | — | INT2IE | T5IE | T4IE | OC4IE | OC3IE | IC8IE | IC7IE | INT1IE | 0000 0000 0000 0000 |
| IEC2 | 0090 | — | — | — | — | FLTAIE | — | — | QE1IE | PWMIE | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| IPC0 | 0094 | — | — | T1IP<2:0> | — | — | OC1IP<2:0> | — | — | — | — | IC1IP<2:0> | — | — | — | INT0IP<2:0> | — | 0100 0100 0100 0100 |
| IPC1 | 0096 | — | — | T3IP<2:0> | — | — | T2IP<2:0> | — | — | — | — | OC2IP<2:0> | — | — | — | IC2IP<2:0> | — | 0100 0100 0100 0100 |
| IPC2 | 0098 | — | — | ADIP<2:0> | — | — | U1TXIP<2:0> | — | — | — | — | U1RXIP<2:0> | — | — | — | SP1IP<2:0> | — | 0100 0100 0100 0100 |
| IPC3 | 009A | — | — | CNIP<2:0> | — | — | M2CIP<2:0> | — | — | — | — | S2CIP<2:0> | — | — | — | NVMIP<2:0> | — | 0100 0100 0100 0100 |
| IPC4 | 009C | — | — | OC3IP<2:0> | — | — | IC8IP<2:0> | — | — | — | — | IC7IP<2:0> | — | — | — | INT1IP<2:0> | — | 0100 0100 0100 0100 |
| IPC5 | 009E | — | — | INT2IP<2:0> | — | — | T5IP<2:0> | — | — | — | — | T4IP<2:0> | — | — | — | OC4IP<2:0> | — | 0100 0100 0100 0100 |
| IPC6 | 00A0 | — | — | C1IP<2:0> | — | — | — | — | — | — | — | U2TXIP<2:0> | — | — | — | U2RXIP<2:0> | — | 0100 0000 0100 0100 |
| IPC7 | 00A2 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| IPC8 | 00A4 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| IPC9 | 00A6 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0100 0000 0100 0100 |
| IPC10 | 00A8 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0100 0000 0000 0100 |
| IPC11 | 00AA | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |

Legend: $\bar{1}$ = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

6.0 FLASH PROGRAM MEMORY

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

The dsPIC30F family of devices contains internal program Flash memory for executing user code. There are two methods by which the user can program this memory:

1. In-Circuit Serial Programming™ (ICSP™)
2. Run-Time Self-Programming (RTSP)

6.1 In-Circuit Serial Programming (ICSP)

dsPIC30F devices can be serially programmed while in the end application circuit. This is simply done with two lines for Programming Clock and Programming Data (which are named PGC and PGD, respectively), and three other lines for Power (VDD), Ground (VSS) and Master Clear (MCLR). This allows customers to manufacture boards with unprogrammed devices, and then program the digital signal controller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

6.2 Run-Time Self-Programming (RTSP)

RTSP is accomplished using TBLRD (table read) and TBLWT (table write) instructions.

With RTSP, the user may erase program memory, 32 instructions (96 bytes) at a time, and can write program memory data, 32 instructions (96 bytes) at a time.

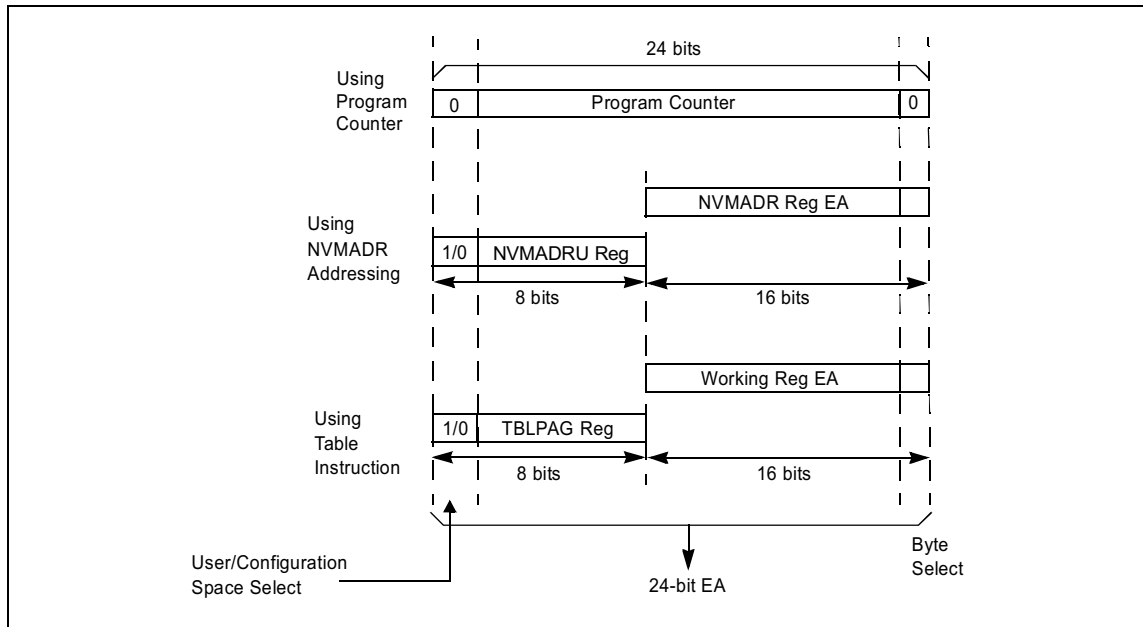
6.3 Table Instruction Operation Summary

The TBLRDL and the TBLWTL instructions are used to read or write to bits <15:0> of program memory. TBLRDL and TBLWTL can access program memory in Word or Byte mode.

The TBLRDH and TBLWTH instructions are used to read or write to bits<23:16> of program memory. TBLRDH and TBLWTH can access program memory in Word or Byte mode.

A 24-bit program memory address is formed using bits<7:0> of the TBLPAG register and the Effective Address (EA) from a W register, specified in the table instruction, as shown in Figure 6-1.

FIGURE 6-1: ADDRESSING FOR TABLE AND NVM REGISTERS



dsPIC30F4011/4012

6.4 RTSP Operation

The dsPIC30F Flash program memory is organized into rows and panels. Each row consists of 32 instructions or 96 bytes. Each panel consists of 128 rows or 4K x 24 instructions. RTSP allows the user to erase one row (32 instructions) at a time and to program 32 instructions at one time.

Each panel of program memory contains write latches that hold 32 instructions of programming data. Prior to the actual programming operation, the write data must be loaded into the panel write latches. The data to be programmed into the panel is loaded in sequential order into the write latches: instruction 0, instruction 1, etc. The addresses loaded must always be from a 32 address boundary.

The basic sequence for RTSP programming is to set up a Table Pointer, then do a series of TBLWT instructions to load the write latches. Programming is performed by setting the special bits in the NVMCON register. 32 TBLWTL and 32 TBLWTH instructions are required to load the 32 instructions.

All of the table write operations are single-word writes (2 instruction cycles) because only the table latches are written.

After the latches are written, a programming operation needs to be initiated to program the data.

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

6.5 RTSP Control Registers

The four SFRs used to read and write the program Flash memory are:

- NVMCON
- NVMADR
- NVMADRU
- NVMKEY

6.5.1 NVMCON REGISTER

The NVMCON register controls which blocks are to be erased, which memory type is to be programmed and the start of the programming cycle.

6.5.2 NVMADR REGISTER

The NVMADR register is used to hold the lower two bytes of the Effective Address. The NVMADR register captures the EA<15:0> of the last table instruction that has been executed and selects the row to write.

6.5.3 NVMADRU REGISTER

The NVMADRU register is used to hold the upper byte of the Effective Address. The NVMADRU register captures the EA<23:16> of the last table instruction that has been executed.

6.5.4 NVMKEY REGISTER

NVMKEY is a write-only register that is used for write protection. To start a programming or an erase sequence, the user must consecutively write 0x55 and 0xAA to the NVMKEY register. Refer to **Section 6.6 “Programming Operations”** for further details.

Note: The user can also directly write to the NVMADR and NVMADRU registers to specify a program memory address for erasing or programming.

6.6 Programming Operations

A complete programming sequence is necessary for programming or erasing the internal Flash in RTSP mode. A programming operation is nominally 2 msec in duration and the processor stalls (waits) until the operation is finished. Setting the WR bit (NVMCON<15>) starts the operation, and the WR bit is automatically cleared when the operation is finished.

6.6.1 PROGRAMMING ALGORITHM FOR PROGRAM FLASH

The user can erase or program one row of program Flash memory at a time. The general process is:

1. Read one row of program Flash (32 instruction words) and store into data RAM as a data "image".
2. Update the data image with the desired new data.
3. Erase program Flash row.
 - a) Set up NVMCON register for multi-word, program Flash, erase and set WREN bit.
 - b) Write address of row to be erased into NVMADRU/NVMADR.
 - c) Write '55' to NVMKEY.
 - d) Write 'AA' to NVMKEY.
 - e) Set the WR bit. This will begin erase cycle.
 - f) CPU will stall for the duration of the erase cycle.
 - g) The WR bit is cleared when erase cycle ends.

4. Write 32 instruction words of data from data RAM "image" into the program Flash write latches.
5. Program 32 instruction words into program Flash.
 - a) Setup NVMCON register for multi-word, program Flash, program and set WREN bit.
 - b) Write '55' to NVMKEY.
 - c) Write 'AA' to NVMKEY.
 - d) Set the WR bit. This will begin program cycle.
 - e) CPU will stall for duration of the program cycle.
 - f) The WR bit is cleared by the hardware when program cycle ends.
6. Repeat steps 1 through 5 as needed to program desired amount of program Flash memory.

6.6.2 ERASING A ROW OF PROGRAM MEMORY

Example 6-1 shows a code sequence that can be used to erase a row (32 instructions) of program memory.

EXAMPLE 6-1: ERASING A ROW OF PROGRAM MEMORY

```
; Setup NVMCON for erase operation, multi word write
; program memory selected, and writes enabled
    MOV    #0x4041,W0
    MOV    W0,NVMCON
; Init pointer to row to be ERASED
    MOV    #tblpage(PROG_ADDR),W0
    MOV    W0,NVMADRU
    MOV    #tbloffset(PROG_ADDR),W0
    MOV    W0,NVMADR
    DISI   #5
; for next 5 instructions

    MOV    #0x55,W0
    MOV    W0,NVMKEY
    MOV    #0xAA,W1
    MOV    W1,NVMKEY
    BSET   NVMCON,#WR
    NOP
    NOP
; Write the 0x55 key
; Write the 0xAA key
; Start the erase sequence
; Insert two NOPs after the erase
; command is asserted
```

dsPIC30F4011/4012

6.6.3 LOADING WRITE LATCHES

Example 6-2 shows a sequence of instructions that can be used to load the 96 bytes of write latches. 32 TBLWTL and 32 TBLWTH instructions are needed to load the write latches selected by the Table Pointer.

EXAMPLE 6-2: LOADING WRITE LATCHES

```
; Set up a pointer to the first program memory location to be written
; program memory selected, and writes enabled
MOV    #0x0000,W0                ;
MOV    W0,TBLPAG                 ; Initialize PM Page Boundary SFR
MOV    #0x6000,W0                ; An example program memory address
; Perform the TBLWT instructions to write the latches
; 0th_program_word
MOV    #LOW_WORD_0,W2            ;
MOV    #HIGH_BYTE_0,W3          ;
TBLWTL W2,[W0]                  ; Write PM low word into program latch
TBLWTH W3,[W0++]                ; Write PM high byte into program latch
; 1st_program_word
MOV    #LOW_WORD_1,W2            ;
MOV    #HIGH_BYTE_1,W3          ;
TBLWTL W2,[W0]                  ; Write PM low word into program latch
TBLWTH W3,[W0++]                ; Write PM high byte into program latch
; 2nd_program_word
MOV    #LOW_WORD_2,W2            ;
MOV    #HIGH_BYTE_2,W3          ;
TBLWTL W2,[W0]                  ; Write PM low word into program latch
TBLWTH W3,[W0++]                ; Write PM high byte into program latch
.
.
; 31st_program_word
MOV    #LOW_WORD_31,W2           ;
MOV    #HIGH_BYTE_31,W3         ;
TBLWTL W2,[W0]                  ; Write PM low word into program latch
TBLWTH W3,[W0++]                ; Write PM high byte into program latch
```

Note: In Example 6-2, the contents of the upper byte of W3 has no effect.

6.6.4 INITIATING THE PROGRAMMING SEQUENCE

For protection, the write initiate sequence for NVMKEY must be used to allow any erase or program operation to proceed. After the programming command has been executed, the user must wait for the programming time until programming is complete. The two instructions following the start of the programming sequence should be NOPS.

EXAMPLE 6-3: INITIATING A PROGRAMMING SEQUENCE

```
DISI   #5                        ; Block all interrupts with priority <7
                                           ; for next 5 instructions
MOV    #0x55,W0                  ;
MOV    W0,NVMKEY                 ; Write the 0x55 key
MOV    #0xAA,W1                  ;
MOV    W1,NVMKEY                 ; Write the 0xAA key
BSET   NVMCON,#WR               ; Start the erase sequence
NOP    ; Insert two NOPS after the erase
NOP    ; command is asserted
```

TABLE 6-1: NVM REGISTER MAP

| File Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | All Resets |
|-----------|-------|--------|--------|--------|--------|--------|--------|-------|--------------|-------|-------|-------|-------|---------------|-------|-------|-------|---------------------|
| NVMCON | 0760 | WR | WREN | WRERR | — | — | — | — | TWRI | — | — | — | — | PROGON<6:0> | — | — | — | 0000 0000 0000 0000 |
| NVMADR | 0762 | — | — | — | — | — | — | — | NVMADR<15:0> | | | | | | | | | 0000 0000 0000 0000 |
| NVMADRU | 0764 | — | — | — | — | — | — | — | — | — | — | — | — | NVMADR<22:16> | — | — | — | 0000 0000 0000 0000 |
| NVMKEY | 0766 | — | — | — | — | — | — | — | — | — | — | — | — | KEY<7:0> | — | — | — | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

7.0 DATA EEPROM MEMORY

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

The data EEPROM memory is readable and writable during normal operation over the entire VDD range. The data EEPROM memory is directly mapped in the program memory address space.

The four SFRs used to read and write the program Flash memory are used to access data EEPROM memory, as well. As described in **Section 6.0 "Flash Program Memory"**, these registers are:

- NVMCON
- NVMADR
- NVMADRU
- NVMKEY

The EEPROM data memory allows read and write of single words and 16-word blocks. When interfacing to data memory, NVMADR, in conjunction with the NVMADRU register, is used to address the EEPROM location being accessed. TBLRD and TBLWTL instructions are used to read and write data EEPROM. The dsPIC30F4011/4012 devices have 1 Kbyte (512 words) of data EEPROM, with an address range from 0x7FFC00 to 0x7FFFE.

A word write operation should be preceded by an erase of the corresponding memory location(s). The write typically requires 2 ms to complete, but the write time will vary with voltage and temperature.

A program or erase operation on the data EEPROM does not stop the instruction flow. The user is responsible for waiting for the appropriate duration of time before initiating another data EEPROM write/erase operation. Attempting to read the data EEPROM while a programming or erase operation is in progress results in unspecified data.

Control bit, WR, initiates write operations, similar to program Flash writes. This bit cannot be cleared, only set, in software. This bit is cleared in hardware at the completion of the write operation. The inability to clear the WR bit in software prevents the accidental or premature termination of a write operation.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a MCLR Reset, or a WDT Time-out Reset, during normal operation. In these situations, following Reset, the user can check the WRERR bit and rewrite the location. The address register, NVMADR, remains unchanged.

Note: Interrupt flag bit, NVMIF in the IFS0 register, is set when write is complete. It must be cleared in software.

7.1 Reading the Data EEPROM

A TBLRD instruction reads a word at the current program word address. This example uses W0 as a pointer to data EEPROM. The result is placed in register W4, as shown in Example 7-1.

EXAMPLE 7-1: DATA EEPROM READ

```
MOV    #LOW_ADDR_WORD,W0    ; Init Pointer
MOV    #HIGH_ADDR_WORD,W1
MOV    W1,TBLPAG
TBLRD  [ W0 ], W4           ; read data EEPROM
```

dsPIC30F4011/4012

7.2 Erasing Data EEPROM

7.2.1 ERASING A BLOCK OF DATA EEPROM

In order to erase a block of data EEPROM, the NVMADRU and NVMADR registers must initially point to the block of memory to be erased. Configure NVMCON for erasing a block of data EEPROM and set the WR and WREN bits in the NVMCON register. Setting the WR bit initiates the erase, as shown in Example 7-2.

EXAMPLE 7-2: DATA EEPROM BLOCK ERASE

```
; Select data EEPROM block, WR, WREN bits
MOV    #0x4045,W0
MOV    W0,NVMCON                ; Initialize NVMCON SFR

; Start erase cycle by setting WR after writing key sequence
DISI   #5                        ; Block all interrupts with priority <7
                                           ; for next 5 instructions

MOV    #0x55,W0                  ;
MOV    W0,NVMKEY                 ; Write the 0x55 key
MOV    #0xAA,W1                  ;
MOV    W1,NVMKEY                 ; Write the 0xAA key
BSET   NVMCON,#WR                ; Initiate erase sequence
NOP
NOP

; Erase cycle will complete in 2mS. CPU is not stalled for the Data Erase Cycle
; User can poll WR bit, use NVMIF or Timer IRQ to determine erasure complete
```

7.2.2 ERASING A WORD OF DATA EEPROM

The TBLPAG and NVMADR registers must point to the block. Select erase a block of data Flash and set the WR and WREN bits in the NVMCON register. Setting the WR bit initiates the erase, as shown in Example 7-3.

EXAMPLE 7-3: DATA EEPROM WORD ERASE

```
; Select data EEPROM word, WR, WREN bits
MOV    #0x4044,W0
MOV    W0,NVMCON

; Start erase cycle by setting WR after writing key sequence
DISI   #5                        ; Block all interrupts with priority <7
                                           ; for next 5 instructions

MOV    #0x55,W0                  ;
MOV    W0,NVMKEY                 ; Write the 0x55 key
MOV    #0xAA,W1                  ;
MOV    W1,NVMKEY                 ; Write the 0xAA key
BSET   NVMCON,#WR                ; Initiate erase sequence
NOP
NOP

; Erase cycle will complete in 2mS. CPU is not stalled for the Data Erase Cycle
; User can poll WR bit, use NVMIF or Timer IRQ to determine erasure complete
```

7.3 Writing to the Data EEPROM

To write an EEPROM data location, the following sequence must be followed:

1. Erase data EEPROM word.
 - a) Select word, data EEPROM; erase and set WREN bit in NVMCON register.
 - b) Write address of word to be erased into NVMADRU/NVMADR.
 - c) Enable NVM interrupt (optional).
 - d) Write '55' to NVMKEY.
 - e) Write 'AA' to NVMKEY.
 - f) Set the WR bit. This will begin erase cycle.
 - g) Either poll NVMIF bit or wait for NVMIF interrupt.
 - h) The WR bit is cleared when the erase cycle ends.
2. Write data word into data EEPROM write latches.
3. Program 1 data word into data EEPROM.
 - a) Select word, data EEPROM; program and set WREN bit in NVMCON register.
 - b) Enable NVM write done interrupt (optional).
 - c) Write '55' to NVMKEY.
 - d) Write 'AA' to NVMKEY.
 - e) Set the WR bit. This will begin program cycle.
 - f) Either poll NVMIF bit or wait for NVM interrupt.
 - g) The WR bit is cleared when the write cycle ends.

The write will not initiate if the above sequence is not exactly followed (write 0x55 to NVMKEY, write 0xAA to NVMCON, then set WR bit) for each word. It is strongly recommended that interrupts be disabled during this code segment.

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

After a write sequence has been initiated, clearing the WREN bit will not affect the current write cycle. The WR bit will be inhibited from being set unless the WREN bit is set. The WREN bit must be set on a previous instruction. Both WR and WREN cannot be set with the same instruction.

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

7.3.1 WRITING A WORD OF DATA EEPROM

Once the user has erased the word to be programmed, then a table write instruction is used to write one write latch, as shown in Example 7-4.

EXAMPLE 7-4: DATA EEPROM WORD WRITE

```
; Point to data memory
MOV      #LOW_ADDR_WORD,W0          ; Init pointer
MOV      #HIGH_ADDR_WORD,W1
MOV      W1,TBLPAG
MOV      #LOW(WORD),W2              ; Get data
TBLWTL   W2,[ W0]                  ; Write data
; The NVMADR captures last table access address
; Select data EEPROM for 1 word op
MOV      #0x4004,W0
MOV      W0,NVMCON

; Operate key to allow write operation
DISI     #5                          ; Block all interrupts with priority <7
; for next 5 instructions

MOV      #0x55,W0
MOV      W0,NVMKEY                  ; Write the 0x55 key
MOV      #0xAA,W1
MOV      W1,NVMKEY                  ; Write the 0xAA key
BSET     NVMCON,#WR                  ; Initiate program sequence
NOP
NOP
; Write cycle will complete in 2mS. CPU is not stalled for the Data Write Cycle
; User can poll WR bit, use NVMIF or Timer IRQ to determine write complete
```

dsPIC30F4011/4012

7.3.2 WRITING A BLOCK OF DATA EEPROM

To write a block of data EEPROM, write to all sixteen latches first, then set the NVMCON register and program the block.

EXAMPLE 7-5: DATA EEPROM BLOCK WRITE

```
MOV      #LOW_ADDR_WORD,W0 ; Init pointer
MOV      #HIGH_ADDR_WORD,W1
MOV      W1,TBLPAG
MOV      #data1,W2          ; Get 1st data
TBLWTL  W2,[W0]++          ; write data
MOV      #data2,W2          ; Get 2nd data
TBLWTL  W2,[W0]++          ; write data
MOV      #data3,W2          ; Get 3rd data
TBLWTL  W2,[W0]++          ; write data
MOV      #data4,W2          ; Get 4th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data5,W2          ; Get 5th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data6,W2          ; Get 6th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data7,W2          ; Get 7th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data8,W2          ; Get 8th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data9,W2          ; Get 9th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data10,W2         ; Get 10th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data11,W2        ; Get 11th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data12,W2        ; Get 12th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data13,W2        ; Get 13th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data14,W2        ; Get 14th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data15,W2        ; Get 15th data
TBLWTL  W2,[W0]++          ; write data
MOV      #data16,W2        ; Get 16th data
TBLWTL  W2,[W0]++          ; write data. The NVMADR captures last table access address.
MOV      #0x400A,W0        ; Select data EEPROM for multi word op
MOV      W0,NVMCON         ; Operate Key to allow program operation
DISI     #5                ; Block all interrupts with priority <7
                          ; for next 5 instructions

MOV      #0x55,W0
MOV      W0,NVMKEY         ; Write the 0x55 key
MOV      #0xAA,W1
MOV      W1,NVMKEY         ; Write the 0xAA key
BSET     NVMCON,#WR        ; Start write cycle
NOP
NOP
```


7.4 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 Protection Against Spurious Write

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built-in. On power-up, the WREN bit is cleared; also, the Power-up Timer prevents EEPROM write.

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

dsPIC30F4011/4012

NOTES:

8.0 I/O PORTS

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

All of the device pins (except VDD, VSS, MCLR and OSC1/CLKI) are shared between the peripherals and the parallel I/O ports.

All I/O input ports feature Schmitt Trigger inputs for improved noise immunity.

8.1 Parallel I/O (PIO) Ports

When a peripheral is enabled and the peripheral is actively driving an associated pin, the use of the pin as a general purpose output pin is disabled. The I/O pin may be read, but the output driver for the Parallel Port bit will be disabled. If a peripheral is enabled, but the peripheral is not actively driving a pin, that pin may be driven by a port.

All port pins have three registers directly associated with the operation of the port pin. The Data Direction register (TRISx) determines whether the pin is an input or an output. If the Data Direction register bit is a '1', then the pin is an input. All port pins are defined as inputs after a Reset. Reads from the latch (LATx), read the latch. Writes to the latch, write the latch (LATx). Reads from the port (PORTx), read the port pins and writes to the port pins, write the latch (LATx).

Any bit and its associated data and control registers that are not valid for a particular device will be disabled. That means the corresponding LATx and TRISx registers and the port pin will read as zeros.

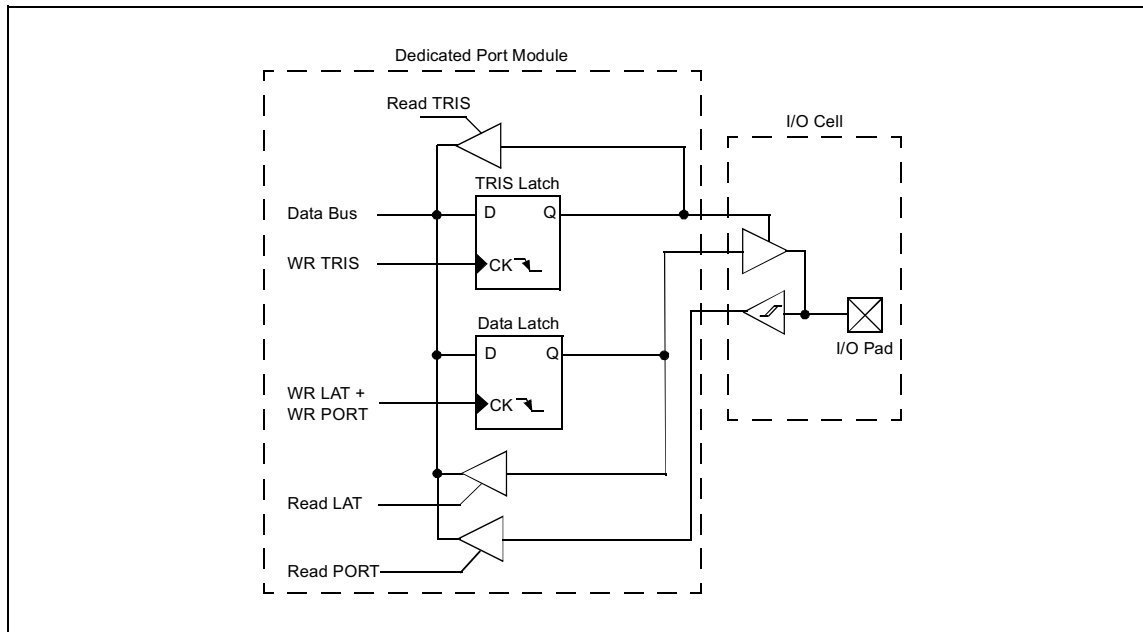
When a pin is shared with another peripheral or function that is defined as an input only, it is nevertheless regarded as a dedicated port because there is no other competing source of outputs. An example is the INT4 pin.

The format of the registers for PORTx are shown in Table 8-1.

The TRISx (Data Direction) register controls the direction of the pins. The LATx register supplies data to the outputs and is readable/writable. Reading the PORTx register yields the state of the input pins, while writing to the PORTx register modifies the contents of the LATx register.

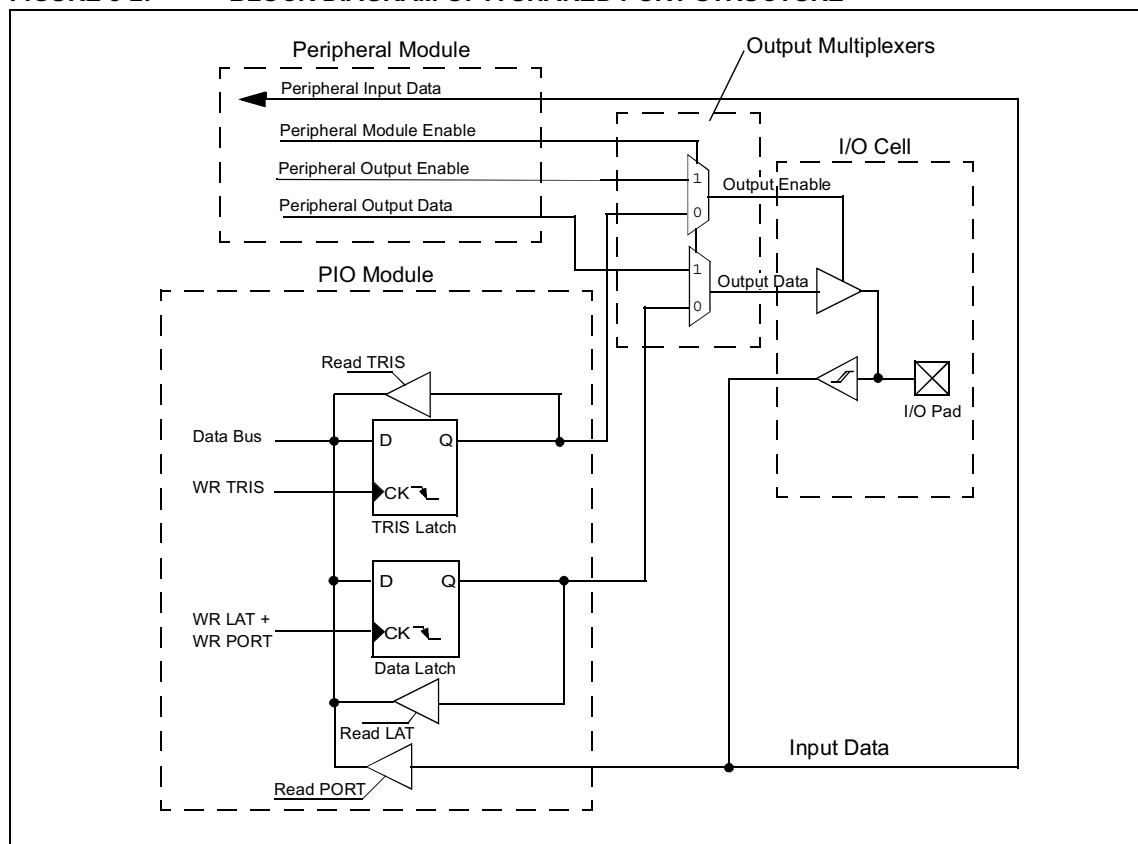
A parallel I/O (PIO) port that shares a pin with a peripheral is, in general, subservient to the peripheral. The peripheral's output buffer data and control signals are provided to a pair of multiplexers. The multiplexers select whether the peripheral or the associated port has ownership of the output data and control signals of the I/O pad cell. Figure 8-2 shows how ports are shared with other peripherals and the associated I/O cell (pad) to which they are connected. Table 8-1 and Table 8-2 show the formats of the registers for the shared ports, PORTB through PORTG.

FIGURE 8-1: BLOCK DIAGRAM OF A DEDICATED PORT STRUCTURE



dsPIC30F4011/4012

FIGURE 8-2: BLOCK DIAGRAM OF A SHARED PORT STRUCTURE



8.2 Configuring Analog Port Pins

The use of the ADPCFG and TRIS registers control the operation of the A/D port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bit set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

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 not convert an analog input. Analog levels on any pin that is defined as a digital input (including the ANx pins), may cause the input buffer to consume current that exceeds the device specifications.

8.2.1 I/O PORT WRITE/READ TIMING

One instruction cycle is required between a port direction change or port write operation and a read operation of the same port. Typically this instruction would be a NOP.

EXAMPLE 8-1: PORT WRITE/READ EXAMPLE

```

MOV  0xFF00, W0 ; Configure PORTB<15:8>
                ; as inputs
MOV  W0, TRISBB ; and PORTB<7:0> as outputs
NOP                    ; Delay 1 cycle
BTSS PORTB, #13 ; Next Instruction
    
```

TABLE 8-1: dsPIC30F4011 PORT REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|---------|---------|---------|--------|--------|--------|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|---------------------|
| TRISB | 02C6 | — | — | — | — | — | — | — | TRISB8 | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 0000 0001 1111 1111 |
| PORTB | 02C8 | — | — | — | — | — | — | — | RB8 | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | 0000 0000 0000 0000 |
| LATB | 02CA | — | — | — | — | — | — | — | LATB8 | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | 0000 0000 0000 0000 |
| TRISC | 02CC | TRISC15 | TRISC14 | TRISC13 | — | — | — | — | — | — | — | — | — | — | — | — | — | 1110 0000 0000 0000 |
| PORTC | 02CE | RC15 | RC14 | RC13 | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| LATC | 02D0 | LATC15 | LATC14 | LATC13 | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| TRISD | 02D2 | — | — | — | — | — | — | — | — | — | — | — | — | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 0000 0000 0000 1111 |
| PORTD | 02D4 | — | — | — | — | — | — | — | — | — | — | — | — | RD3 | RD2 | RD1 | RD0 | 0000 0000 0000 0000 |
| LATD | 02D6 | — | — | — | — | — | — | — | — | — | — | — | — | LATD3 | LATD2 | LATD1 | LATD0 | 0000 0000 0000 0000 |
| TRISE | 02D8 | — | — | — | — | — | — | — | TRISE8 | — | — | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 | 0000 0001 0011 1111 |
| PORTE | 02DA | — | — | — | — | — | — | — | RE8 | — | — | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 | 0000 0000 0000 0000 |
| LATE | 02DC | — | — | — | — | — | — | — | LATE8 | — | — | LATE5 | LATE4 | LATE3 | LATE2 | LATE1 | LATE0 | 0000 0000 0000 0000 |
| TRISF | 02DE | — | — | — | — | — | — | — | — | — | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | TRISF0 | 0000 0000 0111 1111 |
| PORTF | 02E0 | — | — | — | — | — | — | — | — | — | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | RF0 | 0000 0000 0000 0000 |
| LATF | 02E2 | — | — | — | — | — | — | — | — | — | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | LATF0 | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

TABLE 8-2: dsPIC30F4012 PORT REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|---------|---------|---------|--------|--------|--------|--------|-------|-------|-------|--------|--------|--------|--------|--------|---------------------|---------------------|
| TRISB | 02C6 | — | — | — | — | — | — | — | — | — | — | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 0000 0000 0011 1111 |
| PORTB | 02C8 | — | — | — | — | — | — | — | — | — | — | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | 0000 0000 0000 0000 |
| LATB | 02CB | — | — | — | — | — | — | — | — | — | — | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | 0000 0000 0000 0000 |
| TRISC | 02CC | TRISC15 | TRISC14 | TRISC13 | — | — | — | — | — | — | — | — | — | — | — | — | — | 1110 0000 0000 0000 |
| PORTC | 02CE | RC15 | RC14 | RC13 | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| LATC | 02D0 | LATC15 | LATC14 | LATC13 | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| TRISD | 02D2 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | TRISD1 | TRISD0 | 0000 0000 0000 0011 |
| PORTD | 02D4 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | RD1 | RD0 | 0000 0000 0000 0000 |
| LATD | 02D6 | — | — | — | — | — | — | — | — | — | — | — | — | — | LATD1 | LATD0 | 0000 0000 0000 0000 | |
| TRISE | 02D8 | — | — | — | — | — | — | TRISE8 | — | — | — | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 | 0000 0001 0011 1111 |
| PORTE | 02DA | — | — | — | — | — | — | RE8 | — | — | — | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 | 0000 0000 0000 0000 |
| LATE | 02DC | — | — | — | — | — | — | LATE8 | — | — | — | LATE5 | LATE4 | LATE3 | LATE2 | LATE1 | LATE0 | 0000 0000 0000 0000 |
| TRISF | 02EE | — | — | — | — | — | — | — | — | — | — | — | — | TRISF3 | TRISF2 | — | — | 0000 0000 0000 1100 |
| PORTF | 02E0 | — | — | — | — | — | — | — | — | — | — | — | — | RF3 | RF2 | — | — | 0000 0000 0000 0000 |
| LATF | 02E2 | — | — | — | — | — | — | — | — | — | — | — | — | LATF3 | LATF2 | — | — | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

8.3 Input Change Notification Module

The input change notification module provides the dsPIC30F devices the ability to generate interrupt requests to the processor in response to a change of state on selected input pins. This module is capable of detecting input change of states, even in Sleep mode, when the clocks are disabled. There are 10 external signals (CN0 through CN7, CN17 and CN18) that may be selected (enabled) for generating an interrupt request on a change of state.

Please refer to the pin diagrams for CN pin locations.

TABLE 8-3: INPUT CHANGE NOTIFICATION REGISTER MAP (BITS 7-0)

| SFR Name | Addr. | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|--------|--------|--------|--------|--------|----------|----------|--------|---------------------|
| CNEN1 | 00C0 | CN7IE | CN6IE | CN5IE | CN4IE | CN3IE | CN2IE | CN1IE | CN0IE | 0000 0000 0000 0000 |
| CNEN2 | 00C2 | — | — | — | — | — | CN18IE* | CN17IE* | — | 0000 0000 0000 0000 |
| CNPU1 | 00C4 | CN7PUE | CN6PUE | CN5PUE | CN4PUE | CN3PUE | CN2PUE | CN1PUE | CN0PUE | 0000 0000 0000 0000 |
| CNPU2 | 00C6 | — | — | — | — | — | CN18PUE* | CN17PUE* | — | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

* Not available on dsPIC30F4012

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

9.0 TIMER1 MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

This section describes the 16-bit general purpose Timer1 module and associated operational modes. Figure 9-1 depicts the simplified block diagram of the 16-bit Timer1 module.

Note: Timer1 is a 'Type A' timer. Please refer to the specifications for a Type A timer in **Section 24.0 "Electrical Characteristics"** of this document.

The following sections provide a detailed description, including setup and control registers, along with associated block diagrams for the operational modes of the timers.

The Timer1 module is a 16-bit timer which can serve as the time counter for the Real-Time Clock, or operate as a free-running, interval timer/counter. The 16-bit timer has the following modes:

- 16-bit Timer
- 16-bit Synchronous Counter
- 16-bit Asynchronous Counter

Further, the following operational characteristics are supported:

- Timer gate operation
- Selectable prescaler settings
- Timer operation during CPU Idle and Sleep modes
- Interrupt on 16-bit Period register match or falling edge of external gate signal

These operating modes are determined by setting the appropriate bit(s) in the 16-bit SFR, T1CON. Figure 9-1 presents a block diagram of the 16-bit timer module.

16-bit Timer Mode: In the 16-bit Timer mode, the timer increments on every instruction cycle up to a match value, preloaded into the Period register, PR1, then resets to 0 and continues to count.

When the CPU goes into the Idle mode, the timer will stop incrementing unless the TSIDL (T1CON<13>) bit = 0. If TSIDL = 1, the timer module logic will resume the incrementing sequence upon termination of the CPU Idle mode.

16-bit Synchronous Counter Mode: In the 16-bit Synchronous Counter mode, the timer increments on the rising edge of the applied external clock signal, which is synchronized with the internal phase clocks. The timer counts up to a match value preloaded in PR1, then resets to 0 and continues.

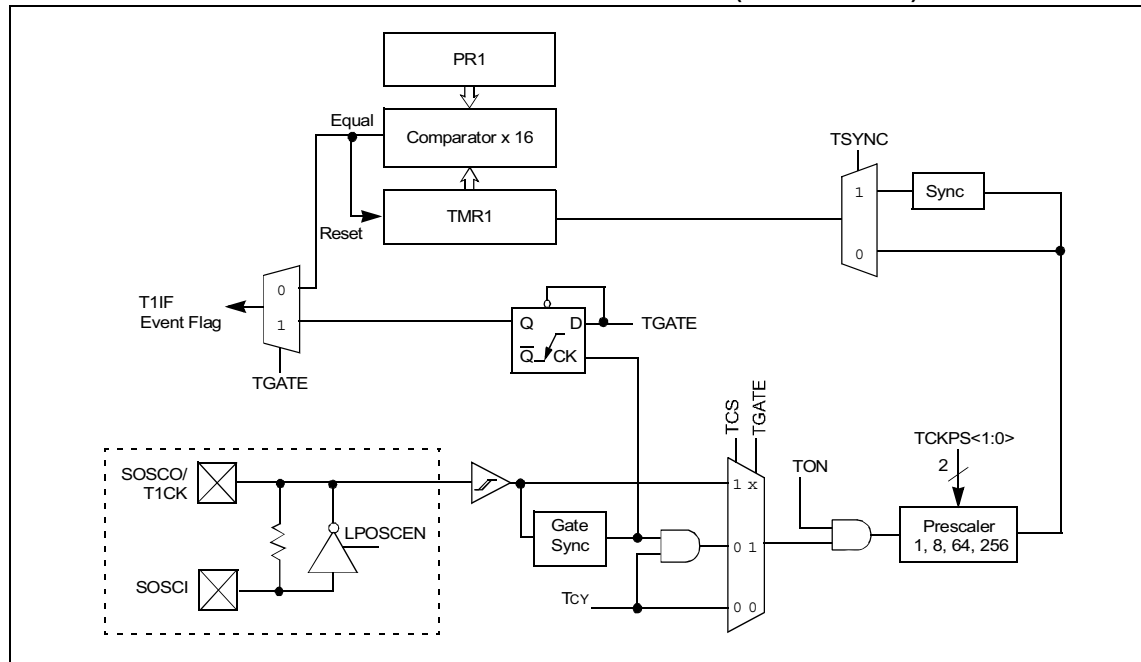
When the CPU goes into the Idle mode, the timer will stop incrementing unless the respective TSIDL bit = 0. If TSIDL = 1, the timer module logic will resume the incrementing sequence upon termination of the CPU Idle mode.

16-bit Asynchronous Counter Mode: In the 16-bit Asynchronous Counter mode, the timer increments on every rising edge of the applied external clock signal. The timer counts up to a match value preloaded in PR1, then resets to 0 and continues.

When the timer is configured for the Asynchronous mode of operation, and the CPU goes into the Idle mode, the timer will stop incrementing if TSIDL = 1.

dsPIC30F4011/4012

FIGURE 9-1: 16-BIT TIMER1 MODULE BLOCK DIAGRAM (TYPE A TIMER)



9.1 Timer Gate Operation

The 16-bit timer can be placed in the Gated Time Accumulation mode. This mode allows the internal TcY to increment the respective timer when the gate input signal (T1CK pin) is asserted high. Control bit, TGATE (T1CON<6>), must be set to enable this mode. The timer must be enabled (TON = 1) and the timer clock source set to internal (TCS = 0).

When the CPU goes into the Idle mode, the timer will stop incrementing unless TSIDL = 0. If TSIDL = 1, the timer will resume the incrementing sequence upon termination of the CPU Idle mode.

9.2 Timer Prescaler

The input clock (Fosc/4 or external clock) to the 16-bit Timer has a prescale option of 1:1, 1:8, 1:64 and 1:256 selected by control bits, TCKPS<1:0> (T1CON<5:4>). The prescaler counter is cleared when any of the following occurs:

- a write to the TMR1 register
- clearing of the TON bit (T1CON<15>)
- device Reset, such as POR and BOR

However, if the timer is disabled (TON = 0), then the timer prescaler cannot be reset since the prescaler clock is halted.

TMR1 is not cleared when T1CON is written. It is cleared by writing to the TMR1 register.

9.3 Timer Operation During Sleep Mode

During CPU Sleep mode, the timer will operate if:

- The timer module is enabled (TON = 1) and
- The timer clock source is selected as external (TCS = 1) and
- The TSYNC bit (T1CON<2>) is asserted to a logic '0', which defines the external clock source as asynchronous

When all three conditions are true, the timer will continue to count up to the Period register and be reset to 0x0000.

When a match between the timer and the Period register occurs, an interrupt can be generated if the respective timer interrupt enable bit is asserted.

9.4 Timer Interrupt

The 16-bit timer has the ability to generate an interrupt on period match. When the timer count matches the Period register, the T1IF bit is asserted and an interrupt will be generated, if enabled. The T1IF bit must be cleared in software. The Timer Interrupt Flag, T1IF, is located in the IFS0 control register in the interrupt controller.

When the Gated Time Accumulation mode is enabled, an interrupt will also be generated on the falling edge of the gate signal (at the end of the accumulation cycle).

Enabling an interrupt is accomplished via the respective Timer Interrupt Enable bit, T1IE. The timer interrupt enable bit is located in the IEC0 Control register in the interrupt controller.

9.5 Real-Time Clock

Timer1, when operating in Real-Time Clock (RTC) mode, provides time-of-day and event time-stamping capabilities. Key operational features of the RTC are:

- Operation from 32 kHz LP oscillator
- 8-bit prescaler
- Low power
- Real-Time Clock interrupts

These operating modes are determined by setting the appropriate bit(s) in the T1CON Control register

9.5.1 RTC OSCILLATOR OPERATION

When $TON = 1$, $TCS = 1$ and $TGATE = 0$, the timer increments on the rising edge of the 32 kHz LP oscillator output signal, up to the value specified in the Period register, and is then reset to '0'.

The TSYNC bit must be asserted to a logic '0' (Asynchronous mode) for correct operation.

Enabling LPOSCEN (OSCCON<1>) will disable the normal Timer and Counter modes and enable a timer carry-out wake-up event.

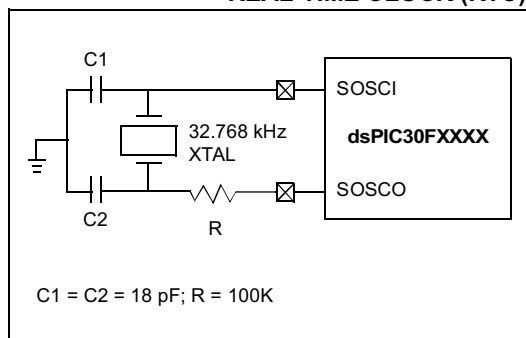
When the CPU enters Sleep mode, the RTC will continue to operate, provided the 32 kHz external crystal oscillator is active and the control bits have not been changed. The TSIDL bit should be cleared to '0' in order for RTC to continue operation in Idle mode.

9.5.2 RTC INTERRUPTS

When an interrupt event occurs, the respective Timer Interrupt Flag, T1IF, is asserted and an interrupt will be generated, if enabled. The T1IF bit must be cleared in software. The respective Timer Interrupt Flag, T1IF, is located in the IFS0 Status register in the interrupt controller.

Enabling an interrupt is accomplished via the respective Timer Interrupt Enable bit, T1IE. The Timer Interrupt Enable bit is located in the IEC0 control register in the interrupt controller.

FIGURE 9-2: RECOMMENDED COMPONENTS FOR TIMER1 LP OSCILLATOR REAL-TIME CLOCK (RTC)



dsPIC30F4011/4012

TABLE 9-1: TIMER1 REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State | |
|----------|-------|-------------------|--------|--------|--------|--------|--------|-------|-------|-------|-------|--------|--------|-------|-------|-------|-------|---------------------|---------------------|
| TMR1 | 0100 | Timer1 Register | | | | | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| PR1 | 0102 | Period Register 1 | | | | | | | | | | | | | | | | | 1111 1111 1111 1111 |
| T1CON | 0104 | TON | — | TSIDL | — | — | — | — | — | — | TGATE | TCKPS1 | TCKPS0 | — | TSYNC | TCS | — | 0000 0000 0000 0000 | |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

10.0 TIMER2/3 MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

This section describes the 32-bit general purpose timer module (Timer2/3) and associated operational modes. Figure 10-1 depicts the simplified block diagram of the 32-bit Timer2/3 module. Figure 10-2 and Figure 10-3 show Timer2/3 configured as two independent 16-bit timers, Timer2 and Timer3, respectively.

Note: Timer2 is a 'Type B' timer and Timer3 is a 'Type C' timer. Please refer to the appropriate timer type in **Section 24.0 "Electrical Characteristics"** of this document.

The Timer2/3 module is a 32-bit timer, which can be configured as two 16-bit timers, with selectable operating modes. These timers are utilized by other peripheral modules, such as:

- Input Capture
- Output Compare/Simple PWM

The following sections provide a detailed description, including setup and control registers, along with associated block diagrams for the operational modes of the timers.

The 32-bit timer has the following modes:

- Two independent 16-bit timers (Timer2 and Timer3) with all 16-bit operating modes (except Asynchronous Counter mode)
- Single 32-bit timer operation
- Single 32-bit synchronous counter

Further, the following operational characteristics are supported:

- ADC Event Trigger
- Timer Gate Operation
- Selectable Prescaler Settings
- Timer Operation During Idle and Sleep Modes
- Interrupt on a 32-bit Period Register Match

These operating modes are determined by setting the appropriate bit(s) in the 16-bit T2CON and T3CON SFRs.

For 32-bit timer/counter operation, Timer2 is the least significant word and Timer3 is the most significant word of the 32-bit timer.

Note: For 32-bit timer operation, T3CON control bits are ignored. Only T2CON control bits are used for setup and control. Timer2 clock and gate inputs are utilized for the 32-bit timer module, but an interrupt is generated with the Timer3 Interrupt Flag (T3IF) and the interrupt is enabled with the Timer3 Interrupt Enable bit (T3IE).

16-bit Mode: In the 16-bit mode, Timer2 and Timer3 can be configured as two independent 16-bit timers. Each timer can be set up in either 16-bit Timer mode or 16-bit Synchronous Counter mode. See **Section 9.0 "Timer1 Module"**, Timer1 Module, for details on these two operating modes.

The only functional difference between Timer2 and Timer3 is that Timer2 provides synchronization of the clock prescaler output. This is useful for high-frequency external clock inputs.

32-bit Timer Mode: In the 32-bit Timer mode, the timer increments on every instruction cycle up to a match value, preloaded into the combined 32-bit Period register, PR3/PR2, then resets to 0 and continues to count.

For synchronous 32-bit reads of the Timer2/Timer3 pair, reading the least significant word (TMR2 register) will cause the msw to be read and latched into a 16-bit holding register, termed TMR3HLD.

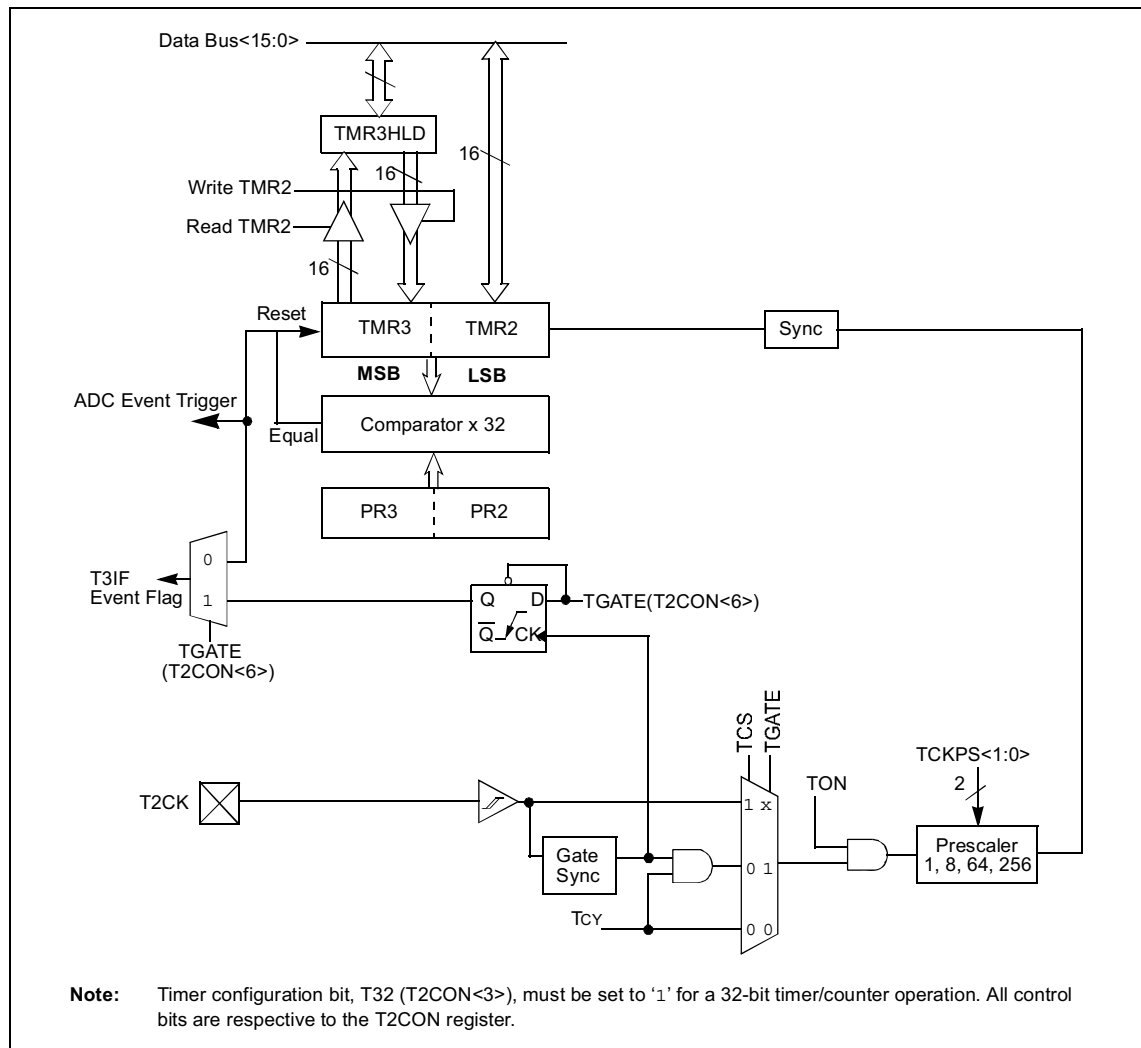
For synchronous 32-bit writes, the holding register (TMR3HLD) must first be written to. When followed by a write to the TMR2 register, the contents of TMR3HLD will be transferred and latched into the MSB of the 32-bit timer (TMR3).

32-bit Synchronous Counter Mode: In the 32-bit Synchronous Counter mode, the timer increments on the rising edge of the applied external clock signal, which is synchronized with the internal phase clocks. The timer counts up to a match value preloaded in the combined 32-bit period register, PR3/PR2, then resets to 0 and continues.

When the timer is configured for the Synchronous Counter mode of operation and the CPU goes into the Idle mode, the timer will stop incrementing unless the TSIDL (T2CON<13>) bit = 0. If TSIDL = 1, the timer module logic will resume the incrementing sequence upon termination of the CPU Idle mode.

dsPIC30F4011/4012

FIGURE 10-1: 32-BIT TIMER2/3 BLOCK DIAGRAM



dsPIC30F4011/4012

FIGURE 10-2: 16-BIT TIMER2 BLOCK DIAGRAM

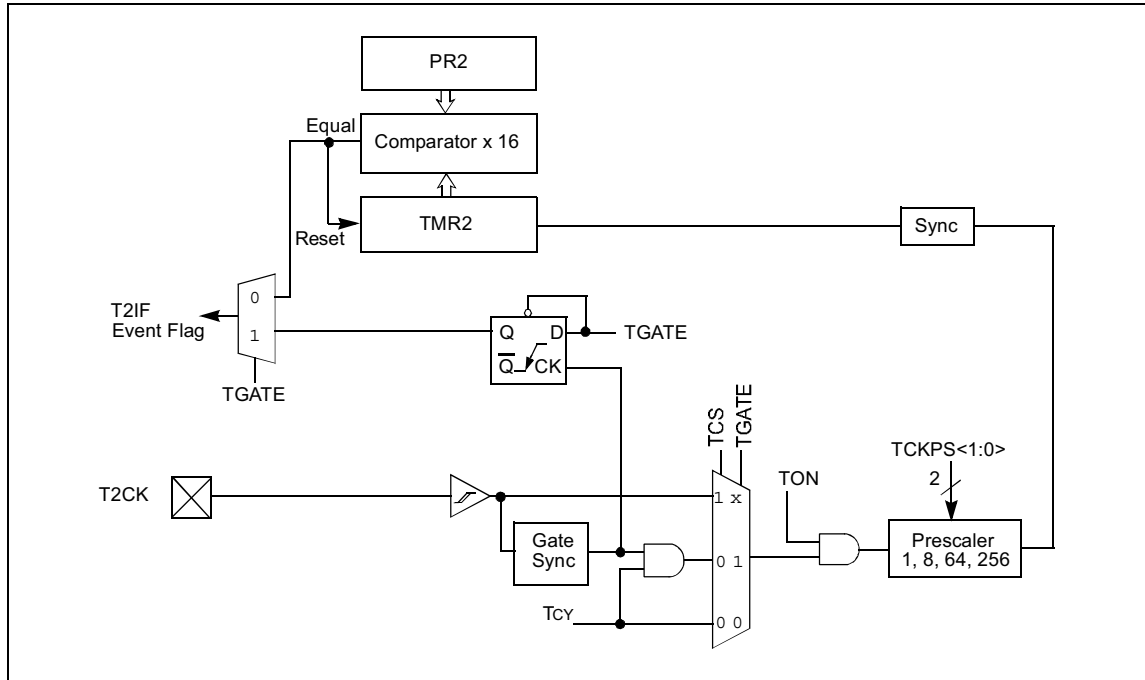
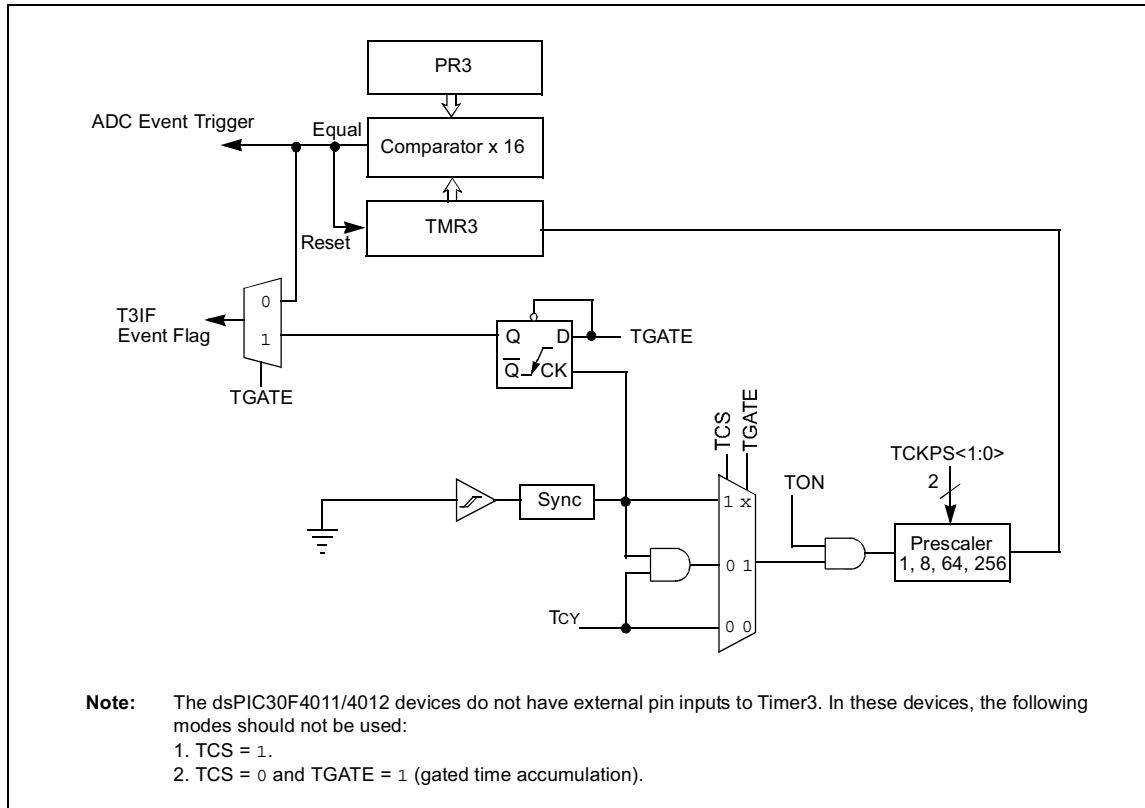


FIGURE 10-3: 16-BIT TIMER3 BLOCK DIAGRAM



dsPIC30F4011/4012

10.1 Timer Gate Operation

The 32-bit timer can be placed in the Gated Time Accumulation mode. This mode allows the internal TCY to increment the respective timer when the gate input signal (T2CK pin) is asserted high. Control bit, TGATE (T2CON<6>), must be set to enable this mode. When in this mode, Timer2 is the originating clock source. The TGATE setting is ignored for Timer3. The timer must be enabled (TON = 1) and the timer clock source set to internal (TCS = 0).

The falling edge of the external signal terminates the count operation but does not reset the timer. The user must reset the timer in order to start counting from zero.

10.2 ADC Event Trigger

When a match occurs between the 32-bit timer (TMR3/TMR2) and the 32-bit combined Period register (PR3/PR2), a special ADC trigger event signal is generated by Timer3.

10.3 Timer Prescaler

The input clock (FOSC/4 or external clock) to the timer has a prescale option of 1:1, 1:8, 1:64 and 1:256, selected by control bits, TCKPS<1:0> (T2CON<5:4> and T3CON<5:4>). For the 32-bit timer operation, the originating clock source is Timer2. The prescaler operation for Timer3 is not applicable in this mode. The prescaler counter is cleared when any of the following occurs:

- a write to the TMR2/TMR3 register
- clearing either of the TON (T2CON<15> or T3CON<15>) bits to '0'
- device Reset, such as POR and BOR

However, if the timer is disabled (TON = 0), then the Timer 2 prescaler cannot be reset, since the prescaler clock is halted.

TMR2/TMR3 is not cleared when T2CON/T3CON is written.

10.4 Timer Operation During Sleep Mode

During CPU Sleep mode, the timer will not operate because the internal clocks are disabled.

10.5 Timer Interrupt

The 32-bit timer module can generate an interrupt on period match, or on the falling edge of the external gate signal. When the 32-bit timer count matches the respective 32-bit Period register, or the falling edge of the external "gate" signal is detected, the T3IF bit (IFS0<7>) is asserted and an interrupt will be generated, if enabled. In this mode, the T3IF interrupt flag is used as the source of the interrupt. The T3IF bit must be cleared in software.

Enabling an interrupt is accomplished via the respective Timer Interrupt Enable bit, T3IE (IEC0<7>).

TABLE 10-1: TIMER2/3 REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State | |
|----------|-------|--|--------|--------|--------|--------|--------|-------|-------|-------|-------|--------|--------|-------|-------|-------|-------|---------------------|---------------------|
| TMR2 | 0106 | Timer2 Register | | | | | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| TMR3HLD | 0108 | Timer3 Holding Register (For 32-bit timer operations only) | | | | | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| TMR3 | 010A | Timer3 Register | | | | | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| PR2 | 010C | Period Register 2 | | | | | | | | | | | | | | | | | 1111 1111 1111 1111 |
| PR3 | 010E | Period Register 3 | | | | | | | | | | | | | | | | | 1111 1111 1111 1111 |
| T2CON | 0110 | TON | — | TSIDL | — | — | — | — | — | — | TGATE | TCKPS1 | TCKPS0 | T32 | — | TCS | — | 0000 0000 0000 0000 | |
| T3CON | 0112 | TON | — | TSIDL | — | — | — | — | — | — | TGATE | TCKPS1 | TCKPS0 | — | — | TCS | — | 0000 0000 0000 0000 | |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

dsPIC30F4011/4012

11.0 TIMER4/5 MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

This section describes the second 32-bit general purpose timer module (Timer4/5) and associated operational modes. Figure 11-1 depicts the simplified block diagram of the 32-bit Timer4/5 module. Figure 11-2 and Figure 11-3 show Timer4/5 configured as two independent 16-bit timers, Timer4 and Timer5, respectively.

Note: Timer4 is a 'Type B' timer and Timer5 is a 'Type C' timer. Please refer to the appropriate timer type in **Section 24.0 "Electrical Characteristics"** of this document.

The Timer4/5 module is similar in operation to the Timer 2/3 module. However, there are some differences, which are as follows:

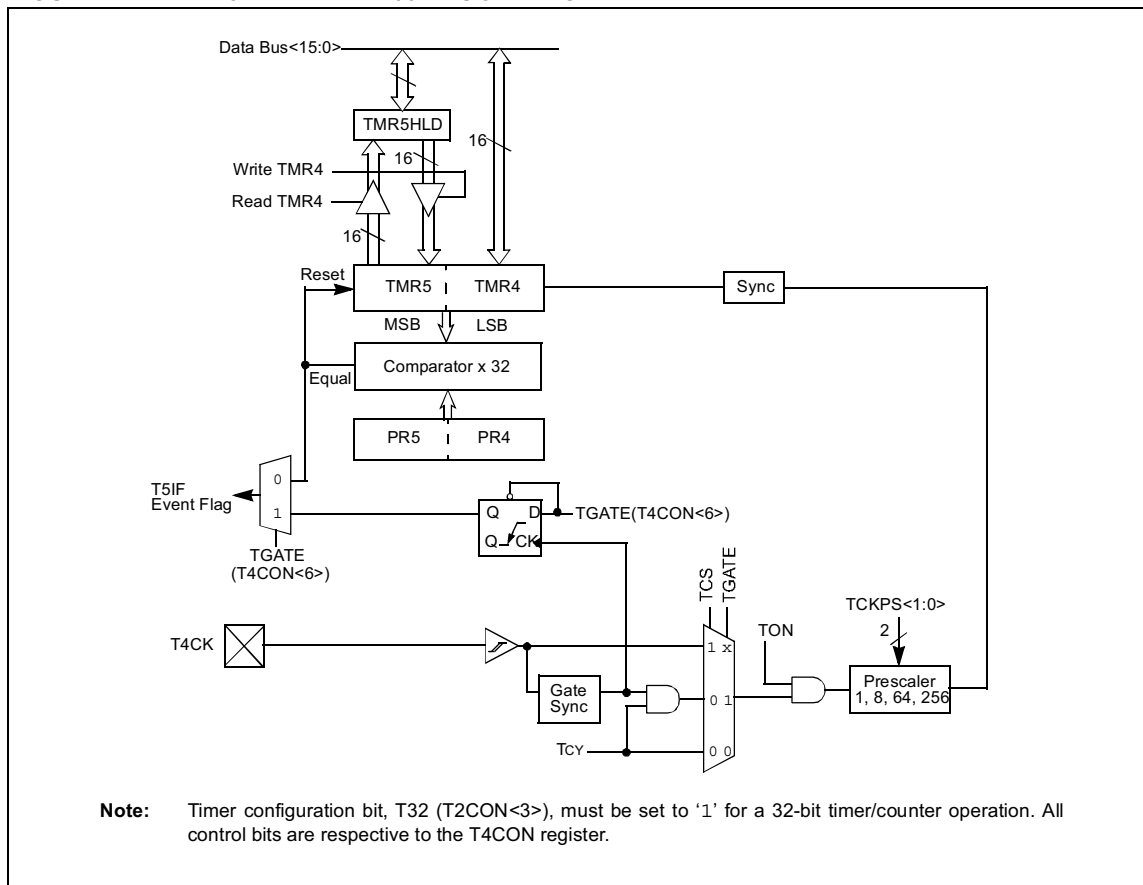
- The Timer4/5 module does not support the ADC event trigger feature
- Timer4/5 can not be utilized by other peripheral modules such as input capture and output compare

The operating modes of the Timer4/5 module are determined by setting the appropriate bit(s) in the 16-bit T4CON and T5CON SFRs.

For 32-bit timer/counter operation, Timer4 is the least significant word and Timer5 is the most significant word of the 32-bit timer.

Note: For 32-bit timer operation, T5CON control bits are ignored. Only T4CON control bits are used for setup and control. Timer4 clock and gate inputs are utilized for the 32-bit timer module, but an interrupt is generated with the Timer5 Interrupt Flag (T5IF) and the interrupt is enabled with the Timer5 Interrupt Enable bit (T5IE).

FIGURE 11-1: 32-BIT TIMER4/5 BLOCK DIAGRAM



dsPIC30F4011/4012

FIGURE 11-2: 16-BIT TIMER4 BLOCK DIAGRAM

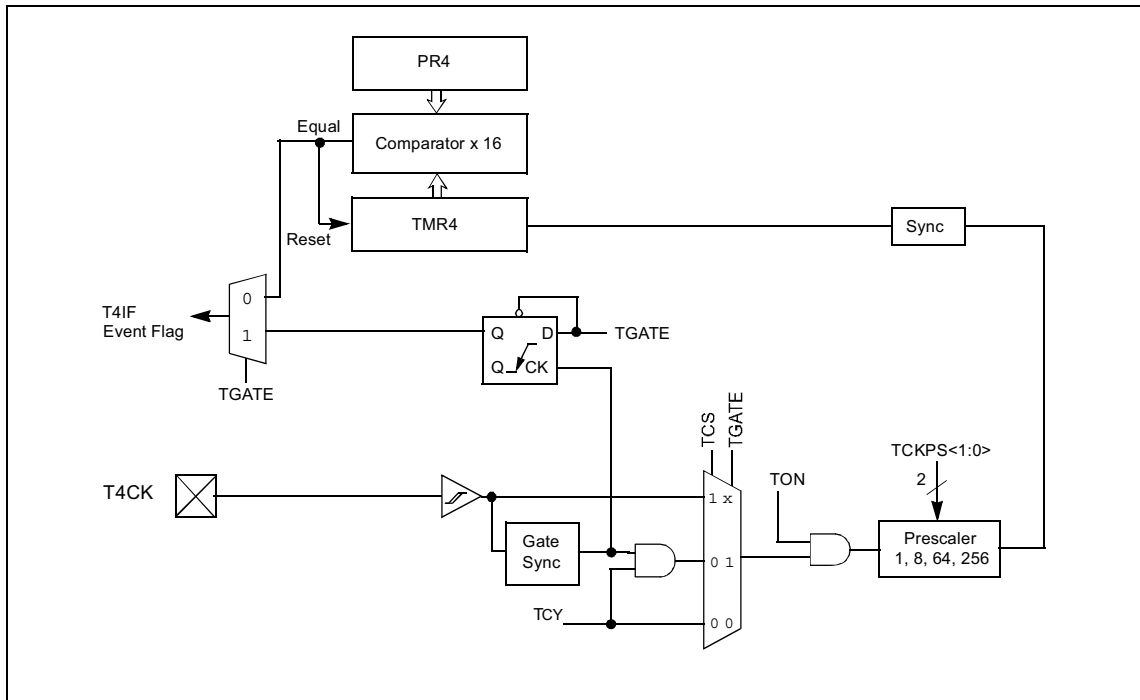
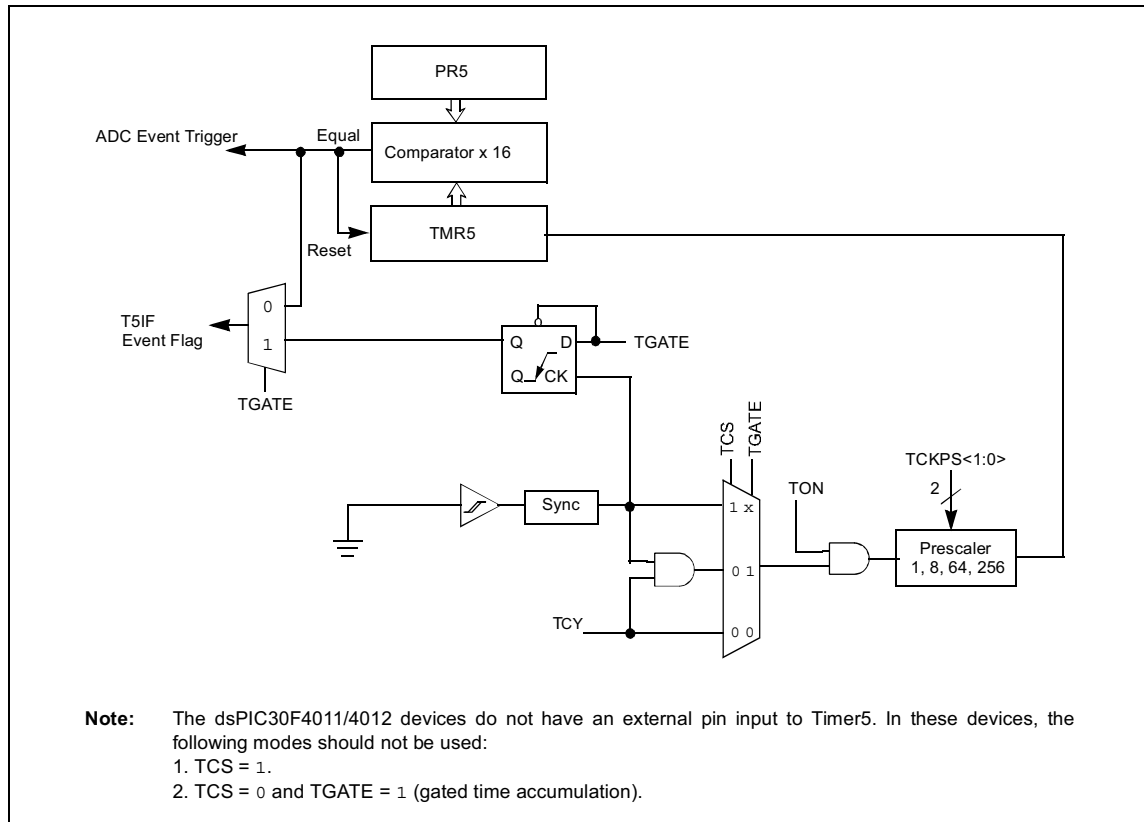


FIGURE 11-3: 16-BIT TIMER5 BLOCK DIAGRAM



dsPIC30F4011/4012

TABLE 11-1: TIMER4/5 REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State | |
|----------|-------|--|--------|--------|--------|--------|--------|-------|-------|-------|-------|--------|--------|-------|-------|-------|-------|---------------------|---------------------|
| TMR4 | 0114 | Timer4 Register | | | | | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| TMR5HLD | 0116 | Timer5 Holding Register (For 32-bit operations only) | | | | | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| TMR5 | 0118 | Timer5 Register | | | | | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| PR4 | 011A | Period Register 4 | | | | | | | | | | | | | | | | | 1111 1111 1111 1111 |
| PR5 | 011C | Period Register 5 | | | | | | | | | | | | | | | | | 1111 1111 1111 1111 |
| T4CON | 011E | TON | — | TSIDL | — | — | — | — | — | — | TGATE | TCKPS0 | TCKPS1 | T45 | — | TCS | — | 0000 0000 0000 0000 | |
| T5CON | 0120 | TON | — | TSIDL | — | — | — | — | — | — | TGATE | TCKPS0 | TCKPS1 | — | — | TCS | — | 0000 0000 0000 0000 | |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

12.0 INPUT CAPTURE MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

This section describes the input capture module and associated operational modes. The features provided by this module are useful in applications requiring frequency (period) and pulse measurement. Figure 12-1 depicts a block diagram of the input capture module. Input capture is useful for such modes as:

- Frequency/Period/Pulse Measurements
- Additional Sources of External Interrupts

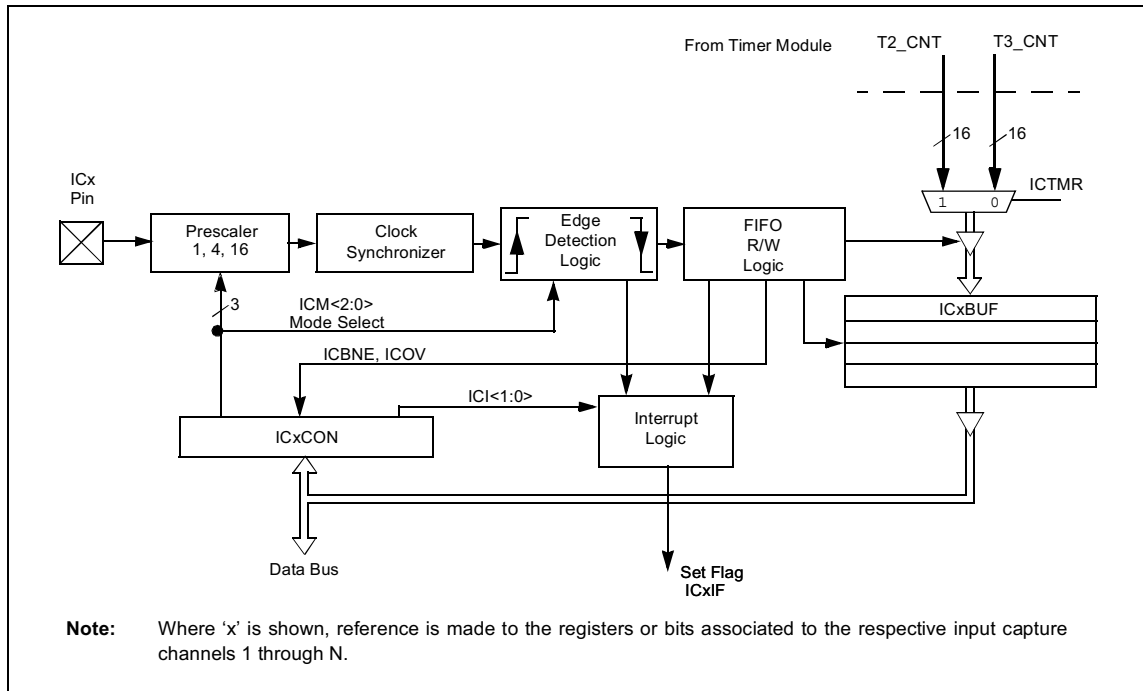
The key operational features of the input capture module are:

- Simple Capture Event mode
- Timer2 and Timer3 mode selection
- Interrupt on input capture event

These operating modes are determined by setting the appropriate bits in the ICxCON register (where x = 1, 2, ..., N). The dsPIC30F4011/4012 devices have 4 capture channels.

Note: The dsPIC30F4011/4012 devices have four capture inputs: IC1, IC2, IC7 and IC8. The naming of these four capture channels is intentional and preserves software compatibility with other dsPIC Digital Signal Controllers.

FIGURE 12-1: INPUT CAPTURE MODE BLOCK DIAGRAM



dsPIC30F4011/4012

12.1 Simple Capture Event Mode

The simple capture events in the dsPIC30F product family are:

- Capture every falling edge
- Capture every rising edge
- Capture every 4th rising edge
- Capture every 16th rising edge
- Capture every rising and falling edge

These simple Input Capture modes are configured by setting the appropriate bits ICM<2:0> (ICxCON<2:0>).

12.1.1 CAPTURE PRESCALER

There are four input capture prescaler settings, specified by bits, ICM<2:0> (ICxCON<2:0>). Whenever the capture channel is turned off, the prescaler counter will be cleared. In addition, any Reset will clear the prescaler counter.

12.1.2 CAPTURE BUFFER OPERATION

Each capture channel has an associated FIFO buffer which is four, 16-bit words deep. There are two status flags which provide status on the FIFO buffer:

- ICBNE – Input Capture Buffer Not Empty
- ICOV – Input Capture Overflow

The ICBNE bit will be set on the first input capture event and remain set until all capture events have been read from the FIFO. As each word is read from the FIFO, the remaining words are advanced by one position within the buffer.

In the event that the FIFO is full with four capture events, and a fifth capture event occurs prior to a read of the FIFO, an overflow condition will occur and the ICOV bit will be set to a logic '1'. The fifth capture event is lost and is not stored in the FIFO. No additional events will be captured till all four events have been read from the buffer.

If a FIFO read is performed after the last read and no new capture event has been received, the read will yield indeterminate results.

12.1.3 TIMER2 AND TIMER3 SELECTION MODE

Each capture channel can select between one of two timers for the time base, Timer2 or Timer3.

Selection of the timer resource is accomplished through SFR bit, ICTMR (ICxCON<7>). Timer3 is the default timer resource available for the input capture module.

12.1.4 HALL SENSOR MODE

When the input capture module is set for capture on every edge, rising and falling, ICM<2:0> = 001, the following operations are performed by the input capture logic:

- The input capture interrupt flag is set on every edge, rising and falling.
- The interrupt on Capture mode setting bits, ICI<1:0>, is ignored, since every capture generates an interrupt.
- A capture overflow condition is not generated in this mode.

12.2 Input Capture Operation During Sleep and Idle Modes

An input capture event will generate a device wake-up or interrupt, if enabled, if the device is in CPU Idle or Sleep mode.

Independent of the timer being enabled, the input capture module will wake-up from the CPU Sleep or Idle mode when a capture event occurs, if ICM<2:0> = 111 and the interrupt enable bit is asserted. The same wake-up can generate an interrupt if the conditions for processing the interrupt have been satisfied. The wake-up feature is useful as a method of adding extra external pin interrupts.

12.2.1 INPUT CAPTURE IN CPU SLEEP MODE

CPU Sleep mode allows input capture module operation with reduced functionality. In the CPU Sleep mode, the ICI<1:0> bits are not applicable, and the input capture module can only function as an external interrupt source.

The capture module must be configured for interrupt only on the rising edge (ICM<2:0> = 111) in order for the input capture module to be used while the device is in Sleep mode. The prescale settings of 4:1 or 16:1 are not applicable in this mode.

12.2.2 INPUT CAPTURE IN CPU IDLE MODE

CPU Idle mode allows input capture module operation with full functionality. In the CPU Idle mode, the interrupt mode selected by the $IC1<1:0>$ bits are applicable, as well as the 4:1 and 16:1 capture prescale settings which are defined by control bits $ICM<2:0>$. This mode requires the selected timer to be enabled. Moreover, the $ICSIDL$ bit must be asserted to a logic '0'.

If the input capture module is defined as $ICM<2:0> = 111$ in CPU Idle mode, the input capture pin will serve only as an external interrupt pin.

12.3 Input Capture Interrupts

The input capture channels have the ability to generate an interrupt, based upon the selected number of capture events. The selection number is set by control bits, $IC1<1:0>$ ($ICxCON<6:5>$).

Each channel provides an interrupt flag ($ICxIF$) bit. The respective capture channel interrupt flag is located in the corresponding $IFSx$ register.

Enabling an interrupt is accomplished via the respective Capture Channel Interrupt Enable ($ICxIE$) bit. The capture interrupt enable bit is located in the corresponding $IECx$ register.

TABLE 12-1: INPUT CAPTURE REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|--------------------------|--------|--------|--------|--------|--------|-------|-------|-------|----------|-------|-------|----------|-------|-------|-------|---------------------|
| IC1BUF | 0140 | Input 1 Capture Register | | | | | | | | | | | | | | | | |
| IC1CON | 0142 | — | — | ICSIDL | — | — | — | — | — | ICTMR | IC1<1:0> | ICOV | ICBNE | ICM<2:0> | — | — | — | uuuu uuuu uuuu uuuu |
| IC2BUF | 0144 | Input 2 Capture Register | | | | | | | | | | | | | | | | |
| IC2CON | 0146 | — | — | ICSIDL | — | — | — | — | — | ICTMR | IC1<1:0> | ICOV | ICBNE | ICM<2:0> | — | — | — | uuuu uuuu uuuu uuuu |
| IC7BUF | 0158 | Input 7 Capture Register | | | | | | | | | | | | | | | | |
| IC7CON | 015A | — | — | ICSIDL | — | — | — | — | — | ICTMR | IC1<1:0> | ICOV | ICBNE | ICM<2:0> | — | — | — | uuuu uuuu uuuu uuuu |
| IC8BUF | 015C | Input 8 Capture Register | | | | | | | | | | | | | | | | |
| IC8CON | 015E | — | — | ICSIDL | — | — | — | — | — | ICTMR | IC1<1:0> | ICOV | ICBNE | ICM<2:0> | — | — | — | uuuu uuuu uuuu uuuu |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

13.0 OUTPUT COMPARE MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

This section describes the output compare module and associated operational modes. The features provided by this module are useful in applications requiring operational modes, such as:

- Generation of Variable Width Output Pulses
- Power Factor Correction

Figure 13-1 depicts a block diagram of the output compare module.

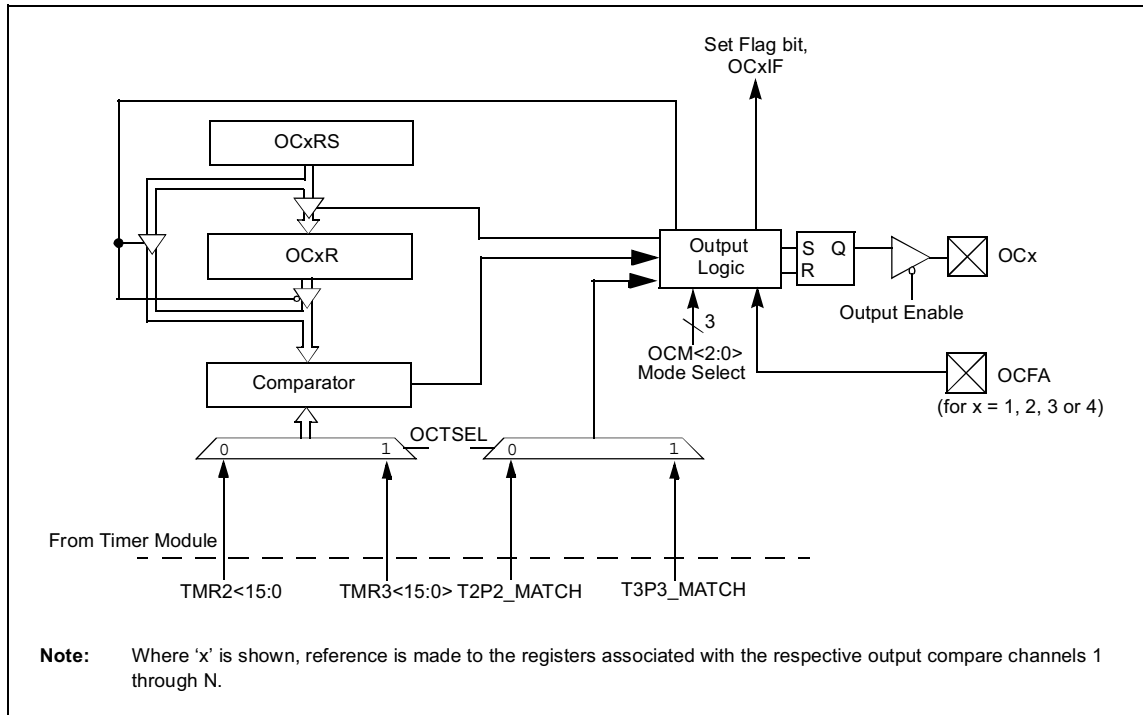
The key operational features of the output compare module include:

- Timer2 and Timer3 Selection mode
- Simple Output Compare Match mode
- Dual Output Compare Match mode
- Simple PWM mode
- Output Compare during Sleep and Idle modes
- Interrupt on Output Compare/PWM Event

These operating modes are determined by setting the appropriate bits in the 16-bit OCxCON SFR (where x = 1, 2, 3, ..., N). The dsPIC30F4011/4012 devices have 4/2 compare channels, respectively.

OCxRS and OCxR in the figure represent the Dual Compare registers. In the Dual Compare mode, the OCxR register is used for the first compare and OCxRS is used for the second compare.

FIGURE 13-1: OUTPUT COMPARE MODE BLOCK DIAGRAM



dsPIC30F4011/4012

13.1 Timer2 and Timer3 Selection Mode

Each output compare channel can select between one of two 16-bit timers: Timer2 or Timer3.

The selection of the timers is controlled by the OCTSEL bit (OCxCON<3>). Timer2 is the default timer resource for the output compare module.

13.2 Simple Output Compare Match Mode

When control bits, OCM<2:0> (OCxCON<2:0>) = 001, 010 or 011, the selected output compare channel is configured for one of three simple Output Compare Match modes:

- Compare forces I/O pin low
- Compare forces I/O pin high
- Compare toggles I/O pin

The OCxR register is used in these modes. The OCxR register is loaded with a value and is compared to the selected incrementing timer count. When a compare occurs, one of these Compare Match modes occurs. If the counter resets to zero before reaching the value in OCxR, the state of the OCx pin remains unchanged.

13.3 Dual Output Compare Match Mode

When control bits, OCM<2:0> (OCxCON<2:0>) = 100 or 101, the selected output compare channel is configured for one of two Dual Output Compare modes which are:

- Single Output Pulse mode
- Continuous Output Pulse mode

13.3.1 SINGLE OUTPUT PULSE MODE

For the user to configure the module for the generation of a single output pulse, the following steps are required (assuming timer is off):

- Determine instruction cycle time T_{CY} .
- Calculate desired pulse width value based on T_{CY} .
- Calculate time to start pulse from timer start value of 0x0000.
- Write pulse width start and stop times into OCxR and OCxRS Compare registers (x denotes channel 1, 2, ..., N).
- Set Timer Period register to value equal to, or greater than, value in OCxRS Compare register.
- Set OCM<2:0> = 100.
- Enable timer, TON (TxCON<15>) = 1.

To initiate another single pulse, issue another write to set OCM<2:0> = 100.

13.3.2 CONTINUOUS OUTPUT PULSE MODE

For the user to configure the module for the generation of a continuous stream of output pulses, the following steps are required:

- Determine instruction cycle time T_{CY} .
- Calculate desired pulse value based on T_{CY} .
- Calculate timer to start pulse width from timer start value of 0x0000.
- Write pulse width start and stop times into OCxR and OCxRS (x denotes channel 1, 2, ..., N) Compare registers, respectively.
- Set Timer Period register to value equal to, or greater than, value in OCxRS Compare register.
- Set OCM<2:0> = 101.
- Enable timer, TON (TxCON<15>) = 1.

13.4 Simple PWM Mode

When control bits, OCM<2:0> (OCxCON<2:0>) = 110 or 111, the selected output compare channel is configured for the PWM mode of operation. When configured for the PWM mode of operation, OCxR is the main latch (read-only) and OCxRS is the secondary latch. This enables glitchless PWM transitions.

The user must perform the following steps in order to configure the output compare module for PWM operation:

1. Set the PWM period by writing to the appropriate Period register.
2. Set the PWM duty cycle by writing to the OCxRS register.
3. Configure the output compare module for PWM operation.
4. Set the TMRx prescale value and enable the timer, TON (TxCON<15>) = 1.

13.4.1 INPUT PIN FAULT PROTECTION FOR PWM

When control bits, OCM<2:0> (OCxCON<2:0>) = 111, the selected output compare channel is again configured for the PWM mode of operation with the additional feature of input Fault protection. While in this mode, if a logic '0' is detected on the OCFA pin, the respective PWM output pin is placed in the high-impedance input state. The OCFLT bit (OCxCON<4>) indicates whether a Fault condition has occurred. This state will be maintained until both of the following events have occurred:

- The external Fault condition has been removed.
- The PWM mode has been re-enabled by writing to the appropriate control bits.

13.4.2 PWM PERIOD

The PWM period is specified by writing to the PRx register. The PWM period can be calculated using Equation 13-1.

EQUATION 13-1: PWM PERIOD

$$\text{PWM period} = [(PRx) + 1] \cdot 4 \cdot \text{Tosc} \cdot (\text{TMRx prescale value})$$

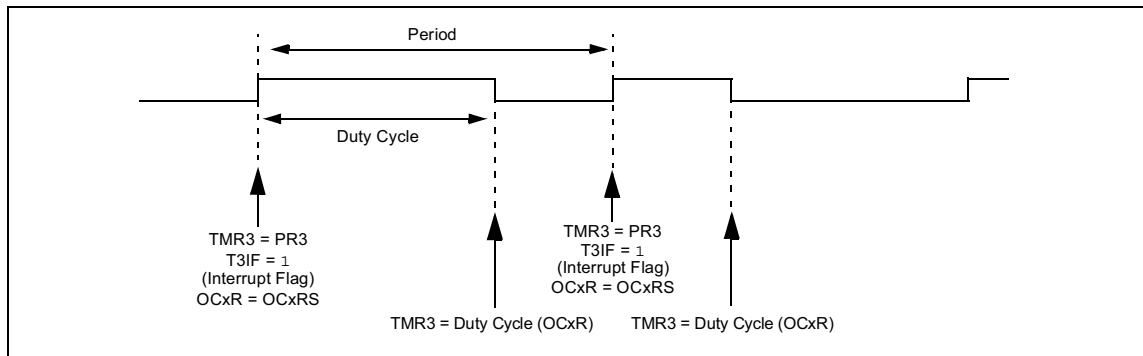
PWM frequency is defined as $1/[\text{PWM period}]$.

When the selected TMRx is equal to its respective Period register, PRx, the following four events occur on the next increment cycle:

- TMRx is cleared.
- The OCx pin is set.
 - Exception 1: If PWM duty cycle is 0x0000, the OCx pin will remain low.
 - Exception 2: If duty cycle is greater than PRx, the pin will remain high.
- The PWM duty cycle is latched from OCxRS into OCxR.
- The corresponding timer interrupt flag is set.

See Figure 13-1 for key PWM period comparisons. Timer3 is referred to in the figure for clarity.

FIGURE 13-1: PWM OUTPUT TIMING



13.5 Output Compare Operation During CPU Sleep Mode

When the CPU enters the Sleep mode, all internal clocks are stopped. Therefore, when the CPU enters the Sleep state, the output compare channel will drive the pin to the active state that was observed prior to entering the CPU Sleep state.

For example, if the pin was high when the CPU entered the Sleep state, the pin will remain high. Likewise, if the pin was low when the CPU entered the Sleep state, the pin will remain low. In either case, the output compare module will resume operation when the device wakes up.

13.6 Output Compare Operation During CPU Idle Mode

When the CPU enters the Idle mode, the output compare module can operate with full functionality.

The output compare channel will operate during the CPU Idle mode if the OCSIDL bit (OCxCON<13>) is at logic '0' and the selected time base (Timer2 or Timer3) is enabled and the TSIDL bit of the selected timer is set to logic '0'.

13.7 Output Compare Interrupts

The output compare channels have the ability to generate an interrupt on a compare match for whichever Match mode has been selected.

For all modes except the PWM mode, when a compare event occurs, the respective interrupt flag (OCxIF) is asserted and an interrupt will be generated if enabled. The OCxIF bit is located in the corresponding IFSx register, and must be cleared in software. The interrupt is enabled via the respective Compare Interrupt Enable (OCxIE) bit, located in the corresponding IECx register.

For the PWM mode, when an event occurs, the respective Timer Interrupt Flag (T2IF or T3IF) is asserted and an interrupt will be generated if enabled. The TxIF bit is located in the IFS0 register and must be cleared in software. The interrupt is enabled via the respective Timer Interrupt Enable bit (T2IE or T3IE) located in the IEC0 register. The output compare interrupt flag is never set during the PWM mode of operation.

TABLE 13-1: OUTPUT COMPARE REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State | |
|----------|-------|-------------------------------------|--------|--------|--------|--------|--------|-------|-------|-------|-------|-------|--------|----------|-------|-------|-------|---------------------|---------------------|
| OC1RS | 0180 | Output Compare 1 Secondary Register | | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC1R | 0182 | Output Compare 1 Main Register | | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC1CON | 0184 | — | — | OCSIDL | — | — | — | — | — | — | — | OCFLT | OCTSEL | OCM<2:0> | | | | 0000 0000 0000 0000 | |
| OC2RS | 0186 | Output Compare 2 Secondary Register | | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC2R | 0188 | Output Compare 2 Main Register | | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC2CON | 018A | — | — | OCSIDL | — | — | — | — | — | — | — | OCFLT | OCTSEL | OCM<2:0> | | | | 0000 0000 0000 0000 | |
| OC3RS* | 018C | Output Compare 3 Secondary Register | | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC3R* | 018E | Output Compare 3 Main Register | | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC3CON* | 0190 | — | — | OCSIDL | — | — | — | — | — | — | — | OCFLT | OCTSEL | OCM<2:0> | | | | 0000 0000 0000 0000 | |
| OC4RS* | 0192 | Output Compare 4 Secondary Register | | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC4R* | 0194 | Output Compare 4 Main Register | | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |
| OC4CON* | 0196 | — | — | OCSIDL | — | — | — | — | — | — | — | OCFLT | OCTSEL | OCM<2:0> | | | | 0000 0000 0000 0000 | |

Legend: u = uninitialized bit

* Not available on dsPIC30F4012.

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

14.0 QUADRATURE ENCODER INTERFACE (QEI) MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

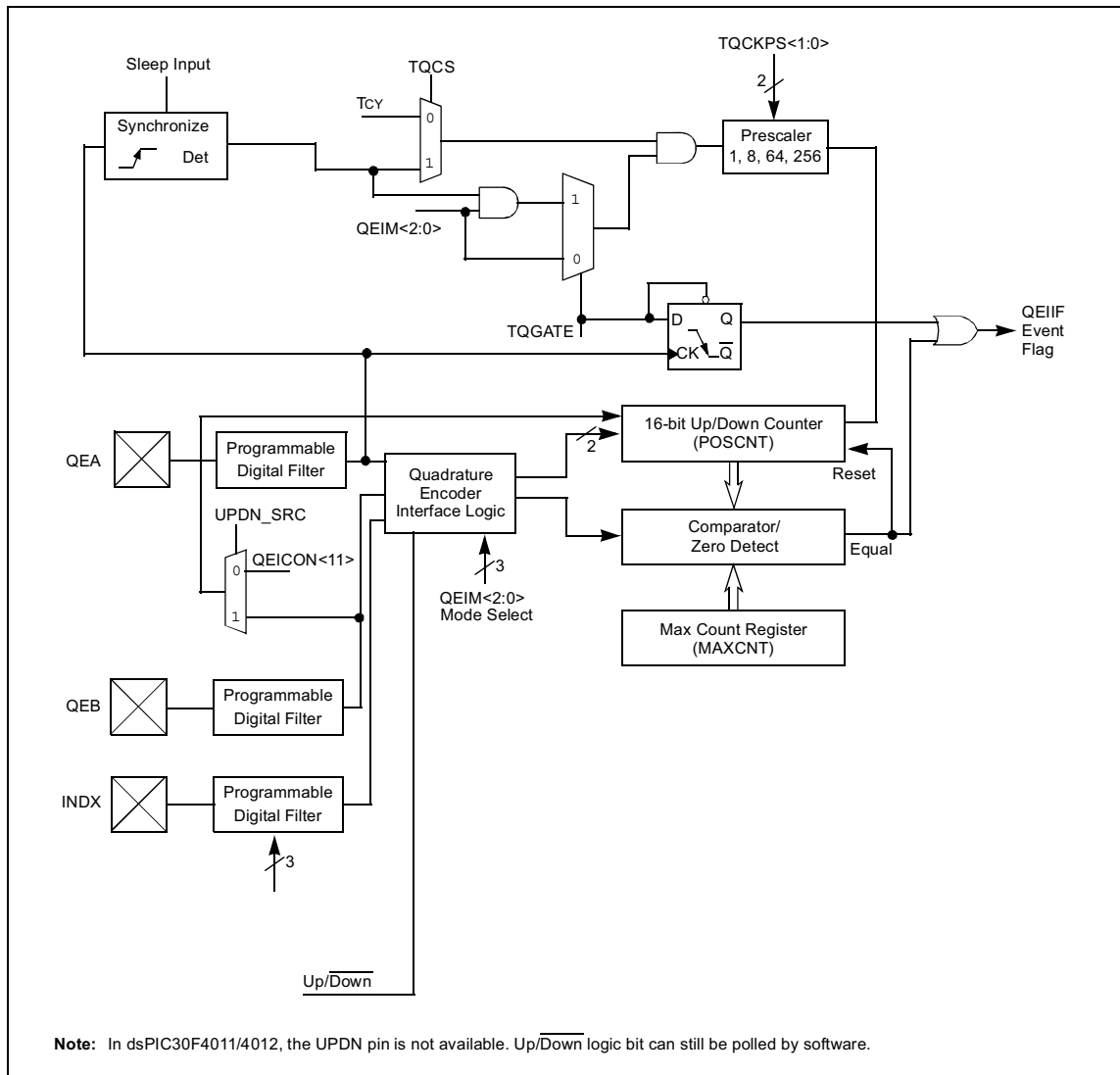
This section describes the Quadrature Encoder Interface (QEI) module and associated operational modes. The QEI module provides the interface to incremental encoders for obtaining mechanical position data.

The operational features of the QEI include:

- Three input channels for two phase signals and index pulse
- 16-bit up/down position counter
- Count direction status
- Position Measurement (x2 and x4) mode
- Programmable digital noise filters on inputs
- Alternate 16-bit Timer/Counter mode
- Quadrature Encoder Interface interrupts

These operating modes are determined by setting the appropriate bits QEIM<2:0> (QEICON<10:8>). Figure 14-1 depicts the Quadrature Encoder Interface block diagram.

FIGURE 14-1: QUADRATURE ENCODER INTERFACE BLOCK DIAGRAM



dsPIC30F4011/4012

14.1 Quadrature Encoder Interface Logic

A typical, incremental (a.k.a. optical) encoder has three outputs: Phase A, Phase B and an index pulse. These signals are useful and often required in position and speed control of ACIM and SR motors.

The two channels, Phase A (QEA) and Phase B (QEB), have a unique relationship. If Phase A leads Phase B, then the direction (of the motor) is deemed positive or forward. If Phase A lags Phase B, then the direction (of the motor) is deemed negative or reverse.

A third channel, termed index pulse, occurs once per revolution and is used as a reference to establish an absolute position. The index pulse coincides with Phase A and Phase B, both low.

14.2 16-bit Up/Down Position Counter Mode

The 16-bit Up/Down Counter counts up or down on every count pulse, which is generated by the difference of the Phase A and Phase B input signals. The counter acts as an integrator, whose count value is proportional to position. The direction of the count is determined by the UPDN signal which is generated by the Quadrature Encoder Interface Logic.

14.2.1 POSITION COUNTER ERROR CHECKING

Position counter error checking in the QEI is provided for and indicated by the CNTERR bit (QEICON<15>). The error checking only applies when the position counter is configured for Reset on the Index Pulse modes (QEIM<2:0> = 110 or 100). In these modes, the contents of the POSCNT register are compared with the values (0xFFFF or MAXCNT + 1, depending on direction). If these values are detected, an error condition is generated by setting the CNTERR bit and a QEI count error interrupt is generated. The QEI count error interrupt can be disabled by setting the CEID bit (DFLTCON<8>). The position counter continues to count encoder edges after an error has been detected. The POSCNT register continues to count up/down until a natural rollover/underflow. No interrupt is generated for the natural rollover/underflow event. The CNTERR bit is a read/write bit and reset in software by the user.

14.2.2 POSITION COUNTER RESET

The Position Counter Reset Enable bit, POSRES (QEICON<2>) controls whether the position counter is reset when the index pulse is detected. This bit is only applicable when QEIM<2:0> = 100 or 110.

If the POSRES bit is set to '1', then the position counter is reset when the index pulse is detected. If the POSRES bit is set to '0', then the position counter is not reset when the index pulse is detected. The position counter will continue counting up or down and will be reset on the rollover or underflow condition.

When selecting the INDX signal to reset the position counter (POSCNT), the user has to specify the states on the QEA and QEB input pins. These states have to be matched in order for a Reset to occur. These states are selected by the IMV<1:0> bits (DFLTCON<10:9>).

The Index Match Value bits (IMV<1:0>) allow the user to specify the state of the QEA and QEB input pins, during an index pulse, when the POSCNT register is to be reset.

In 4x Quadrature Count mode:

IMV1 = Required state of Phase B input signal for match on index pulse

IMV0 = Required state of Phase A input signal for match on index pulse

In 2x Quadrature Count mode:

IMV1 = Selects phase input signal for index state match (0 = Phase A, 1 = Phase B)

IMV0 = Required state of the selected phase input signal for match on index pulse

The interrupt is still generated on the detection of the index pulse and not on the position counter overflow/underflow.

14.2.3 COUNT DIRECTION STATUS

As mentioned in the previous section, the QEI logic generates an UPDN signal, based upon the relationship between Phase A and Phase B. In addition to the output pin, the state of this internal UPDN signal is supplied to an SFR bit, UPDN (QEICON<11>), as a read-only bit.

Note: QEI pins are multiplexed with analog inputs. The user must insure that all QEI associated pins are set as digital inputs in the ADPCFG register.

14.3 Position Measurement Mode

There are two Measurement modes which are supported and are termed x2 and x4. These modes are selected by the QEIM<2:0> (QEICON<10:8>) mode select bits.

When control bits, QEIM<2:0> = 100 or 101, the x2 Measurement mode is selected and the QEI logic only looks at the Phase A input for the position counter increment rate. Every rising and falling edge of the Phase A signal causes the position counter to be incremented or decremented. The Phase B signal is still utilized for the determination of the counter direction just as in the x4 mode.

Within the x2 Measurement mode, there are two variations of how the position counter is reset:

1. Position counter reset by detection of index pulse, QEIM<2:0> = 100.
2. Position counter reset by match with MAXCNT, QEIM<2:0> = 101.

When control bits, QEIM<2:0> = 110 or 111, the x4 Measurement mode is selected and the QEI logic looks at both edges of the Phase A and Phase B input signals. Every edge of both signals causes the position counter to increment or decrement.

Within the x4 Measurement mode, there are two variations of how the position counter is reset:

1. Position counter reset by detection of index pulse, QEIM<2:0> = 110.
2. Position counter reset by match with MAXCNT, QEIM<2:0> = 111.

The x4 Measurement mode provides for finer resolution data (more position counts) for determining motor position.

14.4 Programmable Digital Noise Filters

The digital noise filter section is responsible for rejecting noise on the incoming capture or quadrature signals. Schmitt Trigger inputs and a three-clock cycle delay filter combine to reject low-level noise and large, short duration, noise spikes that typically occur in noise prone applications, such as a motor system.

The filter ensures that the filtered output signal is not permitted to change until a stable value has been registered for three consecutive clock cycles.

For the QEA, QEB and INDX pins, the clock divide frequency for the digital filter is programmed by bits QECK<2:0> (DFLTCON<6:4>) and are derived from the base instruction cycle Tcy.

To enable the filter output for channels QEA, QEB and INDX, the QEOUT bit must be '1'. The filter network for all channels is disabled on POR and BOR.

14.5 Alternate 16-bit Timer/Counter

When the QEI module is not configured for the QEI mode, QEIM<2:0> = 001, the module can be configured as a simple 16-bit timer/counter. The setup and control of the auxiliary timer is accomplished through the QEICON SFR register. This timer functions identically to Timer1. The QEA pin is used as the timer clock input.

When configured as a timer, the POSCNT register serves as the Timer Count register and the MAXCNT register serves as the Period register. When a Timer/Period register match occurs, the QEI interrupt flag will be asserted.

The only exception between the general purpose timers and this timer is the added feature of external up/down input select. When the UPDN pin is asserted high, the timer will increment up. When the UPDN pin is asserted low, the timer will be decremented.

Note: Changing the operational mode (i.e., from QEI to timer or vice versa) will not affect the Timer/Position Count register contents.

The UPDN control/status bit (QEICON<11>) can be used to select the count direction state of the Timer register. When UPDN = 1, the timer will count up. When UPDN = 0, the timer will count down.

In addition, control bit, UPDN_SRC (QEICON<0>), determines whether the timer count direction state is based on the logic state written into the UPDN control/status bit (QEICON<11>) or the QEB pin state. When UPDN_SRC = 1, the timer count direction is controlled from the QEB pin. Likewise, when UPDN_SRC = 0, the timer count direction is controlled by the UPDN bit.

Note: This timer does not support the External Asynchronous Counter mode of operation. If using an external clock source, the clock will automatically be synchronized to the internal instruction cycle.

14.6 QEI Module Operation During CPU Sleep Mode

14.6.1 QEI OPERATION DURING CPU SLEEP MODE

The QEI module will be halted during the CPU Sleep mode.

14.6.2 TIMER OPERATION DURING CPU SLEEP MODE

During CPU Sleep mode, the timer will not operate because the internal clocks are disabled.

dsPIC30F4011/4012

14.7 QEI Module Operation During CPU Idle Mode

Since the QEI module can function as a quadrature encoder interface, or as a 16-bit timer, the following section describes operation of the module in both modes.

14.7.1 QEI OPERATION DURING CPU IDLE MODE

When the CPU is placed in the Idle mode, the QEI module will operate if the QEISIDL bit (QEICON<13>) = 0. This bit defaults to a logic '0' upon executing POR and BOR. For halting the QEI module during the CPU Idle mode, QEISIDL should be set to '1'.

14.7.2 TIMER OPERATION DURING CPU IDLE MODE

When the CPU is placed in the Idle mode and the QEI module is configured in the 16-bit Timer mode, the 16-bit timer will operate if the QEISIDL bit (QEICON<13>) = 0. This bit defaults to a logic '0' upon executing POR and BOR. For halting the timer module during the CPU Idle mode, QEISIDL should be set to '1'.

If the QEISIDL bit is cleared, the timer will function normally as if the CPU Idle mode had not been entered.

14.8 Quadrature Encoder Interface Interrupts

The quadrature encoder interface has the ability to generate an interrupt on occurrence of the following events:

- Interrupt on 16-bit up/down position counter rollover/underflow
- Detection of qualified index pulse or if CNTERR bit is set
- Timer period match event (overflow/underflow)
- Gate accumulation event

The QEI Interrupt Flag bit, QEIIF, is asserted upon occurrence of any of the above events. The QEIIF bit must be cleared in software. QEIIF is located in the IFS2 register.

Enabling an interrupt is accomplished via the respective Enable bit, QEIIE. The QEIIE bit is located in the IEC2 register.

TABLE 14-1: QEI REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|------------------------|--------|---------|--------|--------|--------|-------|-------|-------|--------|--------|---------|---------|--------|-------|----------|---------------------|
| QEICON | 0122 | CNTERR | — | QEISIDL | INDX | UPDN | QEIM2 | QEIM1 | QEIM0 | SWPAB | — | TQGATE | TQCKPS1 | TQCKPS0 | POSRES | TQCS | UPDN_SRC | 0000 0000 0000 0000 |
| DFLTCON | 0124 | — | — | — | — | — | IMV1 | IMV0 | CEID | QEOUT | QEACK2 | QEACK1 | QEACK0 | — | — | — | — | 0000 0000 0000 0000 |
| POSCNT | 0126 | Position Counter<15:0> | | | | | | | | | | | | | | | | |
| MAXCNT | 0128 | Maximum Count<15:0> | | | | | | | | | | | | | | | | |
| ADPCFG | 02A8 | — | — | — | — | — | — | — | PCFG8 | PCFG7 | PCFG6 | PCFG5 | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

15.0 MOTOR CONTROL PWM MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *"dsPIC30F/33F Programmer's Reference Manual"* (DS70157).

This module simplifies the task of generating multiple, synchronized Pulse-Width Modulated (PWM) outputs. In particular, the following power and motion control applications are supported by the PWM module:

- Three-Phase AC Induction Motor
- Switched Reluctance (SR) Motor
- Brushless DC (BLDC) Motor
- Uninterruptible Power Supply (UPS)

The PWM module has the following features:

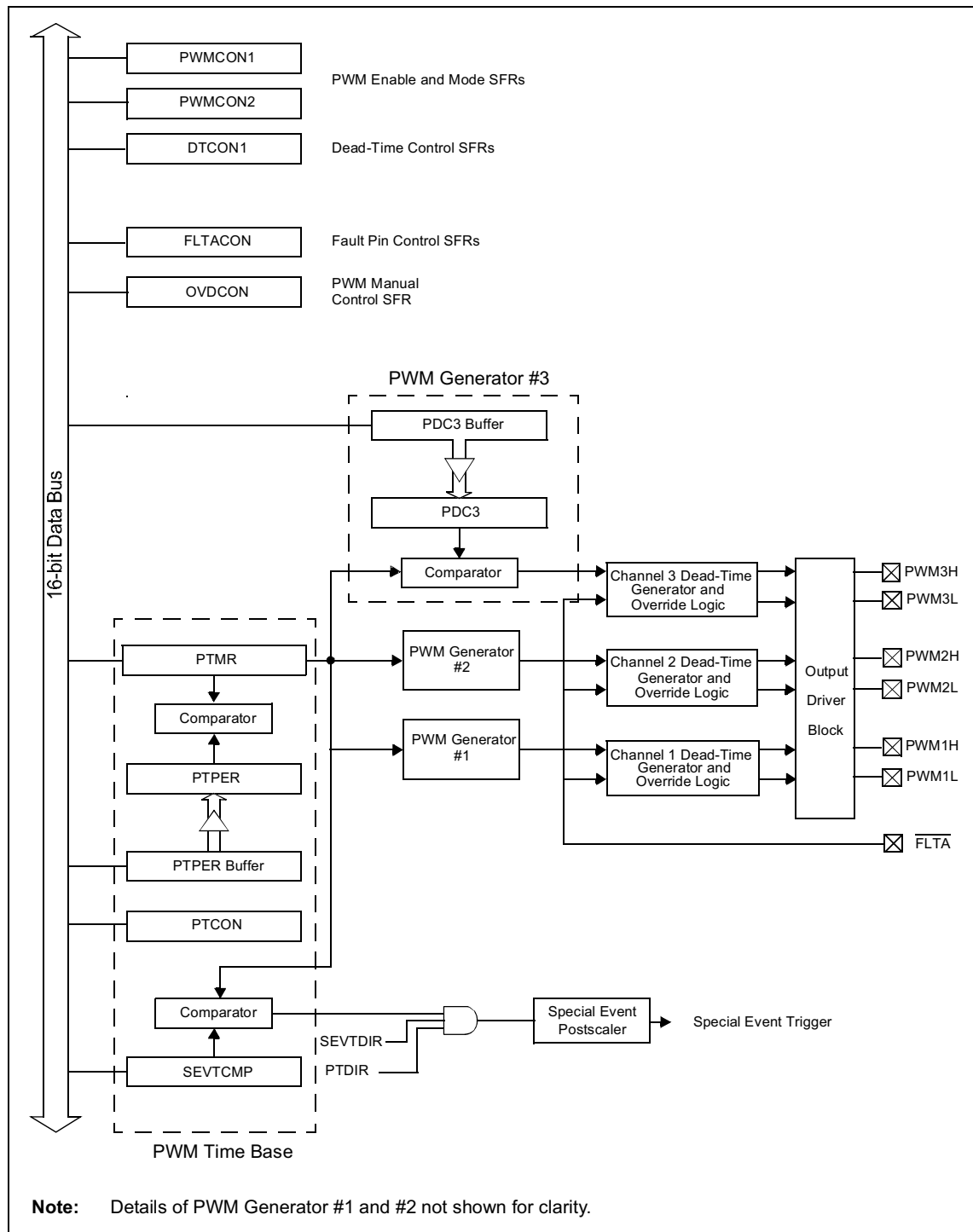
- 6 PWM I/O pins with 3 duty cycle generators
- Up to 16-bit resolution
- 'On-the-Fly' PWM frequency changes
- Edge and Center-Aligned Output modes
- Single Pulse Generation mode
- Interrupt support for asymmetrical updates in Center-Aligned mode
- Output override control for Electrically Commutative Motor (ECM) operation
- 'Special Event' comparator for scheduling other peripheral events
- Fault pins to optionally drive each of the PWM output pins to a defined state

This module contains 3 duty cycle generators, numbered 1 through 3. The module has 6 PWM output pins, numbered PWM1H/PWM1L through PWM3H/PWM3L. The six I/O pins are grouped into high/low numbered pairs, denoted by the suffix H or L, respectively. For complementary loads, the low PWM pins are always the complement of the corresponding high I/O pin.

The PWM module allows several modes of operation which are beneficial for specific power control applications.

dsPIC30F4011/4012

FIGURE 15-1: PWM MODULE BLOCK DIAGRAM



15.1 PWM Time Base

The PWM time base is provided by a 15-bit timer with a prescaler and postscaler. The time base is accessible via the PTMR SFR. PTMR<15> is a read-only status bit, PTDIR, that indicates the present count direction of the PWM time base. If PTDIR is cleared, PTMR is counting upwards. If PTDIR is set, PTMR is counting downwards. The PWM time base is configured via the PTCN SFR. The time base is enabled/disabled by setting/clearing the PTEN bit in the PTCN SFR. PTMR is not cleared when the PTEN bit is cleared in software.

The PTPER SFR sets the counting period for PTMR. The user must write a 15-bit value to PTPER<14:0>. When the value in PTMR<14:0> matches the value in PTPER<14:0>, the time base will either reset to '0', or reverse the count direction on the next occurring clock cycle. The action taken depends on the operating mode of the time base.

Note: If the Period register is set to 0x0000, the timer will stop counting, and the interrupt and special event trigger will not be generated even if the special event value is also 0x0000. The module will not update the Period register if it is already at 0x0000; therefore, the user must disable the module in order to update the Period register.

The PWM time base can be configured for four different modes of operation:

- Free-Running mode
- Single-Shot mode
- Continuous Up/Down Count mode
- Continuous Up/Down Count mode with interrupts for double updates

These four modes are selected by the PTMOD<1:0> bits in the PTCN SFR. The Continuous Up/Down Count modes support center-aligned PWM generation. The Single-Shot mode allows the PWM module to support pulse control of certain Electronically Commutative Motors (ECMs).

The interrupt signals generated by the PWM time base depend on the mode selection bits (PTMOD<1:0>) and the postscaler bits (PTOPS<3:0>) in the PTCN SFR.

15.1.1 FREE-RUNNING MODE

In the Free-Running mode, the PWM time base counts upwards until the value in the Time Base Period register (PTPER) is matched. The PTMR register is reset on the following input clock edge and the time base will continue to count upwards as long as the PTEN bit remains set.

When the PWM time base is in the Free-Running mode (PTMOD<1:0> = 00), an interrupt event is generated each time a match with the PTPER register occurs and the PTMR register is reset to zero. The postscaler selection bits may be used in this mode of the timer to reduce the frequency of the interrupt events.

15.1.2 SINGLE-SHOT MODE

In the Single-Shot mode, the PWM time base begins counting upwards when the PTEN bit is set. When the value in the PTMR register matches the PTPER register, the PTMR register will be reset on the following input clock edge and the PTEN bit will be cleared by the hardware to halt the time base.

When the PWM time base is in the Single-Shot mode (PTMOD<1:0> = 01), an interrupt event is generated when a match with the PTPER register occurs, the PTMR register is reset to zero on the following input clock edge and the PTEN bit is cleared. The postscaler selection bits have no effect in this mode of the timer.

15.1.3 CONTINUOUS UP/DOWN COUNT MODES

In the Continuous Up/Down Count modes, the PWM time base counts upwards until the value in the PTPER register is matched. The timer will begin counting downwards on the following input clock edge. The PTDIR bit in the PTMR SFR is read-only and indicates the counting direction. The PTDIR bit is set when the timer counts downwards.

In the Continuous Up/Down Count mode (PTMOD<1:0> = 10), an interrupt event is generated each time the value of the PTMR register becomes zero and the PWM time base begins to count upwards. The postscaler selection bits may be used in this mode of the timer to reduce the frequency of the interrupt events.

dsPIC30F4011/4012

15.1.4 DOUBLE UPDATE MODE

In the Double Update mode (PTMOD<1:0> = 11), an interrupt event is generated each time the PTMR register is equal to zero, as well as each time a period match occurs. The postscaler selection bits have no effect in this mode of the timer.

The Double Update mode provides two additional functions to the user. First, the control loop bandwidth is doubled because the PWM duty cycles can be updated, twice per period. Second, asymmetrical center-aligned PWM waveforms can be generated which are useful for minimizing output waveform distortion in certain motor control applications.

Note: Programming a value of 0x0001 in the Period register could generate a continuous interrupt pulse, and hence, must be avoided.

15.1.5 PWM TIME BASE PRESCALER

The input clock to PTMR (Fosc/4) has prescaler options of 1:1, 1:4, 1:16 or 1:64, selected by control bits, PTCKPS<1:0>, in the PTCON SFR. The prescaler counter is cleared when any of the following occurs:

- a write to the PTMR register
- a write to the PTCON register
- any device Reset

The PTMR register is not cleared when PTCON is written.

15.1.6 PWM TIME BASE POSTSCALER

The match output of PTMR can optionally be post-scaled through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling).

The postscaler counter is cleared when any of the following occurs:

- a write to the PTMR register
- a write to the PTCON register
- any device Reset

The PTMR register is not cleared when PTCON is written.

15.2 PWM Period

PTPER is a 15-bit register and is used to set the counting period for the PWM time base. PTPER is a double-buffered register. The PTPER buffer contents are loaded into the PTPER register at the following instants:

- Free-Running and Single-Shot modes: When the PTMR register is reset to zero after a match with the PTPER register.
- Continuous Up/Down Count modes: When the PTMR register is zero.

The value held in the PTPER buffer is automatically loaded into the PTPER register when the PWM time base is disabled (PTEN = 0).

The PWM period can be determined using Equation 15-1:

EQUATION 15-1: PWM PERIOD

$$T_{PWM} = \frac{T_{CY} \cdot (PTPER + 1)}{(\text{PTMR Prescale Value})}$$

If the PWM time base is configured for one of the Continuous Up/Down Count modes, the PWM period is provided by Equation 15-2.

EQUATION 15-2: PWM PERIOD (CENTER-ALIGNED MODE)

$$T_{PWM} = \frac{2 T_{CY} \cdot PTPER + 0.75}{(\text{PTMR Prescale Value})}$$

The maximum resolution (in bits) for a given device oscillator and PWM frequency can be determined using Equation 15-3:

EQUATION 15-3: PWM RESOLUTION

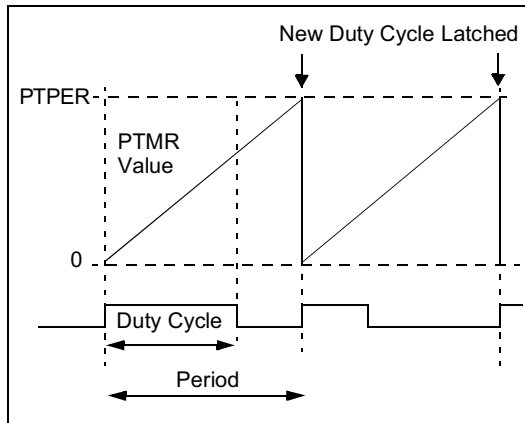
$$\text{Resolution} = \frac{\log(2 \cdot T_{PWM}/T_{CY})}{\log(2)}$$

15.3 Edge-Aligned PWM

Edge-aligned PWM signals are produced by the module when the PWM time base is in the Free-Running or Single-Shot mode. For edge-aligned PWM outputs, the output has a period specified by the value in PTPER and a duty cycle specified by the appropriate duty cycle register (see Figure 15-2). The PWM output is driven active at the beginning of the period ($PTMR = 0$) and is driven inactive when the value in the duty cycle register matches PTMR.

If the value in a particular duty cycle register is zero, then the output on the corresponding PWM pin will be inactive for the entire PWM period. In addition, the output on the PWM pin will be active for the entire PWM period if the value in the duty cycle register is greater than the value held in the PTPER register.

FIGURE 15-2: EDGE-ALIGNED PWM



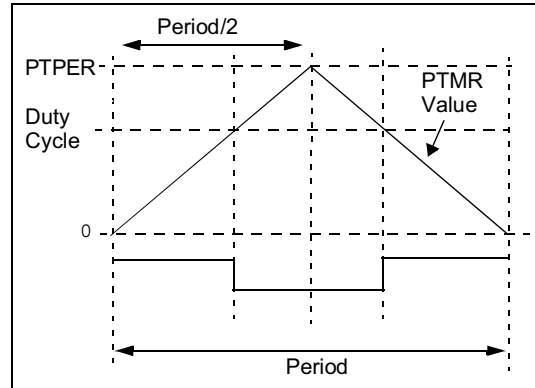
15.4 Center-Aligned PWM

Center-aligned PWM signals are produced by the module when the PWM time base is configured in a Continuous Up/Down Count mode (see Figure 15-3).

The PWM compare output is driven to the active state when the value of the duty cycle register matches the value of PTMR and the PWM time base is counting downwards ($PTDIR = 1$). The PWM compare output is driven to the inactive state when the PWM time base is counting upwards ($PTDIR = 0$) and the value in the PTMR register matches the duty cycle value.

If the value in a particular duty cycle register is zero, then the output on the corresponding PWM pin will be inactive for the entire PWM period. In addition, the output on the PWM pin will be active for the entire PWM period if the value in the duty cycle register is equal to the value held in the PTPER register.

FIGURE 15-3: CENTER-ALIGNED PWM



dsPIC30F4011/4012

15.5 PWM Duty Cycle Comparison Units

There are three 16-bit Special Function Registers (PDC1, PDC2 and PDC3) used to specify duty cycle values for the PWM module.

The value in each duty cycle register determines the amount of time that the PWM output is in the active state. The duty cycle registers are 16-bits wide. The LSB of a duty cycle register determines whether the PWM edge occurs in the beginning. Thus, the PWM resolution is effectively doubled.

15.5.1 DUTY CYCLE REGISTER BUFFERS

The three PWM duty cycle registers are double-buffered to allow glitchless updates of the PWM outputs. For each duty cycle, there is a duty cycle register that is accessible by the user and a second duty cycle register that holds the actual compare value used in the present PWM period.

For edge-aligned PWM output, a new duty cycle value will be updated whenever a match with the PTPER register occurs and PTMR is reset. The contents of the duty cycle buffers are automatically loaded into the duty cycle registers when the PWM time base is disabled (PTEN = 0) and the UDIS bit is cleared in PWMCON2.

When the PWM time base is in the Continuous Up/Down Count mode, new duty cycle values are updated when the value of the PTMR register is zero and the PWM time base begins to count upwards. The contents of the duty cycle buffers are automatically loaded into the duty cycle registers when the PWM time base is disabled (PTEN = 0).

When the PWM time base is in the Continuous Up/Down Count mode with double updates, new duty cycle values are updated when the value of the PTMR register is zero, and when the value of the PTMR register matches the value in the PTPER register. The contents of the duty cycle buffers are automatically loaded into the duty cycle registers when the PWM time base is disabled (PTEN = 0).

15.6 Complementary PWM Operation

In the Complementary mode of operation, each pair of PWM outputs is obtained by a complementary PWM signal. A dead time may be optionally inserted during device switching when both outputs are inactive for a short period (refer to **Section 15.7 “Dead-Time Generators”**).

In Complementary mode, the duty cycle comparison units are assigned to the PWM outputs as follows:

- PDC1 register controls PWM1H/PWM1L outputs
- PDC2 register controls PWM2H/PWM2L outputs
- PDC3 register controls PWM3H/PWM3L outputs

The Complementary mode is selected for each PWM I/O pin pair by clearing the appropriate PTMODx bit in the PWMCON1 SFR. The PWM I/O pins are set to Complementary mode by default upon a device Reset.

15.7 Dead-Time Generators

Dead-time generation may be provided when any of the PWM I/O pin pairs are operating in the Complementary Output mode. The PWM outputs use push-pull drive circuits. Due to the inability of the power output devices to switch instantaneously, some amount of time must be provided between the turn-off event of one PWM output in a complementary pair and the turn-on event of the other transistor.

The PWM module allows two different dead times to be programmed. These two dead times may be used in one of two methods described below to increase user flexibility:

- The PWM output signals can be optimized for different turn-off times in the high side and low side transistors in a complementary pair of transistors. The first dead time is inserted between the turn-off event of the lower transistor of the complementary pair and the turn-on event of the upper transistor. The second dead time is inserted between the turn-off event of the upper transistor and the turn-on event of the lower transistor.
- The two dead times can be assigned to individual PWM I/O pin pairs. This operating mode allows the PWM module to drive different transistor/load combinations with each complementary PWM I/O pin pair.

15.7.1 DEAD-TIME GENERATORS

Each complementary output pair for the PWM module has a 6-bit down counter that is used to produce the dead-time insertion. As shown in Figure 15-4, each dead-time unit has a rising and falling edge detector connected to the duty cycle comparison output.

15.7.2 DEAD-TIME RANGES

The amount of dead time provided by the dead-time unit is selected by specifying the input clock prescaler value and a 6-bit unsigned value.

Four input clock prescaler selections have been provided to allow a suitable range of dead time based on the device operating frequency. The dead-time clock prescaler values are selected using the

DTAPS<1:0> control bits in the DTCON1 SFR. One of four clock prescaler options (T_{CY} , $2 T_{CY}$, $4 T_{CY}$ or $8 T_{CY}$) may be selected.

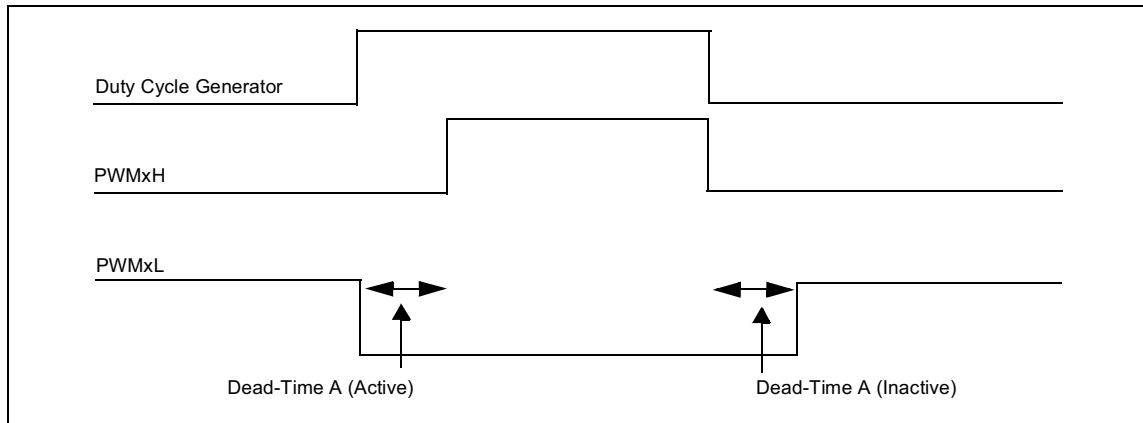
After the prescaler value is selected, the dead time is adjusted by loading 6-bit unsigned values into the DTCON1 SFR.

The dead-time unit prescaler is cleared on the following events:

- On a load of the down timer due to a duty cycle comparison edge event.
- On a write to the DTCON1 register.
- On any device Reset.

Note: The user should not modify the DTCON1 value while the PWM module is operating ($PTEN = 1$). Unexpected results may occur.

FIGURE 15-4: DEAD-TIME TIMING DIAGRAM



dsPIC30F4011/4012

15.8 Independent PWM Output

An Independent PWM Output mode is required for driving certain types of loads. A particular PWM output pair is in the Independent Output mode when the corresponding PTMODx bit in the PWMCON1 register is set. No dead-time control is implemented between adjacent PWM I/O pins when the module is operating in the Independent Output mode and both I/O pins are allowed to be active simultaneously.

In the Independent Output mode, each duty cycle generator is connected to both of the PWM I/O pins in an output pair. By using the associated duty cycle register and the appropriate bits in the OVDCON register, the user may select the following signal output options for each PWM I/O pin operating in the Independent Output mode:

- I/O pin outputs PWM signal
- I/O pin inactive
- I/O pin active

15.9 Single Pulse PWM Operation

The PWM module produces single pulse outputs when the PTCON control bits, PTMOD<1:0> = 10. Only edge-aligned outputs may be produced in the Single Pulse mode. In Single Pulse mode, the PWM I/O pin(s) are driven to the active state when the PTEN bit is set. When a match with a duty cycle register occurs, the PWM I/O pin is driven to the inactive state. When a match with the PTPER register occurs, the PTMR register is cleared, all active PWM I/O pins are driven to the inactive state, the PTEN bit is cleared and an interrupt is generated.

15.10 PWM Output Override

The PWM output override bits allow the user to manually drive the PWM I/O pins to specified logic states, independent of the duty cycle comparison units.

All control bits associated with the PWM output override function are contained in the OVDCON register. The upper half of the OVDCON register contains six bits, POVDxH<3:1> and POVDxL<3:1>, that determine which PWM I/O pins will be overridden. The lower half of the OVDCON register contains six bits, POUTxH<3:1> and POUTxL<3:1>, that determine the state of the PWM I/O pins when a particular output is overridden via the POVD bits.

15.10.1 COMPLEMENTARY OUTPUT MODE

When a PWMxL pin is driven active via the OVDCON register, the output signal is forced to be the complement of the corresponding PWMxH pin in the pair. Dead-time insertion is still performed when PWM channels are overridden manually.

15.10.2 OVERRIDE SYNCHRONIZATION

If the OSYNC bit in the PWMCON2 register is set, all output overrides performed via the OVDCON register are synchronized to the PWM time base. Synchronous output overrides occur at the following times:

- Edge-Aligned mode when PTMR is zero.
- Center-Aligned modes when PTMR is zero and when the value of PTMR matches PTPER.

15.11 PWM Output and Polarity Control

There are three device Configuration bits associated with the PWM module that provide PWM output pin control:

- HPOL Configuration bit
- LPOL Configuration bit
- PWMPIN Configuration bit

These three bits in the FPORBOR Configuration register (see **Section 21.0 “System Integration”**) work in conjunction with the six PWM Enable bits (PENxL and PENxH). The Configuration bits and PWM Enable bits ensure that the PWM pins are in the correct states after a device Reset occurs. The PWMPIN Configuration fuse allows the PWM module outputs to be optionally enabled on a device Reset. If PWMPIN = 0, the PWM outputs will be driven to their inactive states at Reset. If PWMPIN = 1 (default), the PWM outputs will be tri-stated. The HPOL bit specifies the polarity for the PWMxH outputs, whereas the LPOL bit specifies the polarity for the PWMxL outputs.

15.11.1 OUTPUT PIN CONTROL

The PEN<3:1>H and PEN<3:1>L control bits in the PWMCON1 SFR enable each high PWM output pin and each low PWM output pin, respectively. If a particular PWM output pin is not enabled, it is treated as a general purpose I/O pin.

15.12 PWM Fault Pin

There is one Fault pin ($\overline{\text{FLT}}_A$) associated with the PWM module. When asserted, these pins can optionally drive each of the PWM I/O pins to a defined state.

15.12.1 FAULT PIN ENABLE BITS

The FLTACON SFR has 3 control bits that determine whether a particular pair of PWM I/O pins is to be controlled by the Fault input pin. To enable a specific PWM I/O pin pair for Fault overrides, the corresponding bit should be set in the FLTACON register.

If all enable bits are cleared in the FLTACON register, then the corresponding Fault input pin has no effect on the PWM module and the pin may be used as a general purpose interrupt or I/O pin.

Note: The Fault pin logic can operate independent of the PWM logic. If all the enable bits in the FLTACON register are cleared, then the Fault pin could be used as a general purpose interrupt pin. The Fault pin has an interrupt vector, interrupt flag bit and interrupt priority bits associated with it.

15.12.2 FAULT STATES

The FLTACON Special Function Register has 6 bits that determine the state of each PWM I/O pin when it is overridden by a Fault input. When these bits are cleared, the PWM I/O pin is driven to the inactive state. If the bit is set, the PWM I/O pin will be driven to the active state. The active and inactive states are referenced to the polarity defined for each PWM I/O pin (HPOL and LPOL polarity control bits).

A special case exists when a PWM module I/O pair is in the Complementary mode and both pins are programmed to be active on a Fault condition. The PWMxH pin always has priority in the Complementary mode, so that both I/O pins cannot be driven active simultaneously.

15.12.3 FAULT INPUT MODES

The Fault input pin has two modes of operation:

- **Latched Mode:** When the Fault pin is driven low, the PWM outputs will go to the states defined in the FLTACON register. The PWM outputs will remain in this state until the Fault pin is driven high and the corresponding interrupt flag has been cleared in software. When both of these actions have occurred, the PWM outputs will return to normal operation at the beginning of the next PWM cycle or half-cycle boundary. If the interrupt flag is cleared before the Fault condition ends, the PWM module will wait until the Fault pin is no longer asserted to restore the outputs.
- **Cycle-by-Cycle Mode:** When the Fault input pin is driven low, the PWM outputs remain in the defined Fault states for as long as the Fault pin is held low. After the Fault pin is driven high, the PWM outputs return to normal operation at the beginning of the following PWM cycle or half-cycle boundary.

The operating mode for the Fault input pin is selected using the FLTAM control bit in the FLTACON Special Function Register.

The Fault pin can be controlled manually in software.

dsPIC30F4011/4012

15.13 PWM Update Lockout

For a complex PWM application, the user may need to write up to three duty cycle registers and the Time Base Period register, PTPER, at a given time. In some applications, it is important that all buffer registers be written before the new duty cycle and period values are loaded for use by the module.

The PWM update lockout feature is enabled by setting the UDIS control bit in the PWMCON2 SFR. The UDIS bit affects all duty cycle buffer registers and the PWM Time Base Period buffer, PTPER. No duty cycle changes or period value changes will have effect while UDIS = 1.

15.14 PWM Special Event Trigger

The PWM module has a special event trigger that allows A/D conversions to be synchronized to the PWM time base. The A/D sampling and conversion time may be programmed to occur at any point within the PWM period. The special event trigger allows the user to minimize the delay between the time when A/D conversion results are acquired and the time when the duty cycle value is updated.

The PWM special event trigger has an SFR named SEVTCMP and five control bits to control its operation. The PTMR value for which a special event trigger should occur is loaded into the SEVTCMP register. When the PWM time base is in a Continuous Up/Down Count mode, an additional control bit is required to specify the counting phase for the special event trigger. The count phase is selected using the SEVTDIR control bit in the SEVTCMP SFR. If the SEVTDIR bit is cleared, the special event trigger will occur on the upward counting cycle of the PWM time base. If the SEVTDIR bit is set, the special event trigger will occur on the downward count cycle of the PWM time base. The SEVTDIR control bit has no effect unless the PWM time base is configured for a Continuous Up/Down Count mode.

15.14.1 SPECIAL EVENT TRIGGER POSTSCALER

The PWM special event trigger has a postscaler that allows a 1:1 to 1:16 postscale ratio. The postscaler is configured by writing the SEVOPS<3:0> control bits in the PWMCON2 SFR.

The special event output postscaler is cleared on the following events:

- Any write to the SEVTCMP register
- Any device Reset

15.15 PWM Operation During CPU Sleep Mode

The Fault A input pin has the ability to wake the CPU from Sleep mode. The PWM module generates an interrupt if the Fault pin is driven low while in Sleep.

15.16 PWM Operation During CPU Idle Mode

The PTCON SFR contains a PTSIDL control bit. This bit determines if the PWM module will continue to operate or stop when the device enters Idle mode. If PTSIDL = 0, the module will continue to operate. If PTSIDL = 1, the module will stop operation as long as the CPU remains in Idle mode.

TABLE 15-1: 6-OUTPUT PWM REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|---------|--------|--------|--------|--------|----------------------------|--------|--------|------------|------------------------------------|-------|--------|-------------|------------|--------|--------|---------------------|
| PTCON | 01C0 | PTEN | — | PTSIDL | — | — | — | — | — | — | PTOPS<3:0> | | — | PTCKPS<1:0> | PTMOD<1:0> | — | — | 0000 0000 0000 0000 |
| PTMR | 01C2 | PTDIR | — | — | — | — | — | — | — | — | PWM Timer Count Value | | — | — | — | — | — | 0000 0000 0000 0000 |
| PTPER | 01C4 | — | — | — | — | — | — | — | — | — | PWM Time Base Period Register | | — | — | — | — | — | 0111 1111 1111 1111 |
| SEVTCMP | 01C6 | SEVTDIR | — | — | — | — | — | — | — | — | PWM Special Event Compare Register | | — | — | — | — | — | 0000 0000 0000 0000 |
| PWMCON1 | 01C8 | — | — | — | — | — | PTMOD3 | PTMOD2 | PTMOD1 | — | PEN3H | PEN2H | PEN1H | — | PEN3L | PEN2L | PEN1L | 0000 0000 1111 1111 |
| PWMCON2 | 01CA | — | — | — | — | — | SEVOPS<3:0> | | — | — | — | — | — | — | — | OSYNC | UDIS | 0000 0000 0000 0000 |
| DTCON1 | 01CC | — | — | — | — | — | — | — | — | DTAPS<1:0> | Dead-Time A Value | | — | — | — | — | — | 0000 0000 0000 0000 |
| FLTACON | 01D0 | — | — | FAOV3H | FAOV3L | FAOV2H | FAOV2L | FAOV1H | FAOV1L | FLTAM | — | — | — | — | FAEN3 | FAEN2 | FAEN1 | 0000 0000 0000 0000 |
| OVDCON | 01D4 | — | POVD3H | POVD3L | POVD2H | POVD2L | POVD1H | POVD1L | POVD1L | — | — | — | POUT3H | POUT2H | POUT2L | POUT1H | POUT1L | 1111 1111 0000 0000 |
| PDC1 | 01D6 | — | — | — | — | — | PWM Duty Cycle #1 Register | | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| PDC2 | 01D8 | — | — | — | — | — | PWM Duty Cycle #2 Register | | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| PDC3 | 01DA | — | — | — | — | — | PWM Duty Cycle #3 Register | | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

16.0 SPI MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

The Serial Peripheral Interface (SPI) module is a synchronous serial interface. It is useful for communicating with other peripheral devices, such as EEPROMs, shift registers, display drivers and A/D converters, or other microcontrollers. It is compatible with Motorola's SPI and SIOP interfaces.

16.1 Operating Function Description

The SPI module consists of a 16-bit shift register, SPI1SR, used for shifting data in and out, and a buffer register, SPI1BUF. A control register, SPI1CON, configures the module. Additionally, a status register, SPI1STAT, indicates various status conditions.

The serial interface consists of 4 pins: SDI1 (Serial Data Input), SDO1 (Serial Data Output), SCK1 (Shift Clock Input or Output), and SS1 (active-low Slave Select).

In Master mode operation, SCK1 is a clock output, but in Slave mode, it is a clock input.

A series of eight (8) or sixteen (16) clock pulses shift out bits from the SPI1SR to SDO1 pin, and simultaneously, shifts in data from SDI1 pin. An interrupt is generated when the transfer is complete and the corresponding interrupt flag bit (SPI1IF) is set. This interrupt can be disabled through an interrupt enable bit (SPI1IE).

The receive operation is double-buffered. When a complete byte is received, it is transferred from SPI1SR to SPI1BUF.

If the receive buffer is full when new data is being transferred from SPI1SR to SPI1BUF, the module will set the SPIROV bit, indicating an overflow condition. The transfer of the data from SPI1SR to SPI1BUF will not be completed and the new data will be lost. The module will not respond to SCL transitions while SPIROV is '1', effectively disabling the module until SPI1BUF is read by user software.

Transmit writes are also double-buffered. The user writes to SPI1BUF. When the master or slave transfer is completed, the contents of the shift register (SPI1SR) is moved to the receive buffer. If any transmit data has been written to the buffer register, the contents of the transmit buffer are moved to SPI1SR. The received data is thus placed in SPI1BUF and the transmit data in SPI1SR is ready for the next transfer.

Note: Both the transmit buffer (SPI1TXB) and the receive buffer (SPI1RXB) are mapped to the same register address, SPI1BUF.

In Master mode, the clock is generated by prescaling the system clock. Data is transmitted as soon as a value is written to SPI1BUF. The interrupt is generated at the middle of the transfer of the last bit.

In Slave mode, data is transmitted and received as external clock pulses appear on SCK1. Again, the interrupt is generated when the last bit is latched. If SS1 control is enabled, then transmission and reception are enabled only when SS1 = low. The SDO1 output will be disabled in SS1 mode with SS1 high.

The clock provided to the module is (FOSC/4). This clock is then prescaled by the primary (PPRE<1:0>) and secondary (SPRE<2:0>) prescale factors. The CKE bit determines whether transmit occurs on transition from active clock state to Idle clock state, or vice versa. The CKP bit selects the Idle state (high or low) for the clock.

16.1.1 WORD AND BYTE COMMUNICATION

A control bit, MODE16 (SPI1CON<10>), allows the module to communicate in either 16-bit or 8-bit mode. 16-bit operation is identical to 8-bit operation, except that the number of bits transmitted is 16 instead of 8.

The user software must disable the module prior to changing the MODE16 bit. The SPI module is reset when the MODE16 bit is changed by the user.

A basic difference between 8-bit and 16-bit operation is that the data is transmitted out of bit 7 of the SPI1SR for 8-bit operation, and data is transmitted out of bit 15 of the SPI1SR for 16-bit operation. In both modes, data is shifted into bit 0 of the SPI1SR.

16.1.2 SDO1 DISABLE

A control bit, DISSDO, is provided to the SPI1CON register to allow the SDO1 output to be disabled. This will allow the SPI module to be connected in an input only configuration. SDO1 can also be used for general purpose I/O.

16.2 Framed SPI Support

The module supports a basic framed SPI protocol in Master or Slave mode. The control bit, FRMEN, enables framed SPI support and causes the SS1 pin to perform the frame synchronization pulse (FSYNC) function. The control bit, SPIFSD, determines whether the SS1 pin is an input or an output (i.e., whether the module receives or generates the frame synchronization pulse). The frame pulse is an active-high pulse for a single SPI clock cycle. When frame synchronization is enabled, the data transmission starts only on the subsequent transmit edge of the SPI clock.

dsPIC30F4011/4012

FIGURE 16-1: SPI BLOCK DIAGRAM

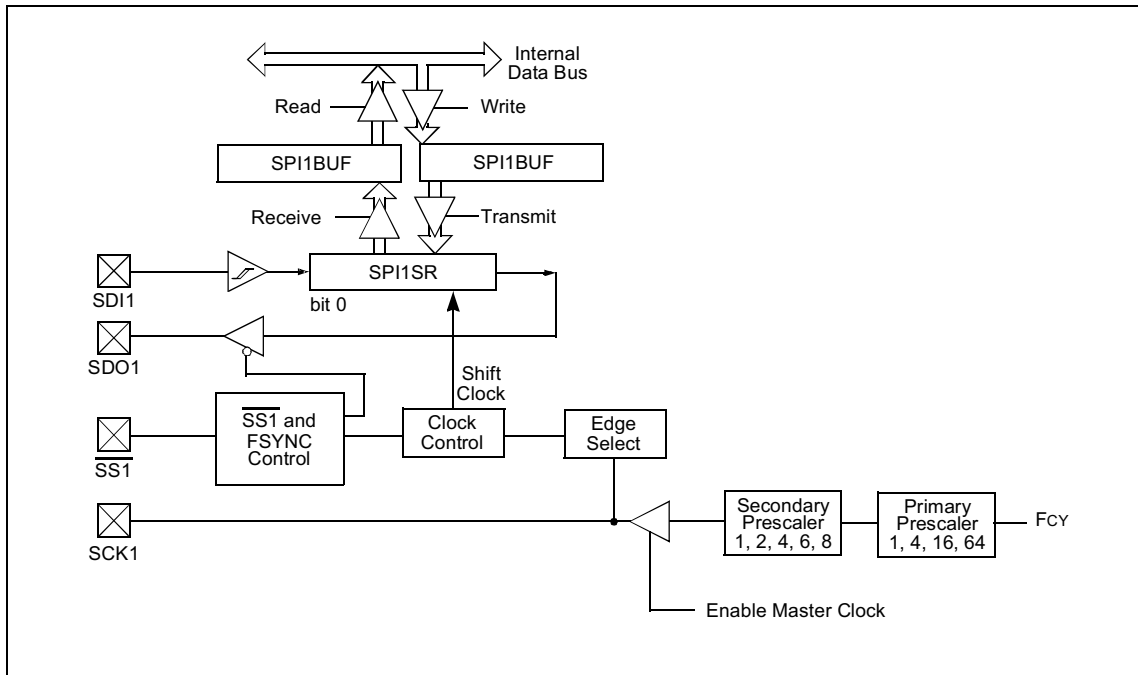
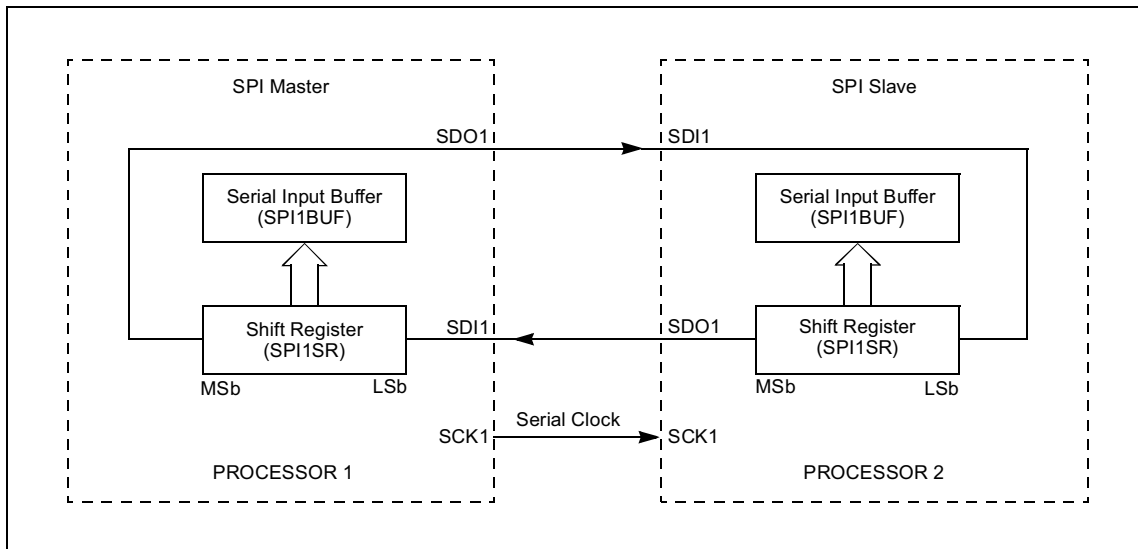


FIGURE 16-2: SPI MASTER/SLAVE CONNECTION



16.3 Slave Select Synchronization

The $\overline{SS1}$ pin allows a Synchronous Slave mode. The SPI must be configured in SPI Slave mode, with $\overline{SS1}$ pin control enabled ($SSEN = 1$). When the $\overline{SS1}$ pin is low, transmission and reception are enabled and the SDO1 pin is driven. When $\overline{SS1}$ pin goes high, the SDO1 pin is no longer driven. Also, the SPI module is resynchronized and all counters/control circuitry are reset. Therefore, when the $\overline{SS1}$ pin is asserted low again, transmission/reception will begin at the MSb, even if $\overline{SS1}$ had been deasserted in the middle of a transmit/receive.

16.4 SPI Operation During CPU Sleep Mode

During Sleep mode, the SPI module is shut down. If the CPU enters Sleep mode while an SPI transaction is in progress, then the transmission and reception is aborted.

The transmitter and receiver will stop in Sleep mode. However, register contents are not affected by entering or exiting Sleep mode.

16.5 SPI Operation During CPU Idle Mode

When the device enters Idle mode, all clock sources remain functional. The SPISIDL bit ($SPI1STAT<13>$) selects if the SPI module will stop or continue on Idle. If $SPISIDL = 0$, the module will continue to operate when the CPU enters Idle mode. If $SPISIDL = 1$, the module will stop when the CPU enters Idle mode.

dsPIC30F4011/4012

TABLE 16-1: SPI REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|-----------------------------|--------|--------|--------|--------|--------|-------|-------|-------|--------|-------|-------|-------|-------|--------|--------|---------------------|
| SPI1STAT | 0220 | SPIEN | — | SPIIDL | — | — | — | — | — | — | SPIROV | — | — | — | — | SPITBF | SPIRBF | 0000 0000 0000 0000 |
| SPI1CON | 0222 | — | FRMEN | SPIFSD | — | DISSDO | MODE16 | SMP | CKE | SSEN | CKP | MSTEN | SPRE2 | SPRE1 | SPRE0 | PPRE1 | PPRE0 | 0000 0000 0000 0000 |
| SPI1BUF | 0224 | Transmit and Receive Buffer | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

17.0 I²C™ MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

The Inter-Integrated Circuit (I²C) module provides complete hardware support for both Slave and Multi-Master modes of the I²C serial communication standard with a 16-bit interface.

This module offers the following key features:

- I²C Interface supporting both Master and Slave Operation.
- I²C Slave mode supports 7 and 10-bit addressing.
- I²C Master mode supports 7 and 10-bit addressing.
- I²C Port allows Bidirectional Transfers between Master and Slaves.
- Serial Clock Synchronization for I²C Port can be used as a Handshake Mechanism to Suspend and Resume Serial Transfer (SCLREL control).
- I²C supports Multi-Master Operation; Detects Bus Collision and will Arbitrate accordingly.

17.1 Operating Function Description

The hardware fully implements all the master and slave functions of the I²C Standard and Fast mode specifications, as well as 7 and 10-bit addressing.

Thus, the I²C module can operate either as a slave or a master on an I²C bus.

17.1.1 VARIOUS I²C MODES

The following types of I²C operation are supported:

- I²C slave operation with 7-bit addressing.
- I²C slave operation with 10-bit addressing.
- I²C master operation with 7 or 10-bit addressing.

See the I²C programmer's model in Figure 17-1.

17.1.2 PIN CONFIGURATION IN I²C MODE

I²C has a 2-pin interface; SCL pin is clock and SDA pin is data.

17.1.3 I²C REGISTERS

I2CCON and I2CSTAT are control and status registers, respectively. The I2CCON register is readable and writable. The lower 6 bits of I2CSTAT are read-only. The remaining bits of the I2CSTAT are read/write.

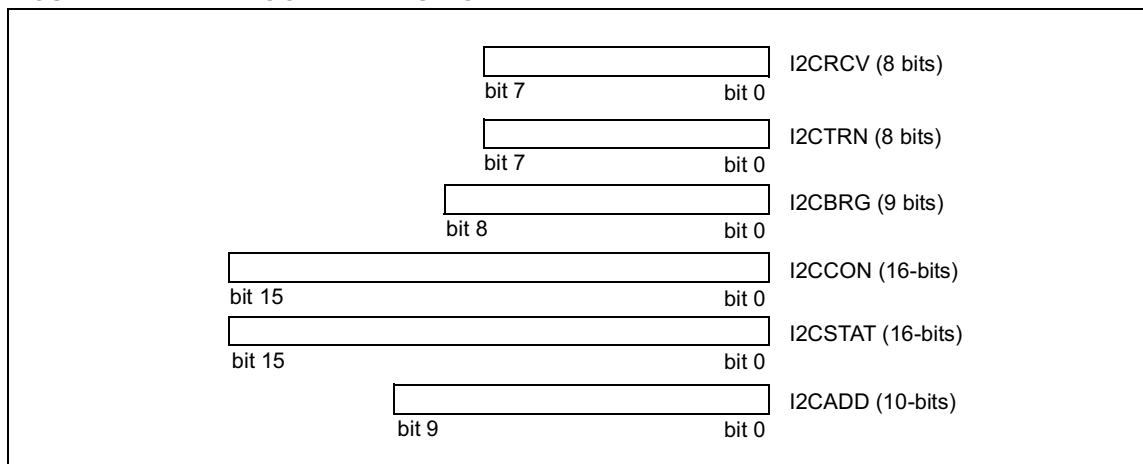
I2CRSR is the shift register used for shifting data, whereas I2CRCV is the buffer register to which data bytes are written or from which data bytes are read. I2CRCV is the receive buffer, as shown in Figure 17-1. I2CTRN is the transmit register to which bytes are written during a transmit operation, as shown in Figure 17-2.

The I2CADD register holds the slave address. A status bit, ADD10, indicates 10-bit Address mode. The I2CBRG acts as the Baud Rate Generator reload value.

In receive operations, I2CRSR and I2CRCV together form a double-buffered receiver. When I2CRSR receives a complete byte, it is transferred to I2CRCV and an interrupt pulse is generated. During transmission, the I2CTRN is not double-buffered.

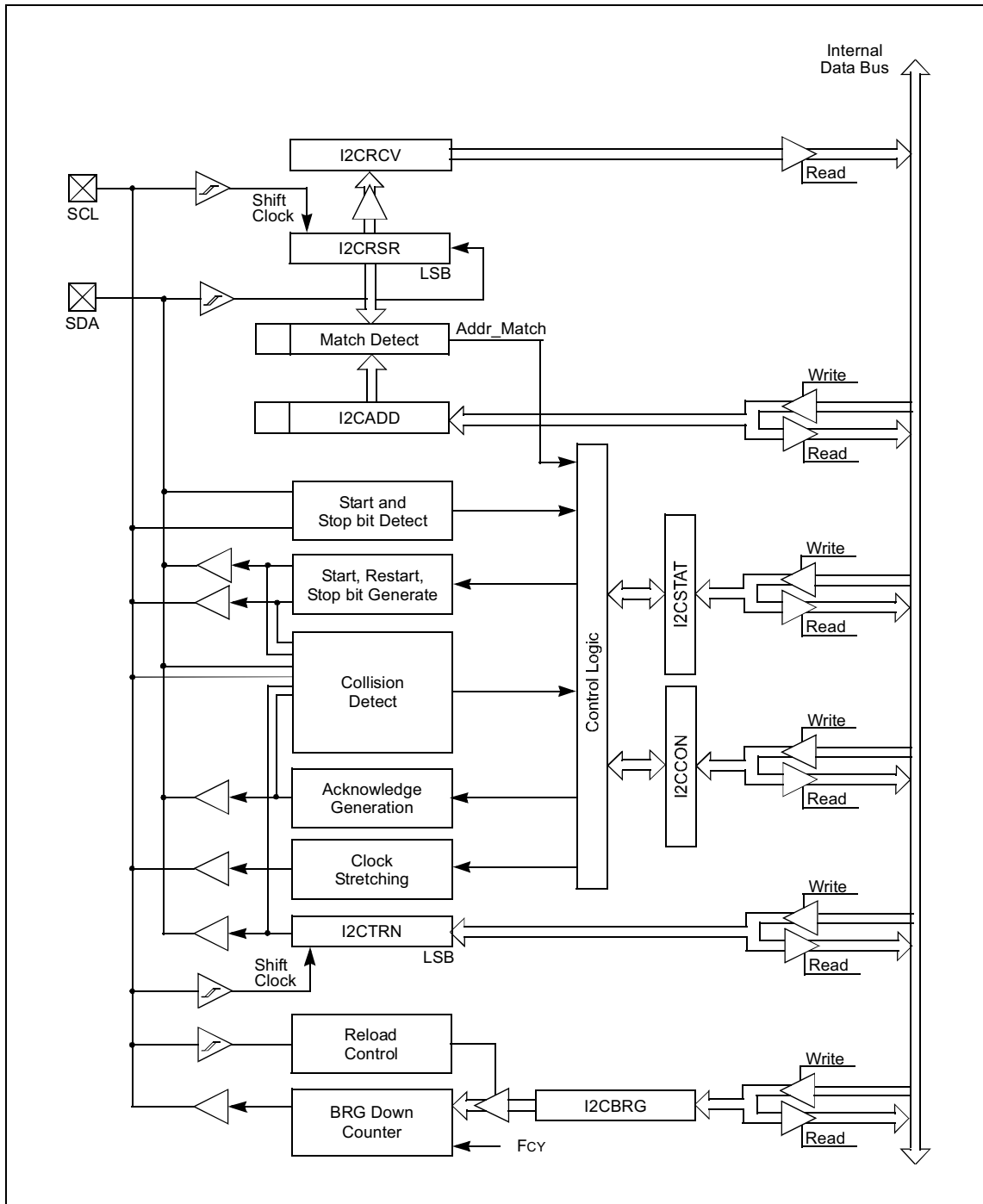
Note: Following a Restart condition in 10-bit mode, the user only needs to match the first 7-bit address.

FIGURE 17-1: PROGRAMMER'S MODEL



dsPIC30F4011/4012

FIGURE 17-2: I²C™ BLOCK DIAGRAM



17.2 I²C Module Addresses

The I2CADD register contains the Slave mode addresses. The register is a 10-bit register.

If the A10M bit (I2CCON<10>) is '0', the address is interpreted by the module as a 7-bit address. When an address is received, it is compared to the 7 LSBs of the I2CADD register.

If the A10M bit is '1', the address is assumed to be a 10-bit address. When an address is received, it will be compared with the binary value, '11110 A₉ A₈' (where A₉ and A₈ are two Most Significant bits of I2CADD). If that value matches, the next address will be compared with the Least Significant 8 bits of I2CADD, as specified in the 10-bit addressing protocol.

The 7-bit I²C slave addresses supported by the dsPIC30F are shown in Table 17-1.

TABLE 17-1: 7-BIT I²C™ SLAVE ADDRESSES

| | |
|-----------|---------------------------------------|
| 0x00 | General call address or Start byte |
| 0x01-0x03 | Reserved |
| 0x04-0x07 | HS mode master codes |
| 0x08-0x77 | Valid 7-bit addresses |
| 0x78-0x7B | Valid 10-bit addresses (lower 7 bits) |
| 0x7C-0x7F | Reserved |

17.3 I²C 7-bit Slave Mode Operation

Once enabled (I2CEN = 1), the slave module will wait for a Start bit to occur (i.e., the I²C module is 'Idle'). Following the detection of a Start bit, 8 bits are shifted into I2CRSR and the address is compared against I2CADD. In 7-bit mode (A10M = 0), I2CADD<6:0> bits are compared against I2CRSR<7:1>, and I2CRSR<0> is the R_W bit. All incoming bits are sampled on the rising edge of SCL.

If an address match occurs, an Acknowledgement will be sent, and the Slave Event Interrupt Flag (SI2CIF) is set on the falling edge of the ninth (ACK) bit. The address match does not affect the contents of the I2CRCV buffer or the RBF bit.

17.3.1 SLAVE TRANSMISSION

If the R_W bit received is a '1', then the serial port will go into Transmit mode. It will send ACK on the ninth bit and then hold SCL to '0' until the CPU responds by writing to I2CTRN. SCL is released by setting the SCLREL bit and 8 bits of data are shifted out. Data bits are shifted out on the falling edge of SCL, such that SDA is valid during SCL high (see timing diagram). The interrupt pulse is sent on the falling edge of the ninth clock pulse, regardless of the status of the ACK received from the master.

17.3.2 SLAVE RECEPTION

If the R_W bit received is a '0' during an address match, then Receive mode is initiated. Incoming bits are sampled on the rising edge of SCL. After 8 bits are received, if I2CRCV is not full or I2COV is not set, I2CRSR is transferred to I2CRCV. ACK is sent on the ninth clock.

If the RBF flag is set, indicating that I2CRCV is still holding data from a previous operation (RBF = 1), then ACK is not sent; however, the interrupt pulse is generated. In the case of an overflow, the contents of the I2CRSR are not loaded into the I2CRCV.

Note: The I2CRCV will be loaded if the I2COV bit = 1 and the RBF flag = 0. In this case, a read of the I2CRCV was performed but the user did not clear the state of the I2COV bit before the next receive occurred. The Acknowledgement is not sent (ACK = 1) and the I2CRCV is updated.

17.4 I²C 10-bit Slave Mode Operation

In 10-bit mode, the basic receive and transmit operations are the same as in the 7-bit mode. However, the criteria for address match is more complex.

The I²C specification dictates that a slave must be addressed for a write operation, with two address bytes following a Start bit.

The A10M bit is a control bit that signifies that the address in I2CADD is a 10-bit address rather than a 7-bit address. The address detection protocol for the first byte of a message address is identical for 7-bit and 10-bit messages but the bits being compared are different.

I2CADD holds the entire 10-bit address. Upon receiving an address following a Start bit, I2CRSR<7:3> is compared against a literal '11110' (the default 10-bit address) and I2CRSR<2:1> are compared against I2CADD<9:8>. If a match occurs and if R_W = 0, the interrupt pulse is sent. The ADD10 bit will be cleared to indicate a partial address match. If a match fails or R_W = 1, the ADD10 bit is cleared and the module returns to the Idle state.

The low byte of the address is then received and compared with I2CADD<7:0>. If an address match occurs, the interrupt pulse is generated and the ADD10 bit is set, indicating a complete 10-bit address match. If an address match did not occur, the ADD10 bit is cleared and the module returns to the Idle state.

dsPIC30F4011/4012

17.4.1 10-BIT MODE SLAVE TRANSMISSION

Once a slave is addressed in this fashion, with the full 10-bit address (we will refer to this state as "PRIOR_ADDR_MATCH"), the master can begin sending data bytes for a slave reception operation.

17.4.2 10-BIT MODE SLAVE RECEPTION

Once addressed, the master can generate a Repeated Start, reset the high byte of the address and set the R_W bit without generating a Stop bit, thus initiating a slave transmit operation.

17.5 Automatic Clock Stretch

In the Slave modes, the module can synchronize buffer reads and write to the master device by clock stretching.

17.5.1 TRANSMIT CLOCK STRETCHING

Both 10-bit and 7-bit Transmit modes implement clock stretching by asserting the SCLREL bit after the falling edge of the ninth clock if the TBF bit is cleared, indicating the buffer is empty.

In Slave Transmit modes, clock stretching is always performed, irrespective of the STREN bit.

Clock synchronization takes place following the ninth clock of the transmit sequence. If the device samples an $\overline{\text{ACK}}$ on the falling edge of the ninth clock, and if the TBF bit is still clear, then the SCLREL bit is automatically cleared. The SCLREL being cleared to '0' will assert the SCL line low. The user's ISR must set the SCLREL bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the I2CTRN before the master device can initiate another transmit sequence.

Note 1: If the user loads the contents of I2CTRN, setting the TBF bit before the falling edge of the ninth clock, the SCLREL bit will not be cleared and clock stretching will not occur.

2: The SCLREL bit can be set in software regardless of the state of the TBF bit.

17.5.2 RECEIVE CLOCK STRETCHING

The STREN bit in the I2CCON register can be used to enable clock stretching in Slave Receive mode. When the STREN bit is set, the SCL pin will be held low at the end of each data receive sequence.

17.5.3 CLOCK STRETCHING DURING 7-BIT ADDRESSING (STREN = 1)

When the STREN bit is set in Slave Receive mode, the SCL line is held low when the buffer register is full. The method for stretching the SCL output is the same for both 7 and 10-bit Addressing modes.

Clock stretching takes place following the ninth clock of the receive sequence. On the falling edge of the ninth clock at the end of the ACK sequence, if the RBF bit is set, the SCLREL bit is automatically cleared, forcing the SCL output to be held low. The user's ISR must set the SCLREL bit before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the I2CRCV before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring.

Note 1: If the user reads the contents of the I2CRCV, clearing the RBF bit before the falling edge of the ninth clock, the SCLREL bit will not be cleared and clock stretching will not occur.

2: The SCLREL bit can be set in software, regardless of the state of the RBF bit. The user should be careful to clear the RBF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

17.5.4 CLOCK STRETCHING DURING 10-BIT ADDRESSING (STREN = 1)

Clock stretching takes place automatically during the addressing sequence. Because this module has a register for the entire address, it is not necessary for the protocol to wait for the address to be updated.

After the address phase is complete, clock stretching will occur on each data receive or transmit sequence as was described earlier.

17.6 Software Controlled Clock Stretching (STREN = 1)

When the STREN bit is '1', the SCLREL bit may be cleared by software to allow software to control the clock stretching. The logic will synchronize writes to the SCLREL bit with the SCL clock. Clearing the SCLREL bit will not assert the SCL output until the module detects a falling edge on the SCL output and SCL is sampled low. If the SCLREL bit is cleared by the user while the SCL line has been sampled low, the SCL output will be asserted (held low). The SCL output will remain low until the SCLREL bit is set and all other devices on the I²C bus have deasserted SCL. This ensures that a write to the SCLREL bit will not violate the minimum high time requirement for SCL.

If the STREN bit is '0', a software write to the SCLREL bit will be disregarded and have no effect on the SCLREL bit.

17.7 Interrupts

The I²C module generates two interrupt flags, MI2CIF (I²C Master Interrupt Flag) and SI2CIF (I²C Slave Interrupt Flag). The MI2CIF interrupt flag is activated on completion of a master message event. The SI2CIF interrupt flag is activated on detection of a message directed to the slave.

17.8 Slope Control

The I²C standard requires slope control on the SDA and SCL signals for Fast mode (400 kHz). The control bit, DISSLW, enables the user to disable slew rate control, if desired. It is necessary to disable the slew rate control for 1 MHz mode.

17.9 IPMI Support

The control bit, IPMIEN, enables the module to support Intelligent Peripheral Management Interface (IPMI). When this bit is set, the module accepts and acts upon all addresses.

17.10 General Call Address Support

The general call address can address all devices. When this address is used, all devices should, in theory, respond with an Acknowledgement.

The general call address is one of eight addresses reserved for specific purposes by the I²C protocol. It consists of all '0's with R_W = 0.

The general call address is recognized when the General Call Enable (GCEN) bit is set (I2CCON<7> = 1). Following a Start bit detection, 8 bits are shifted into I2CRSR and the address is compared with I2CADD, and is also compared with the general call address which is fixed in hardware.

If a general call address match occurs, the I2CRSR is transferred to the I2CRCV after the eighth clock, the RBF flag is set, and on the falling edge of the ninth bit (ACK bit), the Master Event Interrupt Flag (MI2CIF) is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the I2CRCV to determine if the address was device-specific or a general call address.

17.11 I²C Master Support

As a master device, six operations are supported.

- Assert a Start condition on SDA and SCL.
- Assert a Restart condition on SDA and SCL.
- Write to the I2CTRN register initiating transmission of data/address.
- Generate a Stop condition on SDA and SCL.
- Configure the I²C port to receive data.
- Generate an ACK condition at the end of a received byte of data.

17.12 I²C Master Operation

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this case, the data direction bit (R_W) is logic '0'. Serial data is transmitted 8 bits at a time. After each byte is transmitted, an ACK bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the data direction bit. In this case, the data direction bit (R_W) is 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 SDA while SCL outputs the serial clock. Serial data is received 8 bits at a time. After each byte is received, an ACK bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

17.12.1 I²C MASTER TRANSMISSION

Transmission of a data byte, a 7-bit address or the second half of a 10-bit address, is accomplished by simply writing a value to the I2CTRN register. The user should only write to I2CTRN when the module is in a Wait state. This action will set the buffer full flag (TBF) and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted. The Transmit Status Flag, TRSTAT (I2CSTAT<14>), indicates that a master transmit is in progress.

dsPIC30F4011/4012

17.12.2 I²C MASTER RECEPTION

Master mode reception is enabled by programming the receive enable (RCEN) bit (I2CCON<3>). The I²C module must be Idle before the RCEN bit is set, otherwise the RCEN bit will be disregarded. The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin toggles and data is shifted in to the I2CRSR on the rising edge of each clock.

17.12.3 BAUD RATE GENERATOR (BRG)

In I²C Master mode, the reload value for the BRG is located in the I2CBRG register. When the BRG is loaded with this value, the BRG counts down to '0' and stops until another reload has taken place. If clock arbitration is taking place, for instance, the BRG is reloaded when the SCL pin is sampled high.

As per the I²C standard, F_{SCK} may be 100 kHz or 400 kHz. However, the user can specify any baud rate up to 1 MHz. I2CBRG values of '0' or '1' are illegal.

EQUATION 17-1: I2CBRG VALUE

$$I2CBRG = \left(\frac{FCY}{F_{SCL}} - \frac{FCY}{1,111,111} \right) - 1$$

17.12.4 CLOCK ARBITRATION

Clock arbitration occurs when the master deasserts the SCL pin (SCL allowed to float high) during any receive, transmit or Restart/Stop condition. When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of I2CBRG and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device.

17.12.5 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-master operation support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high, while another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin = 0, then a bus collision has taken place. The master will set the MI2CIF pulse and reset the master portion of the I²C port to its Idle state.

If a transmit was in progress when the bus collision occurred, the transmission is halted, the TBF flag is cleared, the SDA and SCL lines are deasserted and a value can now be written to I2CTRN. When the user services the I²C master event Interrupt Service Routine and if the I²C bus is free (i.e., the P bit is set), the user can resume communication by asserting a Start condition.

If a Start, Restart, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the I2CCON register are cleared to '0'. When the user services the bus collision Interrupt Service Routine, and if the I²C bus is free, the user can resume communication by asserting a Start condition.

The Master will continue to monitor the SDA and SCL pins, and if a Stop condition occurs, the MI2CIF bit will be set.

A write to the I2CTRN will start the transmission of data at the first data bit, regardless of where the transmitter left off when bus collision occurred.

In a Multi-Master environment, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I²C bus can be taken when the P bit is set in the I2CSTAT register, or the bus is Idle and the S and P bits are cleared.

17.13 I²C Module Operation During CPU Sleep and Idle Modes

17.13.1 I²C OPERATION DURING CPU SLEEP MODE

When the device enters Sleep mode, all clock sources to the module are shut down and stay at logic '0'. If Sleep occurs in the middle of a transmission, and the state machine is partially into a transmission as the clocks stop, then the transmission is aborted. Similarly, if Sleep occurs in the middle of a reception, then the reception is aborted.

17.13.2 I²C OPERATION DURING CPU IDLE MODE

For the I²C, the I2CSIDL bit selects if the module will stop on Idle or continue on Idle. If I2CSIDL = 0, the module will continue operation on assertion of the Idle mode. If I2CSIDL = 1, the module will stop on Idle.

TABLE 17-2: I²C™ REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|---------|--------|---------|--------|--------|--------|---------|-------|-------|-------|-------|-------|-------|-------|-------|-------|---------------------|
| I2CRCV | 0200 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| I2CTRN | 0202 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 1111 1111 |
| I2CBRG | 0204 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| I2CCON | 0206 | I2CEN | — | I2CSIDL | SCLREL | IPMIEN | A10M | DISSLW | SMEN | GCEN | STREN | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 0001 0000 0000 0000 |
| I2CSTAT | 0208 | ACKSTAT | TRSTAT | — | — | — | BCL | GCSSTAT | ADD10 | IWCOL | I2COV | D_A | P | S | R_W | RBF | TBF | 0000 0000 0000 0000 |
| I2CADD | 020A | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

dsPIC30F4011/4012

18.0 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART) MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

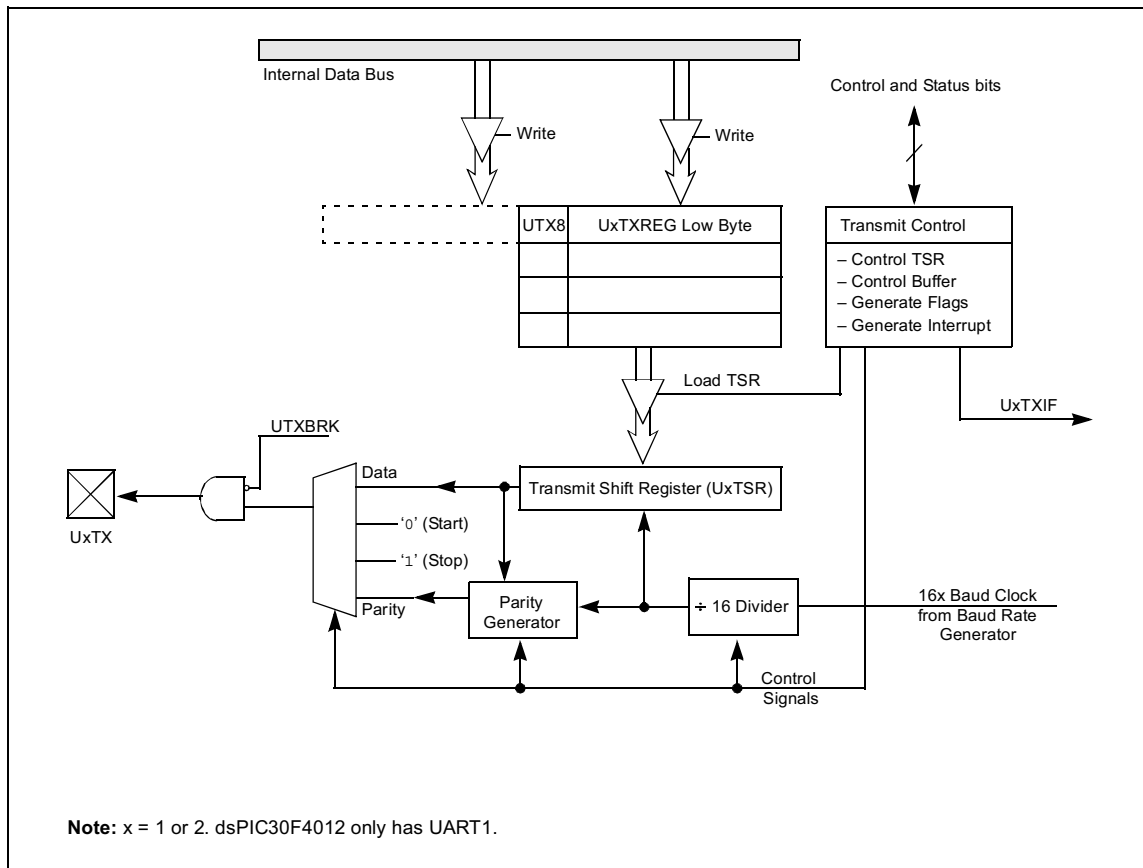
This section describes the Universal Asynchronous Receiver/Transmitter communications module.

18.1 UART Module Overview

The key features of the UART module are:

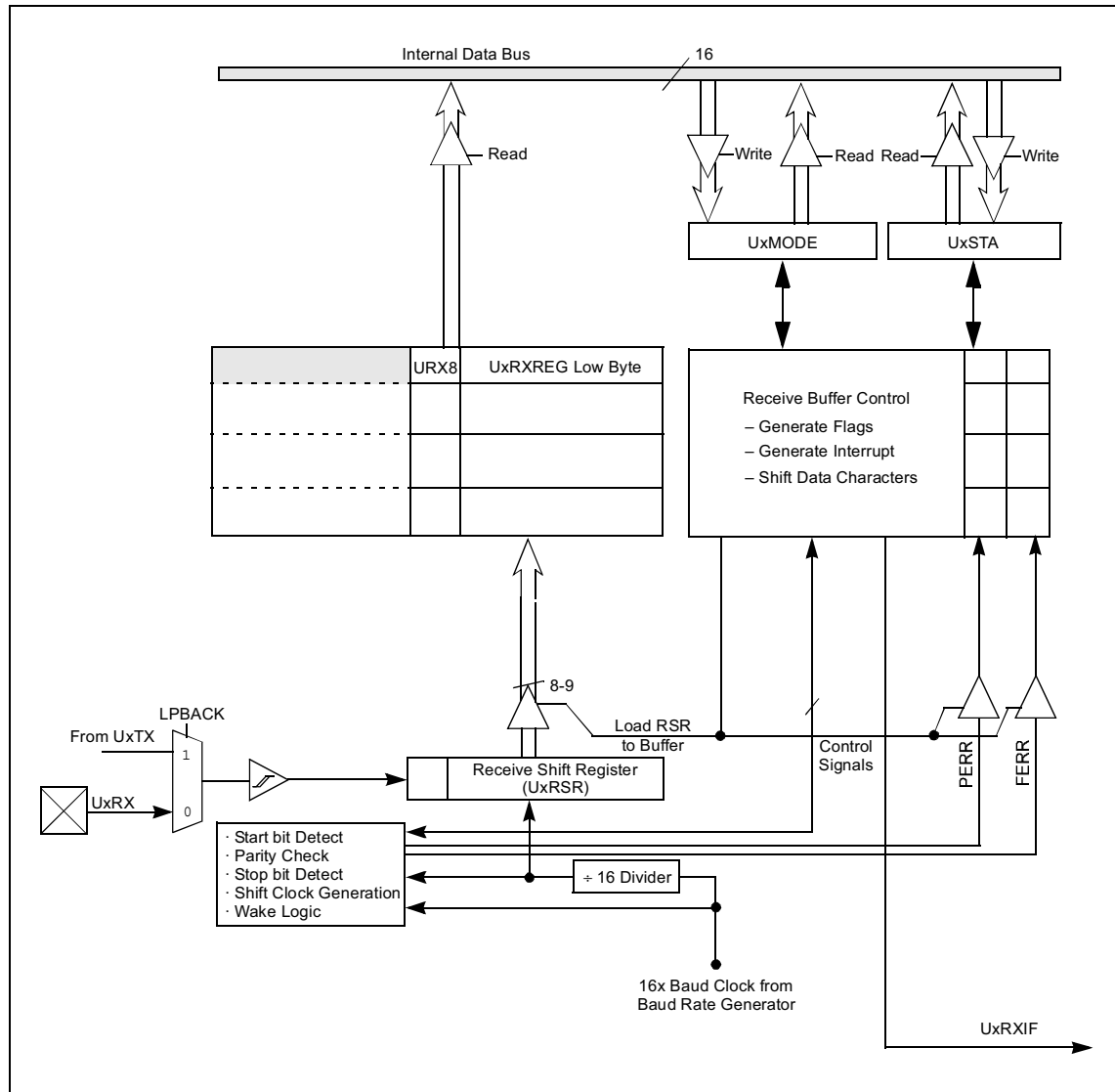
- Full-Duplex, 8 or 9-bit Data Communication
- Even, Odd or No Parity Options (for 8-bit data)
- One or Two Stop bits
- Fully Integrated Baud Rate Generator with 16-bit Prescaler
- Baud Rates ranging from 38 bps to 1.875 Mbps at a 30 MHz Instruction Rate
- 4-Word Deep Transmit Data Buffer
- 4-Word Deep Receive Data Buffer
- Parity, Framing and Buffer Overrun Error Detection
- Support for Interrupt Only on Address Detect (9th bit = 1)
- Separate Transmit and Receive Interrupts
- Loopback mode for Diagnostic Support

FIGURE 18-1: UART TRANSMITTER BLOCK DIAGRAM



dsPIC30F4011/4012

FIGURE 18-2: UART RECEIVER BLOCK DIAGRAM



18.2 Enabling and Setting Up UART

18.2.1 ENABLING THE UART

The UART module is enabled by setting the UARTEN bit in the UxMODE register (where x = 1 or 2). Once enabled, the UxTX and UxRX pins are configured as an output and an input, respectively, overriding the TRIS and LATCH register bit settings for the corresponding I/O port pins. The UxTX pin is at logic '1' when no transmission is taking place.

18.2.2 DISABLING THE UART

The UART module is disabled by clearing the UARTEN bit in the UxMODE register. This is the default state after any Reset. If the UART is disabled, all I/O pins operate as port pins under the control of the LATCH and TRIS bits of the corresponding port pins.

Disabling the UART module resets the buffers to empty states. Any data characters in the buffers are lost and the baud rate counter is reset.

All error and status flags associated with the UART module are reset when the module is disabled. The URXDA, OERR, FERR, PERR, UTXEN, UTXBRK and UTXBF bits are cleared, whereas RIDLE and TRMT are set. Other control bits, including ADDEN, URXISEL<1:0>, UTXISEL, as well as the UxMODE and UxBRG registers, are not affected.

Clearing the UARTEN bit while the UART is active will abort all pending transmissions and receptions and reset the module as defined above. Re-enabling the UART will restart the UART in the same configuration.

18.2.3 ALTERNATE I/O

The alternate I/O function is enabled by setting the ALTIO bit (UxMODE<10>). If ALTIO = 1, the UxATX and UxARX pins (alternate transmit and alternate receive pins, respectively) are used by the UART module instead of the UxTX and UxRX pins. If ALTIO = 0, the UxTX and UxRX pins are used by the UART module.

18.2.4 SETTING UP DATA, PARITY AND STOP BIT SELECTIONS

Control bits, PDSEL<1:0> in the UxMODE register, are used to select the data length and parity used in the transmission. The data length may either be 8 bits with even, odd or no parity, or 9-bits with no parity.

The STSEL bit determines whether one or two Stop bits will be used during data transmission.

The default (power-on) setting of the UART is 8 bits, no parity and 1 Stop bit (typically represented as 8, N, 1).

18.3 Transmitting Data

18.3.1 TRANSMITTING IN 8-BIT DATA MODE

The following steps must be performed in order to transmit 8-bit data:

1. Set up the UART:
First, the data length, parity and number of Stop bits must be selected. Then, the transmit and receive interrupt enable and priority bits are set up in the UxMODE and UxSTA registers. Also, the appropriate baud rate value must be written to the UxBRG register.
2. Enable the UART by setting the UARTEN bit (UxMODE<15>).
3. Set the UTXEN bit (UxSTA<10>), thereby enabling a transmission.
4. Write the byte to be transmitted to the lower byte of UxTXREG. The value will be transferred to the Transmit Shift register (UxTSR) immediately and the serial bit stream will start shifting out during the next rising edge of the baud clock. Alternatively, the data byte may be written while UTXEN = 0, following which, the user may set UTXEN. This will cause the serial bit stream to begin immediately because the baud clock will start from a cleared state.
5. A transmit interrupt will be generated depending on the value of the interrupt control bit UTXISEL (UxSTA<15>).

18.3.2 TRANSMITTING IN 9-BIT DATA MODE

The sequence of steps involved in the transmission of 9-bit data is similar to 8-bit transmission, except that a 16-bit data word (of which the upper 7 bits are always clear) must be written to the UxTXREG register.

18.3.3 TRANSMIT BUFFER (UxTXB)

The transmit buffer is 9 bits wide and 4 characters deep. Including the Transmit Shift register (UxTSR), the user effectively has a 5-deep FIFO (First In First Out) buffer. The UTXBF Status bit (UxSTA<9>) indicates whether the transmit buffer is full.

If a user attempts to write to a full buffer, the new data will not be accepted into the FIFO, and no data shift will occur within the buffer. This enables recovery from a buffer overrun condition.

The FIFO is reset during any device Reset, but is not affected when the device enters or wakes up from a power-saving mode.

dsPIC30F4011/4012

18.3.4 TRANSMIT INTERRUPT

The Transmit Interrupt Flag (U1TXIF or U2TXIF) is located in the corresponding interrupt flag register.

The transmitter generates an edge to set the UxTXIF bit. The condition for generating the interrupt depends on UTXISEL control bit:

- a) If UTXISEL = 0, an interrupt is generated when a word is transferred from the transmit buffer to the Transmit Shift register (UxTSR). This implies that the transmit buffer has at least one empty word.
- b) If UTXISEL = 1, an interrupt is generated when a word is transferred from the transmit buffer to the Transmit Shift register (UxTSR) and the transmit buffer is empty.

Switching between the two interrupt modes during operation is possible and sometimes offers more flexibility.

18.3.5 TRANSMIT BREAK

Setting the UTXBRK bit (UxSTA<11>) will cause the UxTX line to be driven to logic '0'. The UTXBRK bit overrides all transmission activity. Therefore, the user should generally wait for the transmitter to be Idle before setting UTXBRK.

To send a Break character, the UTXBRK bit must be set by software and must remain set for a minimum of 13 baud clock cycles. The UTXBRK bit is then cleared by software to generate Stop bits. The user must wait for a duration of at least one or two baud clock cycles in order to ensure a valid Stop bit(s) before reloading the UxTXB or starting other transmitter activity. Transmission of a Break character does not generate a transmit interrupt.

18.4 Receiving Data

18.4.1 RECEIVING IN 8-BIT OR 9-BIT DATA MODE

The following steps must be performed while receiving 8-bit or 9-bit data:

1. Set up the UART (see **Section 18.3.1 "Transmitting in 8-bit Data Mode"**).
2. Enable the UART (see **Section 18.3.1 "Transmitting in 8-bit Data Mode"**).
3. A receive interrupt will be generated when one or more data words have been received, depending on the receive interrupt settings specified by the URXISEL bits (UxSTA<7:6>).
4. Read the OERR bit to determine if an overrun error has occurred. The OERR bit must be reset in software.
5. Read the received data from UxRXREG. The act of reading UxRXREG will move the next word to the top of the receive FIFO and the PERR and FERR values will be updated.

18.4.2 RECEIVE BUFFER (UxRXB)

The receive buffer is 4 words deep. Including the Receive Shift register (UxRSR), the user effectively has a 5-word deep FIFO buffer.

URXDA (UxSTA<0>) = 1 indicates that the receive buffer has data available. URXDA = 0 implies that the buffer is empty. If a user attempts to read an empty buffer, the old values in the buffer will be read and no data shift will occur within the FIFO.

The FIFO is reset during any device Reset. It is not affected when the device enters or wakes up from a power-saving mode.

18.4.3 RECEIVE INTERRUPT

The Receive Interrupt Flag (U1RXIF or U2RXIF) can be read from the corresponding interrupt flag register. The interrupt flag is set by an edge generated by the receiver. The condition for setting the receive interrupt flag depends on the settings specified by the URXISEL<1:0> (UxSTA<7:6>) control bits.

- a) If URXISEL<1:0> = 00 or 01, an interrupt is generated every time a data word is transferred from the Receive Shift register (UxRSR) to the receive buffer. There may be one or more characters in the receive buffer.
- b) If URXISEL<1:0> = 10, an interrupt is generated when a word is transferred from the Receive Shift register (UxRSR) to the receive buffer which, as a result of the transfer, contains 3 characters.
- c) If URXISEL<1:0> = 11, an interrupt is set when a word is transferred from the Receive Shift register (UxRSR) to the receive buffer which, as a result of the transfer, contains 4 characters (i.e., becomes full).

Switching between the interrupt modes during operation is possible, though generally not advisable during normal operation.

18.5 Reception Error Handling

18.5.1 RECEIVE BUFFER OVERRUN ERROR (OERR BIT)

The OERR bit (UxSTA<1>) is set if all of the following conditions occur:

- a) The receive buffer is full.
- b) The Receive Shift register is full, but unable to transfer the character to the receive buffer.
- c) The Stop bit of the character in the UxRSR is detected, indicating that the UxRSR needs to transfer the character to the buffer.

Once OERR is set, no further data is shifted in UxRSR (until the OERR bit is cleared in software or a Reset occurs). The data held in UxRSR and UxRXREG remains valid.

18.5.2 FRAMING ERROR (FERR)

The FERR bit (UxSTA<2>) is set if a '0' is detected instead of a Stop bit. If two Stop bits are selected, both Stop bits must be '1', otherwise FERR will be set. The read-only FERR bit is buffered along with the received data; it is cleared on any Reset.

18.5.3 PARITY ERROR (PERR)

The PERR bit (UxSTA<3>) is set if the parity of the received word is incorrect. This error bit is applicable only if a Parity mode (odd or even) is selected. The read-only PERR bit is buffered along with the received data bytes; it is cleared on any Reset.

18.5.4 IDLE STATUS

When the receiver is active (i.e., between the initial detection of the Start bit and the completion of the Stop bit), the RIDLE bit (UxSTA<4>) is '0'. Between the completion of the Stop bit and detection of the next Start bit, the RIDLE bit is '1', indicating that the UART is Idle.

18.5.5 RECEIVE BREAK

The receiver will count and expect a certain number of bit times based on the values programmed in the PDSEL<1:0> (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.

If the Break is longer than 13 bit times, the reception is considered complete after the number of bit times specified by PDSEL and STSEL. The URXDA bit is set, FERR is set, zeros are loaded into the receive FIFO, interrupts are generated, if appropriate and the RIDLE bit is set.

When the module receives a long Break signal and the receiver has detected the Start bit, the data bits and the invalid Stop bit (which sets the FERR), the receiver must wait for a valid Stop bit before looking for the next Start bit. It cannot assume that the Break condition on the line is the next Start bit.

Break is regarded as a character containing all '0's, with the FERR bit set. The Break character is loaded into the buffer. No further reception can occur until a Stop bit is received. Note that RIDLE goes high when the Stop bit has not been received yet.

18.6 Address Detect Mode

Setting the ADDEN bit (UxSTA<5>) enables this special mode in which a 9th bit (URX8) value of '1' identifies the received word as an address, rather than data. This mode is only applicable for 9-bit data communication. The URXISELx control bit does not have any impact on interrupt generation in this mode, since an interrupt (if enabled) will be generated every time the received word has the 9th bit set.

18.7 Loopback Mode

Setting the LPBACK bit enables this special mode in which the UxTX pin is internally connected to the UxRX pin. When configured for the Loopback mode, the UxRX pin is disconnected from the internal UART receive logic. However, the UxTX pin still functions as in a normal operation.

To select this mode:

- Configure UART for desired mode of operation.
- Set LPBACK = 1 to enable Loopback mode.
- Enable transmission as defined in **Section 18.3 "Transmitting Data"**.

18.8 Baud Rate Generator

The UART has a 16-bit Baud Rate Generator to allow maximum flexibility in baud rate generation. The Baud Rate Generator register (UxBRG) is readable and writable. The baud rate is computed as follows:

BRG = 16-bit value held in UxBRG register
(0 through 65535)

FCY = Instruction Clock Rate (1/TCY)

The baud rate is given by Equation 18-1.

EQUATION 18-1: BAUD RATE

$$\text{Baud Rate} = \text{FCY} / (16 * (\text{BRG} + 1))$$

Therefore, maximum baud rate possible is:

$\text{FCY} / 16$ (if BRG = 0),

and the minimum baud rate possible is:

$\text{FCY} / (16 * 65536)$.

With a full, 16-bit Baud Rate Generator, at 30 MIPS operation, the minimum baud rate achievable is 28.5 bps.

dsPIC30F4011/4012

18.9 Auto-Baud Support

To allow the system to determine baud rates of received characters, the input can be optionally linked to a selected capture input (IC1 for UART1, IC2 for UART2). To enable this mode, the user must program the input capture module to detect the falling and rising edges of the Start bit.

18.10 UART Operation During CPU Sleep and Idle Modes

18.10.1 UART OPERATION DURING CPU SLEEP MODE

When the device enters Sleep mode, all clock sources to the module are shut down and stay at logic '0'. If entry into Sleep mode occurs while a transmission is in progress, then the transmission is aborted. The UxTX pin is driven to logic '1'. Similarly, if entry into Sleep mode occurs while a reception is in progress, then the reception is aborted. The UxSTA, UxMODE, UxBRG, transmit and receive registers and buffers, are not affected by Sleep mode.

If the WAKE bit (UxMODE<7>) is set before the device enters Sleep mode, then a falling edge on the UxRX pin will generate a receive interrupt. The Receive Interrupt Select Mode bit (URXISEL) has no effect for this function. If the receive interrupt is enabled, then this will wake-up the device from Sleep. The UARTEN bit must be set in order to generate a wake-up interrupt.

18.10.2 UART OPERATION DURING CPU IDLE MODE

For the UART, the USIDL bit selects if the module will stop operation when the device enters Idle mode, or whether the module will continue on Idle. If USIDL = 0, the module will continue operation during Idle mode. If USIDL = 1, the module will stop on Idle.

TABLE 18-1: UART1 REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|-------------------------------|--------|--------|--------|--------|--------|-------|-------|-------------------|----------|-------|-------|-------|--------|--------|-------|---------------------|
| U1MODE | 020C | UARTEN | — | USIDL | — | — | ALTI0 | — | — | WAKE | LPBACK | ABAUD | — | — | PDSEL1 | PDSEL0 | STSEL | 0000 0000 0000 0000 |
| U1STA | 020E | UTXISEL | — | — | — | UTXBRK | UTXEN | UTXBF | TRMT | URXISEL1 | URXISEL0 | ADDEN | RIDL | PERR | FERR | OERR | URXDA | 0000 0001 0001 0000 |
| U1TXREG | 0210 | — | — | — | — | — | — | — | UTX8 | Transmit Register | | | | | | | | 0000 000u uuuu uuuu |
| U1RXREG | 0212 | — | — | — | — | — | — | — | URX8 | Receive Register | | | | | | | | 0000 0000 0000 0000 |
| U1BRG | 0214 | Baud Rate Generator Prescaler | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

TABLE 18-2: UART2 REGISTER MAP (NOT AVAILABLE ON dsPIC30F4012)

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|-------------------------------|--------|--------|--------|--------|--------|-------|-------|-------------------|----------|-------|-------|-------|--------|--------|-------|---------------------|
| U2MODE | 0216 | UARTEN | — | USIDL | — | — | — | — | — | WAKE | LPBACK | ABAUD | — | — | PDSEL1 | PDSEL0 | STSEL | 0000 0000 0000 0000 |
| U2STA | 0218 | UTXISEL | — | — | — | UTXBRK | UTXEN | UTXBF | TRMT | URXISEL1 | URXISEL0 | ADDEN | RIDL | PERR | FERR | OERR | URXDA | 0000 0001 0001 0000 |
| U2TXREG | 021A | — | — | — | — | — | — | — | UTX8 | Transmit Register | | | | | | | | 0000 000u uuuu uuuu |
| U2RXREG | 021C | — | — | — | — | — | — | — | URX8 | Receive Register | | | | | | | | 0000 0000 0000 0000 |
| U2BRG | 021E | Baud Rate Generator Prescaler | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

19.0 CAN MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

19.1 Overview

The Controller Area Network (CAN) module is a serial interface, useful for communicating with other CAN modules or digital signal controller devices. This interface/protocol was designed to allow communications within noisy environments. The dsPIC30F4011/4012 devices have 1 CAN module.

The CAN module is a communication controller implementing the CAN 2.0 A/B protocol, as defined in the BOSCH specification. The module will support CAN 1.2, CAN 2.0A, CAN2.0B Passive and CAN 2.0B Active versions of the protocol. The module implementation is a full CAN system. The CAN specification is not covered within this data sheet. The reader may refer to the BOSCH CAN specification for further details.

The module features are as follows:

- Implementation of the CAN protocol CAN 1.2, CAN 2.0A and CAN 2.0B
- Standard and extended data frames
- 0-8 bytes data length
- Programmable bit rate up to 1 Mbit/sec
- Support for remote frames
- Double-buffered receiver with two prioritized received message storage buffers (each buffer may contain up to 8 bytes of data)
- 6 full (standard/extended identifier), acceptance filters, 2 associated with the high priority receive buffer and 4 associated with the low priority receive buffer
- 2 full, acceptance filter masks, one each associated with the high and low priority receive buffers
- Three transmit buffers with application specified prioritization and abort capability (each buffer may contain up to 8 bytes of data)
- Programmable wake-up functionality with integrated low-pass filter
- Programmable Loopback mode supports self-test operation
- Signaling via interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to input capture module (IC2, for both CAN1 and CAN2) for time-stamping and network synchronization
- Low-power Sleep and Idle mode

The CAN bus module consists of a protocol engine and message buffering/control. The CAN protocol engine handles all functions for receiving and transmitting messages on the CAN bus. Messages are transmitted by first loading the appropriate data registers. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against filters to see if it should be received and stored in one of the receive registers.

19.2 Frame Types

The CAN module transmits various types of frames which include data messages or remote transmission requests, initiated by the user, as other frames that are automatically generated for control purposes. The following frame types are supported:

19.2.1 STANDARD DATA FRAME

A standard data frame is generated by a node when the node wishes to transmit data. It includes an 11-bit Standard Identifier (SID) but not an 18-bit Extended Identifier (EID).

19.2.2 EXTENDED DATA FRAME

An extended data frame is similar to a standard data frame but includes an extended identifier as well.

19.2.3 REMOTE FRAME

It is possible for a destination node to request the data from the source. For this purpose, the destination node sends a remote frame with an identifier that matches the identifier of the required data frame. The appropriate data source node will then send a data frame as a response to this remote request.

19.2.4 ERROR FRAME

An error frame is generated by any node that detects a bus error. An error frame consists of 2 fields: an error flag field and an error delimiter field.

19.2.5 OVERLOAD FRAME

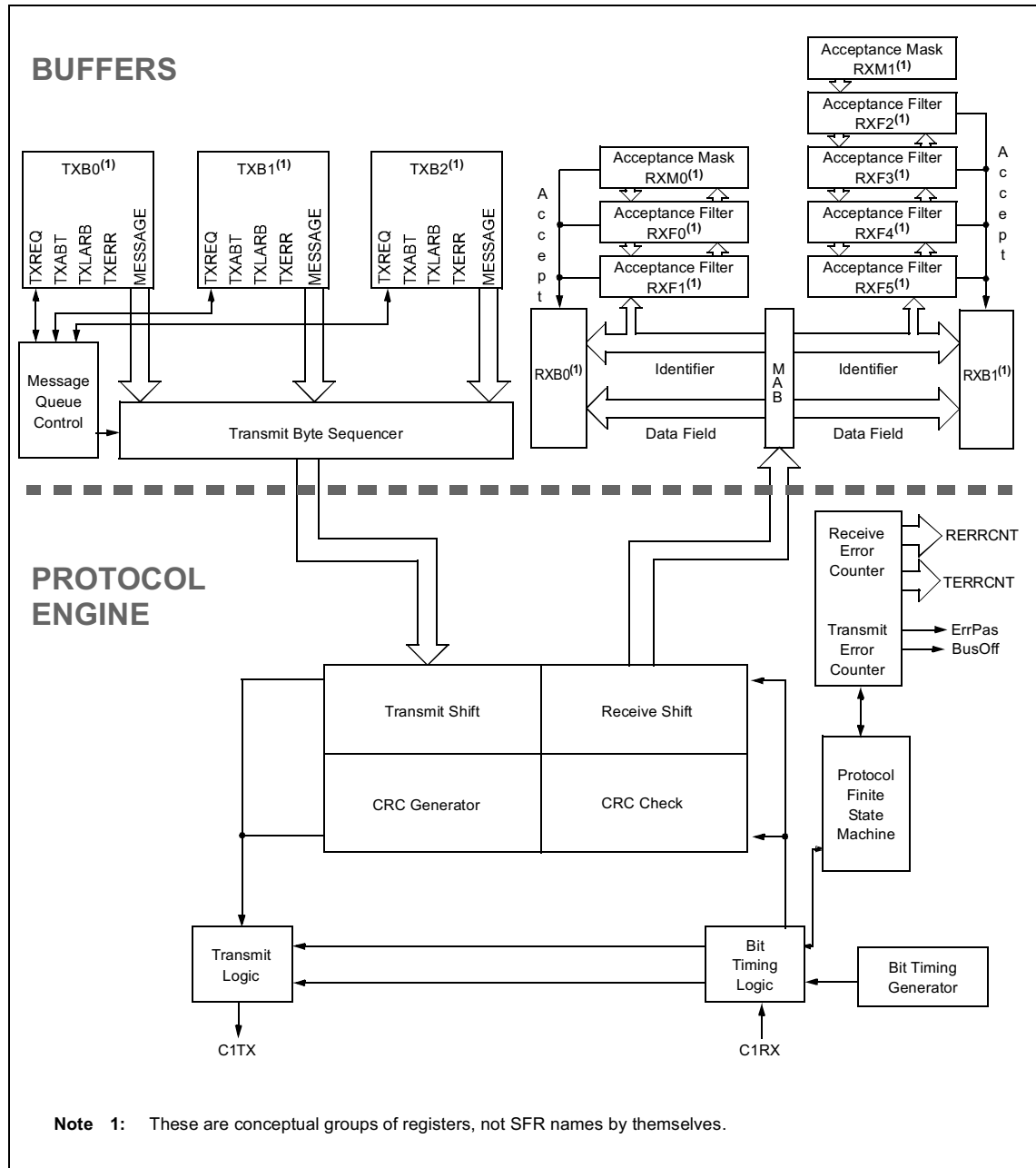
An overload frame can be generated by a node as a result of 2 conditions. First, the node detects a dominant bit during interframe space which is an illegal condition. Second, due to internal conditions, the node is not yet able to start reception of the next message. A node may generate a maximum of 2 sequential overload frames to delay the start of the next message.

19.2.6 INTERFRAME SPACE

Interframe space separates a proceeding frame (of whatever type) from a following data or remote frame.

dsPIC30F4011/4012

FIGURE 19-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM



19.3 Modes of Operation

The CAN module can operate in one of several operation modes selected by the user. These modes include:

- Initialization Mode
- Disable Mode
- Normal Operation Mode
- Listen Only Mode
- Loopback Mode
- Error Recognition Mode

Modes are requested by setting the REQOP<2:0> bits (C1CTRL<10:8>). Entry into a mode is acknowledged by monitoring the OPMODE<2:0> bits (C1CTRL<7:5>). The module will not change the mode and the OPMODE bits until a change in mode is acceptable, generally during bus Idle time which is defined as at least 11 consecutive recessive bits.

19.3.1 INITIALIZATION MODE

In the Initialization mode, the module will not transmit or receive. The error counters are cleared and the interrupt flags remain unchanged. The programmer will have access to Configuration registers that are access restricted in other modes. The module will protect the user from accidentally violating the CAN protocol through programming errors. All registers which control the configuration of the module can not be modified while the module is online. The CAN module will not be allowed to enter the Configuration mode while a transmission is taking place. The Configuration mode serves as a lock to protect the following registers.

- All Module Control Registers
- Baud Rate and Interrupt Configuration Registers
- Bus Timing Registers
- Identifier Acceptance Filter Registers
- Identifier Acceptance Mask Registers

19.3.2 DISABLE MODE

In Disable mode, the module will not transmit or receive. The module has the ability to set the WAKIF bit due to bus activity, however, any pending interrupts will remain and the error counters will retain their value.

If the REQOP<2:0> bits (C1CTRL<10:8>) = 001, the module will enter the Disable mode. If the module is active, the module will wait for 11 recessive bits on the CAN bus, detect that condition as an Idle bus, then accept the disable command. When the OPMODE<2:0> bits (C1CTRL<7:5>) = 001, that indicates whether the module successfully went into Disable mode. The I/O pins will revert to normal I/O function when the module is in the Disable mode.

The module can be programmed to apply a low-pass filter function to the C1RX input line while the module or the CPU is in Sleep mode. The WAKFIL bit (C1CFG2<14>) enables or disables the filter.

Note: Typically, if the CAN module is allowed to transmit in a particular mode of operation and a transmission is requested immediately after the CAN module has been placed in that mode of operation, the module waits for 11 consecutive recessive bits on the bus before starting transmission. If the user switches to Disable mode within this 11-bit period, then this transmission is aborted and the corresponding TXABT bit is set and TXREQ bit is cleared.

19.3.3 NORMAL OPERATION MODE

Normal Operation mode is selected when REQOP<2:0> = 000. In this mode, the module is activated and the I/O pins will assume the CAN bus functions. The module will transmit and receive CAN bus messages via the C1TX and C1RX pins.

19.3.4 LISTEN ONLY MODE

If the Listen Only mode is activated, the module on the CAN bus is passive. The transmitter buffers revert to the port I/O function. The receive pins remain inputs. For the receiver, no error flags or Acknowledge signals are sent. The error counters are deactivated in this state. The Listen Only mode can be used for detecting the baud rate on the CAN bus. To use this, it is necessary that there are at least two further nodes that communicate with each other.

19.3.5 ERROR RECOGNITION MODE

The module can be set to ignore all errors and receive any message. In this mode, the data which is in the message assembly buffer until the time an error occurred, is copied in the receive buffer and can be read via the CPU interface.

19.3.6 LOOPBACK MODE

If the Loopback mode is activated, the module will connect the internal transmit signal to the internal receive signal at the module boundary. The transmit and receive pins revert to their port I/O function.

dsPIC30F4011/4012

19.4 Message Reception

19.4.1 RECEIVE BUFFERS

The CAN bus module has 3 receive buffers. However, one of the receive buffers is always committed to monitoring the bus for incoming messages. This buffer is called the message assembly buffer (MAB). There are two receive buffers, visibly denoted as RXB0 and RXB1, that can essentially instantaneously receive a complete message from the protocol engine.

All messages are assembled by the MAB, and are transferred to the RXBn buffers only if the acceptance filter criterion is met. When a message is received, the RXxIF flag (C1INTF<0> or C1INIF<1>) will be set. This bit can only be set by the module when a message is received. The bit is cleared by the CPU when it has completed processing the message in the buffer. If the RXxIE bit (C1INTE<0> or C1INTE<1>) is set, an interrupt will be generated when a message is received.

RXF0 and RXF1 filters with the RXM0 mask are associated with RXB0. The filters, RXF2, RXF3, RXF4 and RXF5, and the mask, RXM1, are associated with RXB1.

19.4.2 MESSAGE ACCEPTANCE FILTERS

The message acceptance filters and masks are used to determine if a message in the message assembly buffer should be loaded into either of the receive buffers. Once a valid message has been received into the Message Assembly Buffer (MAB), the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer.

The acceptance filter looks at incoming messages for the RXIDE bit (CiRXnSID<0>) to determine how to compare the identifiers. If the RXIDE bit is clear, the message is a standard frame and only filters with the EXIDE bit (C1RXFxSID<0>) clear are compared. If the RXIDE bit is set, the message is an extended frame and only filters with the EXIDE bit set are compared.

19.4.3 MESSAGE ACCEPTANCE FILTER MASKS

The mask bits essentially determine which bits to apply the filter to. If any mask bit is set to a zero, then that bit will automatically be accepted regardless of the filter bit. There are 2 programmable acceptance filter masks associated with the receive buffers, one for each buffer.

19.4.4 RECEIVE OVERRUN

An overrun condition occurs when the Message Assembly Buffer (MAB) has assembled a valid received message, the message is accepted through the acceptance filters, and when the receive buffer associated with the filter has not been designated as clear of the previous message.

The overrun error flag, RXxOVR (C1INTF<15> or C1INTF<14>) and the ERRIF bit (C1INTF<5>) will be set and the message in the MAB will be discarded.

If the DBEN bit is clear, RXB1 and RXB0 operate independently. When this is the case, a message intended for RXB0 will not be diverted into RXB1 if RXB0 contains an unread message and the RXOVR bit will be set.

If the DBEN bit is set, the overrun for RXB0 is handled differently. If a valid message is received for RXB0 and RXFUL = 1, it indicates that RXB0 is full and RXFUL = 0 indicates that RXB1 is empty, the message for RXB0 will be loaded into RXB1. An overrun error will not be generated for RXB0. If a valid message is received for RXB0 and RXFUL = 1, and RXFUL = 1 indicates that both RXB0 and RXB1 are full, the message will be lost and an overrun will be indicated for RXB1.

19.4.5 RECEIVE ERRORS

The CAN module will detect the following receive errors:

- Cyclic Redundancy Check (CRC) Error
- Bit Stuffing Error
- Invalid Message Receive Error

These receive errors do not generate an interrupt. However, the receive error counter is incremented by one in case one of these errors occur. The RXWAR bit (C1INTF<9>) indicates that the receive error counter has reached the CPU warning limit of 96 and an interrupt is generated.

19.4.6 RECEIVE INTERRUPTS

Receive interrupts can be divided into 3 major groups, each including various conditions that generate interrupts:

19.4.6.1 Receive Interrupt

A message has been successfully received and loaded into one of the receive buffers. This interrupt is activated immediately after receiving the End-of-Frame (EOF) field. Reading the RXxIF flag will indicate which receive buffer caused the interrupt.

19.4.6.2 Wake-up Interrupt

The CAN module has woken up from Disable mode or the device has woken up from Sleep mode.

19.4.6.3 Receive Error Interrupts

A receive error interrupt will be indicated by the ERRIF bit. This bit shows that an error condition occurred. The source of the error can be determined by checking the bits in the CAN Interrupt Status register, C1INTF.

- Invalid message received.
- If any type of error occurred during reception of the last message, an error will be indicated by the IVRIF bit.
- Receiver overrun.
- The RXxOVR bit indicates that an overrun condition occurred.
- Receiver warning.
- The RXWAR bit indicates that the Receive Error Counter (RERRCNT<7:0>) has reached the warning limit of 96.
- Receiver error passive.
- The RXEP bit indicates that the Receive Error Counter has exceeded the error passive limit of 127 and the module has gone into error passive state.

19.5 Message Transmission

19.5.1 TRANSMIT BUFFERS

The CAN module has three transmit buffers. Each of the three buffers occupies 14 bytes of data. Eight of the bytes are the maximum 8 bytes of the transmitted message. Five bytes hold the standard and extended identifiers and other message arbitration information.

19.5.2 TRANSMIT MESSAGE PRIORITY

Transmit priority is a prioritization within each node of the pending transmittable messages. There are 4 levels of transmit priority. If TXPRI<1:0> (C1TXxCON<1:0>, where x = 0, 1 or 2, represents a particular transmit buffer) for a particular message buffer is set to '11', that buffer has the highest priority. If TXPRI<1:0> for a particular message buffer is set to '10' or '01', that buffer has an intermediate priority. If TXPRI<1:0> for a particular message buffer is '00', that buffer has the lowest priority.

19.5.3 TRANSMISSION SEQUENCE

To initiate transmission of the message, the TXREQ bit (C1TXxCON<3>) must be set. The CAN bus module resolves any timing conflicts between setting of the TXREQ bit and the Start-of-Frame (SOF), ensuring that if the priority was changed, it is resolved correctly before the SOF occurs. When TXREQ is set, the TXABT (C1TXxCON<6>), TXLARB (C1TXxCON<5>) and TXERR (C1TXxCON<4>) flag bits are automatically cleared.

Setting TXREQ bit simply flags a message buffer as enqueued for transmission. When the module detects an available bus, it begins transmitting the message which has been determined to have the highest priority.

If the transmission completes successfully on the first attempt, the TXREQ bit is cleared automatically and an interrupt is generated if TXxIE was set.

If the message transmission fails, one of the error condition flags will be set and the TXREQ bit will remain set, indicating that the message is still pending for transmission. If the message encountered an error condition during the transmission attempt, the TXERR bit will be set and the error condition may cause an interrupt. If the message loses arbitration during the transmission attempt, the TXLARB bit is set. No interrupt is generated to signal the loss of arbitration.

19.5.4 ABORTING MESSAGE TRANSMISSION

The system can also abort a message by clearing the TXREQ bit associated with each message buffer. Setting the ABAT bit (C1CTRL<12>) will request an abort of all pending messages. If the message has not yet started transmission, or if the message started but is interrupted by loss of arbitration or an error, the abort will be processed. The abort is indicated when the module sets the TXABT bit, and the TXxIF flag is not automatically set.

19.5.5 TRANSMISSION ERRORS

The CAN module will detect the following transmission errors:

- Acknowledge Error
- Form Error
- Bit Error

These transmission errors will not necessarily generate an interrupt but are indicated by the transmission error counter. However, each of these errors will cause the transmission error counter to be incremented by one. Once the value of the error counter exceeds the value of 96, the ERRIF (C1INTF<5>) and the TXWAR bit (C1INTF<10>) are set. Once the value of the error counter exceeds the value of 96, an interrupt is generated and the TXWAR bit in the error flag register is set.

dsPIC30F4011/4012

19.5.6 TRANSMIT INTERRUPTS

Transmit interrupts can be divided into 2 major groups, each including various conditions that generate interrupts:

- Transmit Interrupt

At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. Reading the TXxIF flags will indicate which transmit buffer is available and caused the interrupt.

- Transmit Error Interrupts

A transmission error interrupt will be indicated by the ERRIF flag. This flag shows that an error condition occurred. The source of the error can be determined by checking the error flags in the CAN Interrupt Status register, C1INTF. The flags in this register are related to receive and transmit errors.

- Transmitter warning interrupt.
- The TXWAR bit indicates that the Transmit Error Counter has reached the CPU warning limit of 96.
- Transmitter error passive.
- The TXEP bit (C1INTF<12>) indicates that the Transmit Error Counter has exceeded the error passive limit of 127 and the module has gone to error passive state.
- Bus off.
- The TXBO bit (C1INTF<13>) indicates that the Transmit Error Counter (TERRCNT<7:0>) has exceeded 255 and the module has gone to bus off state.

19.6 Baud Rate Setting

All nodes on any particular CAN bus must have the same nominal bit rate. In order to set the baud rate, the following parameters have to be initialized:

- Synchronization Jump Width
- Baud Rate Prescaler
- Phase Segments
- Length Determination of Phase2 Seg
- Sample Point
- Propagation Segment Bits

19.6.1 BIT TIMING

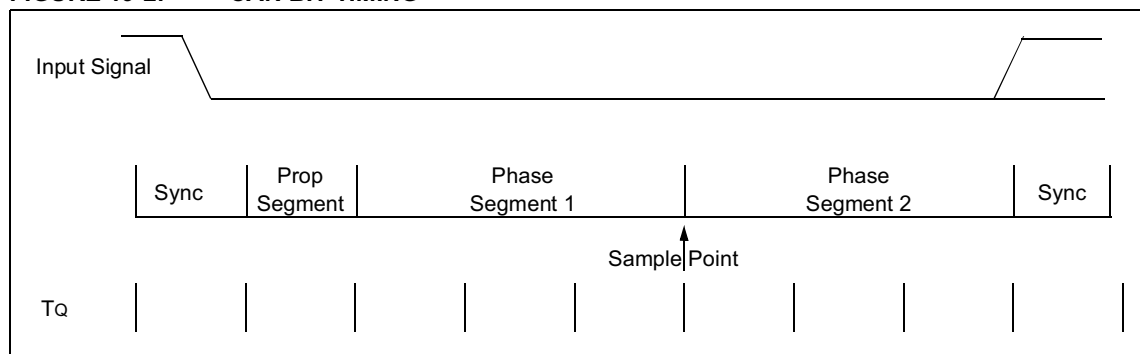
All controllers on the CAN bus must have the same baud rate and bit length. However, different controllers are not required to have the same master oscillator clock. At different clock frequencies of the individual controllers, the baud rate has to be adjusted by adjusting the number of time quanta in each segment.

The nominal bit time can be thought of as being divided into separate non-overlapping time segments. These segments are shown in Figure 19-2.

- Synchronization segment (Sync Seg)
- Propagation time segment (Prop Seg)
- Phase segment 1 (Phase1 Seg)
- Phase segment 2 (Phase2 Seg)

The time segments and also the nominal bit time are made up of integer units of time called time quanta or T_Q. By definition, the nominal bit time has a minimum of 8 T_Q and a maximum of 25 T_Q. Also, by definition, the minimum nominal bit time is 1 μsec, corresponding to a maximum bit rate of 1 MHz.

FIGURE 19-2: CAN BIT TIMING



19.6.2 PRESCALER SETTING

There is a programmable prescaler with integral values ranging from 1 to 64 in addition to a fixed divide-by-2 for clock generation. The Time Quantum (T_Q) is a fixed unit of time derived from the oscillator period, shown in Equation 19-1, where F_{CAN} is F_{CY} (if the CANCKS bit is set) or 4 F_{CY} (if CANCKS is cleared).

Note: F_{CAN} must not exceed 30 MHz. If CANCKS = 0, then F_{CY} must not exceed 7.5 MHz.

EQUATION 19-1: TIME QUANTUM FOR CLOCK GENERATION

$$T_Q = 2 (BRP<5:0> + 1) / F_{CAN}$$

19.6.3 PROPAGATION SEGMENT

This part of the bit time is used to compensate physical delay times within the network. These delay times consist of the signal propagation time on the bus line and the internal delay time of the nodes. The propagation segment can be programmed from 1 T_Q to 8 T_Q by setting the PRSEG<2:0> bits (C1CFG2<2:0>).

19.6.4 PHASE SEGMENTS

The phase segments are used to optimally locate the sampling of the received bit within the transmitted bit time. The sampling point is between Phase1 Seg and Phase2 Seg. These segments are lengthened or shortened by resynchronization. The end of the Phase1 Seg determines the sampling point within a bit period. The segment is programmable from 1 T_Q to 8 T_Q. Phase2 Seg provides delay to the next transmitted data transition. The segment is programmable from 1 T_Q to 8 T_Q, or it may be defined to be equal to the greater of Phase1 Seg or the information processing time (2 T_Q). The Phase1 Seg is initialized by setting bits SEG1PH<2:0> (C1CFG2<5:3>), and Phase2 Seg is initialized by setting SEG2PH<2:0> (C1CFG2<10:8>).

The following requirement must be fulfilled while setting the lengths of the phase segments:

$$\text{Propagation Segment} + \text{Phase1 Seg} > = \text{Phase2 Seg}$$

19.6.5 SAMPLE POINT

The sample point is the point of time at which the bus level is read and interpreted as the value of that respective bit. The location is at the end of Phase1 Seg. If the bit timing is slow and contains many T_Q, it is possible to specify multiple sampling of the bus line at the sample point. The level determined by the CAN bus then corresponds to the result from the majority decision of three values. The majority samples are taken at the sample point and twice before with a distance of T_Q/2. The CAN module allows the user to choose between sampling three times at the same point, or once at the same point, by setting or clearing the SAM bit (C1CFG2<6>).

Typically, the sampling of the bit should take place at about 60-70% through the bit time depending on the system parameters.

19.6.6 SYNCHRONIZATION

To compensate for phase shifts between the oscillator frequencies of the different bus stations, each CAN controller must be able to synchronize to the relevant signal edge of the incoming signal. When an edge in the transmitted data is detected, the logic will compare the location of the edge to the expected time (synchronous segment). The circuit will then adjust the values of Phase1 Seg and Phase2 Seg. There are 2 mechanisms used to synchronize.

19.6.6.1 Hard Synchronization

Hard synchronization is only done whenever there is a 'recessive' to 'dominant' edge during bus Idle, indicating the start of a message. After hard synchronization, the bit time counters are restarted with the synchronous segment. Hard synchronization forces the edge which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time. If a hard synchronization is done, there will not be a resynchronization within that bit time.

19.6.6.2 Resynchronization

As a result of resynchronization, Phase1 Seg may be lengthened or Phase2 Seg may be shortened. The amount of lengthening or shortening of the phase buffer segment has an upper bound known as the synchronization jump width, and is specified by the SJW<1:0> bits (C1CFG1<7:6>). The value of the synchronization jump width will be added to Phase1 Seg or subtracted from Phase2 Seg. The resynchronization jump width is programmable between 1 T_Q and 4 T_Q.

The following requirement must be fulfilled while setting the SJW<1:0> bits:

$$\text{Phase2 Seg} > \text{Synchronization Jump Width}$$

dsPIC30F4011/4012

TABLE 19-1: CAN1 REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|------------|-------|--------|--------|--------|--|--------|---|-------|-------|-------|-------|--------|-------|-------|-------|-------|-------|---------------------|
| C1RXF0SID | 0300 | — | — | — | — | — | Receive Acceptance Filter 0 Standard Identifier<10:0> | — | — | — | — | — | — | — | — | — | EXIDE | 0000 uuuu uuuu uuuu |
| C1RXF0EIDL | 0302 | — | — | — | — | — | Receive Acceptance Filter 0 Extended Identifier<17:6> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF0EIDL | 0304 | — | — | — | Receive Acceptance Filter 0 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uu00 0000 0000 |
| C1RXF1SID | 0308 | — | — | — | — | — | Receive Acceptance Filter 1 Standard Identifier<10:0> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF1EIDL | 030A | — | — | — | — | — | Receive Acceptance Filter 1 Extended Identifier<17:6> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF1EIDL | 030C | — | — | — | Receive Acceptance Filter 1 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uu00 0000 0000 |
| C1RXF2SID | 0310 | — | — | — | — | — | Receive Acceptance Filter 2 Standard Identifier<10:0> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF2EIDL | 0312 | — | — | — | — | — | Receive Acceptance Filter 2 Extended Identifier<17:6> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF2EIDL | 0314 | — | — | — | Receive Acceptance Filter 2 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uu00 0000 0000 |
| C1RXF3SID | 0318 | — | — | — | — | — | Receive Acceptance Filter 3 Standard Identifier<10:0> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF3EIDL | 031A | — | — | — | — | — | Receive Acceptance Filter 3 Extended Identifier<17:6> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF3EIDL | 031C | — | — | — | Receive Acceptance Filter 3 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uu00 0000 0000 |
| C1RXF4SID | 0320 | — | — | — | — | — | Receive Acceptance Filter 4 Standard Identifier<10:0> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF4EIDL | 0322 | — | — | — | — | — | Receive Acceptance Filter 4 Extended Identifier<17:6> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF4EIDL | 0324 | — | — | — | Receive Acceptance Filter 4 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uu00 0000 0000 |
| C1RXF5SID | 0328 | — | — | — | — | — | Receive Acceptance Filter 5 Standard Identifier<10:0> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF5EIDL | 032A | — | — | — | — | — | Receive Acceptance Filter 5 Extended Identifier<17:6> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXF5EIDL | 032C | — | — | — | Receive Acceptance Filter 5 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uu00 0000 0000 |
| C1RXM0SID | 0330 | — | — | — | — | — | Receive Acceptance Mask 0 Standard Identifier<10:0> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXM0EIDL | 0332 | — | — | — | — | — | Receive Acceptance Mask 0 Extended Identifier<17:6> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXM0EIDL | 0334 | — | — | — | Receive Acceptance Mask 0 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uu00 0000 0000 |
| C1RXM1SID | 0338 | — | — | — | — | — | Receive Acceptance Mask 1 Standard Identifier<10:0> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXM1EIDL | 033A | — | — | — | — | — | Receive Acceptance Mask 1 Extended Identifier<17:6> | — | — | — | — | — | — | — | — | — | — | 0000 uuuu uuuu uuuu |
| C1RXM1EIDL | 033C | — | — | — | Receive Acceptance Mask 1 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uu00 0000 0000 |
| C1TX2SID | 0340 | — | — | — | Transmit Buffer 2 Standard Identifier<10:6> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uu00 0000 0000 |
| C1TX2EIDL | 0342 | — | — | — | Transmit Buffer 2 Extended Identifier<17:14> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu 0000 uuuu uuuu |
| C1TX2DLCL | 0344 | — | — | — | Transmit Buffer 2 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uuuu uuuu u000 |
| C1TX2B1 | 0346 | — | — | — | Transmit Buffer 2 Byte 1 | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uuuu uuuu uuuu |
| C1TX2B2 | 0348 | — | — | — | Transmit Buffer 2 Byte 3 | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uuuu uuuu uuuu |
| C1TX2B3 | 034A | — | — | — | Transmit Buffer 2 Byte 5 | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uuuu uuuu uuuu |
| C1TX2B4 | 034C | — | — | — | Transmit Buffer 2 Byte 7 | — | — | — | — | — | — | — | — | — | — | — | — | uuuu uuuu uuuu uuuu |
| C1TX2CON | 034E | — | — | — | — | — | — | — | — | — | TXABT | TXLARB | TXERR | TXREQ | — | — | — | 0000 0000 0000 0000 |
| C1TX1SID | 0350 | — | — | — | Transmit Buffer 1 Standard Identifier<10:6> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu u000 uuuu uuuu |
| C1TX1EIDL | 0352 | — | — | — | Transmit Buffer 1 Extended Identifier<17:14> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu 0000 uuuu uuuu |
| C1TX1DLCL | 0354 | — | — | — | Transmit Buffer 1 Extended Identifier<5:0> | — | — | — | — | — | — | — | — | — | — | — | — | uuuu 0000 uuuu uuuu |

Legend: u = uninitialized bit
Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

TABLE 19-1: CAN1 REGISTER MAP (CONTINUED)

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|--------------|--------|--------|--|--------|-------------|-------|-------|-------|---|----------|-------------|----------|-------------|---------------------|---------------------|---------------------|
| C1TX1B1 | 0356 | | | | Transmit Buffer 1 Byte 1 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1TX1B2 | 0358 | | | | Transmit Buffer 1 Byte 3 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1TX1B3 | 035A | | | | Transmit Buffer 1 Byte 5 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1TX1B4 | 035C | | | | Transmit Buffer 1 Byte 7 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1TX1CON | 035E | — | — | — | — | — | — | — | — | — | TXABT TXLARB TXERR | TXREQ | — | — | — | TXPRI<1:0> | 0000 0000 0000 0000 | |
| C1TX0SID | 0360 | | | | Transmit Buffer 0 Standard Identifier<10:6> | | | | | | | | | | | | | uuuu u000 uuuu uuuu |
| C1TX0EID | 0362 | | | | Transmit Buffer 0 Extended Identifier<17:14> | | | | | | | | | | | | | uuuu 0000 uuuu uuuu |
| C1TX0DLC | 0364 | | | | | | | TXRTR | TXRB1 | TXRB0 | Transmit Buffer 0 Extended Identifier<13:6> | | | | | | | uuuu 0000 uuuu uuuu |
| C1TX0B1 | 0366 | | | | Transmit Buffer 0 Byte 1 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1TX0B2 | 0368 | | | | Transmit Buffer 0 Byte 3 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1TX0B3 | 036A | | | | Transmit Buffer 0 Byte 5 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1TX0B4 | 036C | | | | Transmit Buffer 0 Byte 7 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1TX0CON | 036E | — | — | — | — | — | — | — | — | — | TXABT TXLARB TXERR | TXREQ | — | — | — | TXPRI<1:0> | 0000 0000 0000 0000 | |
| C1RX1SID | 0370 | — | — | — | Receive Buffer 1 Standard Identifier<10:0> | | | | | | | | | | | | | 000u uuuu uuuu uuuu |
| C1RX1EID | 0372 | — | — | — | Receive Buffer 1 Extended Identifier<17:6> | | | | | | | | | | | | | 0000 uuuu uuuu uuuu |
| C1RX1DLC | 0374 | | | | | | | RXRTR | RXRB1 | — | — | — | RXRB0 | DLC<3:0> | | | uuuu uuuu 000u uuuu | |
| C1RX1B1 | 0376 | | | | Receive Buffer 1 Byte 1 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1RX1B2 | 0378 | | | | Receive Buffer 1 Byte 3 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1RX1B3 | 037A | | | | Receive Buffer 1 Byte 5 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1RX1B4 | 037C | | | | Receive Buffer 1 Byte 7 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1RX1CON | 037E | — | — | — | — | — | — | — | — | — | RXFUL | — | — | RXRTRRO | FILHIT<2:0> | 0000 0000 0000 0000 | | |
| C1RX0SID | 0380 | — | — | — | Receive Buffer 0 Standard Identifier<10:0> | | | | | | | | | | | | | 000u uuuu uuuu uuuu |
| C1RX0EID | 0382 | — | — | — | Receive Buffer 0 Extended Identifier<17:6> | | | | | | | | | | | | | 0000 uuuu uuuu uuuu |
| C1RX0DLC | 0384 | | | | | | | RXRTR | RXRB1 | — | — | — | RXRB0 | DLC<3:0> | | | uuuu uuuu 000u uuuu | |
| C1RX0B1 | 0386 | | | | Receive Buffer 0 Byte 1 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1RX0B2 | 0388 | | | | Receive Buffer 0 Byte 3 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1RX0B3 | 038A | | | | Receive Buffer 0 Byte 5 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1RX0B4 | 038C | | | | Receive Buffer 0 Byte 7 | | | | | | | | | | | | | uuuu uuuu uuuu uuuu |
| C1RX0CON | 038E | — | — | — | — | — | — | — | — | — | RXFUL | — | — | RXRTRRO | DBEN | JTOFF | FILHIT0 | 0000 0000 0000 0000 |
| C1CTRL | 0390 | CANCAP | — | CSIDL | ABAT | CANCKS | REQOP<2:0> | — | — | — | OPMODE<2:0> | — | — | — | — | — | — | 0000 0100 1000 0000 |
| C1CFG1 | 0392 | — | — | — | — | — | — | — | — | — | SJW<1:0> | BRP<5:0> | | | | | | 0000 0000 0000 0000 |
| C1CFG2 | 0394 | — | WAKFIL | — | — | — | SEG2PH<2:0> | — | — | — | SEG2PHTS | SAM | SEG1PH<2:0> | | | | | 0000 0000 uuuu uuuu |
| C1INTF | 0396 | RX0OVR | RX1OVR | TXBO | TXEP | RXEP | TXWAR | RXWAR | EWARN | IVRIF | WAKIF | ERRIF | TX2IF | TX1IF | TX0IF | RX1IF | RX0IF | 0000 0000 0000 0000 |
| C1INTE | 0398 | — | — | — | — | — | — | — | — | — | IVRIE | WAKIE | TX2IE | TX1IE | TX0IE | RX1IE | RX0IE | 0000 0000 0000 0000 |
| C1IEC | 039A | TERRCNT<7:0> | | | | | | | | | | | | | | | | 0000 0000 0000 0000 |

Legend: u = uninitialized bit
 Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

20.0 10-BIT, HIGH-SPEED ANALOG-TO-DIGITAL CONVERTER (ADC) MODULE

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

The 10-bit, high-speed Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit digital number. This module is based on a Successive Approximation Register (SAR) architecture and provides a maximum sampling rate of 1 Msps. The ADC module has 16 analog inputs which are multiplexed into four sample and hold amplifiers. The output of the sample and hold is the input into the converter which generates the result. The analog reference voltages are software selectable to either the device supply voltage (AVDD/AVSS) or the voltage level on the (VREF+/VREF-) pins. The ADC module has a unique feature of being able to operate while the device is in Sleep mode.

The ADC module has six, 16-bit registers:

- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)
- A/D Control Register 3 (ADCON3)
- A/D Input Select Register (ADCHS)
- A/D Port Configuration Register (ADPCFG)
- A/D Input Scan Selection Register (ADCSSL)

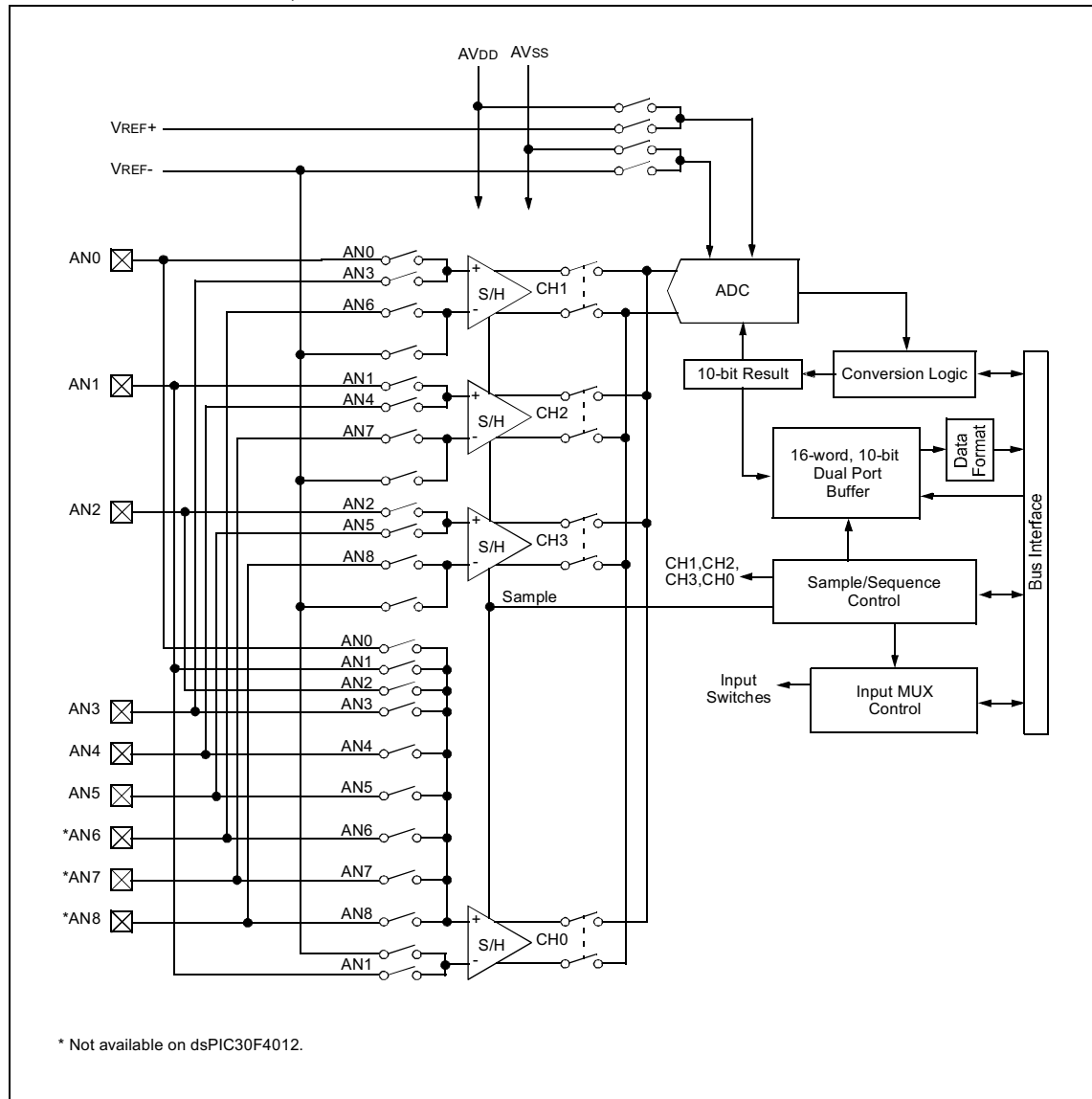
The ADCON1, ADCON2 and ADCON3 registers control the operation of the ADC module. The ADCHS register selects the input channels to be converted. The ADPCFG register configures the port pins as analog inputs or as digital I/O. The ADCSSL register selects inputs for scanning.

Note: The SSRC<2:0>, ASAM, SIMSAM, SMPI<3:0>, BUFM and ALTS bits, as well as the ADCON3 and ADCSSL registers, must not be written to while ADON = 1. This would lead to indeterminate results.

The block diagram of the ADC module is shown in Figure 20-1.

dsPIC30F4011/4012

FIGURE 20-1: 10-BIT, HIGH-SPEED ADC FUNCTIONAL BLOCK DIAGRAM



20.1 A/D Result Buffer

The module contains a 16-word, dual port, read-only buffer, called ADCBUF0...ADCBUFF, to buffer the A/D results. The RAM is 10 bits wide, but is read into different format 16-bit words. The contents of the sixteen A/D Conversion Result Buffer registers, ADCBUF0 through ADCBUFF, cannot be written by user software.

20.2 Conversion Operation

After the ADC module has been configured, the sample acquisition is started by setting the SAMP bit. Various sources, such as a programmable bit, timer time-outs and external events, terminate acquisition and start a conversion. When the A/D conversion is complete, the result is loaded into ADCBUF0...ADCBUFF, and the A/D Interrupt Flag, ADIF, and the DONE bit are set after the number of samples specified by the SMPI<3:0> bits.

The following steps should be followed for doing an A/D conversion:

1. Configure the ADC module:
 - Configure analog pins, voltage reference and digital I/O
 - Select A/D input channels
 - Select A/D conversion clock
 - Select A/D conversion trigger
 - Turn on ADC module
2. Configure the A/D interrupt (if required):
 - Clear ADIF bit
 - Select A/D interrupt priority
3. Start sampling.
4. Wait the required acquisition time.
5. Trigger acquisition end, start conversion.
6. Wait for A/D conversion to complete by either:
 - Waiting for the A/D interrupt
 - Waiting for the DONE bit to get set
7. Read A/D result buffer, clear ADIF if required.

20.3 Selecting the Conversion Sequence

Several groups of control bits select the sequence in which the A/D connects inputs to the sample/hold channels, converts channels, writes the buffer memory and generates interrupts. The sequence is controlled by the sampling clocks.

The SIMSAM bit controls the acquire/convert sequence for multiple channels. If the SIMSAM bit is '0', the two or four selected channels are acquired and converted sequentially with two or four sample clocks. If the SIMSAM bit is '1', two or four selected channels are acquired simultaneously with one sample clock. The channels are then converted sequentially. Obviously, if there is only 1 channel selected, the SIMSAM bit is not applicable.

The CHPS<1:0> bits select how many channels are sampled. This selection can vary from 1, 2 or 4 channels. If the CHPS bits select 1 channel, the CH0 channel is sampled at the sample clock and converted. The result is stored in the buffer. If the CHPS bits select 2 channels, the CH0 and CH1 channels are sampled and converted. If the CHPS bits select 4 channels, the CH0, CH1, CH2 and CH3 channels are sampled and converted.

The SMPI<3:0> bits select the number of acquisition/conversion sequences that would be performed before an interrupt occurs. This can vary from 1 sample per interrupt to 16 samples per interrupt.

The user cannot program a combination of CHPS and SMPI bits that specifies more than 16 conversions per interrupt, or 8 conversions per interrupt, depending on the BUFM bit. The BUFM bit, when set, splits the 16-word results buffer (ADCBUF0...ADCBUFF) into two, 8-word groups. Writing to the 8-word buffers is alternated on each interrupt event. Use of the BUFM bit depends on how much time is available for moving data out of the buffers after the interrupt, as determined by the application.

If the processor can quickly unload a full buffer within the time it takes to acquire and convert one channel, the BUFM bit can be '0' and up to 16 conversions may be done per interrupt. The processor has one sample-and-conversion time to move the sixteen conversions.

If the processor cannot unload the buffer within the acquisition and conversion time, the BUFM bit should be '1'. For example, if SMPI<3:0> (ADCON2<5:2>) = 0111, then eight conversions are loaded into half of the buffer, following which an interrupt occurs. The next eight conversions are loaded into the other half of the buffer. The processor has the entire time between interrupts to move the eight conversions.

The ALTS bit can be used to alternate the inputs selected during the sampling sequence. The input multiplexer has two sets of sample inputs: MUX A and MUX B. If the ALTS bit is '0', only the MUX A inputs are selected for sampling. If the ALTS bit is '1' and SMPI<3:0> = 0000, on the first sample/convert sequence, the MUX A inputs are selected, and on the next acquire/convert sequence, the MUX B inputs are selected.

The CSCNA bit (ADCON2<10>) allows the CH0 channel inputs to be alternately scanned across a selected number of analog inputs for the MUX A group. The inputs are selected by the ADCSSL register. If a particular bit in the ADCSSL register is '1', the corresponding input is selected. The inputs are always scanned from lower to higher numbered inputs, starting after each interrupt. If the number of inputs selected is greater than the number of samples taken per interrupt, the higher numbered inputs are unused.

dsPIC30F4011/4012

20.4 Programming the Start of the Conversion Trigger

The conversion trigger terminates acquisition and starts the requested conversions.

The SSRC<2:0> bits select the source of the conversion trigger.

The SSRC bits provide for up to 5 alternate sources of conversion trigger.

When SSRC<2:0> = 000, the conversion trigger is under software control. Clearing the SAMP bit causes the conversion trigger.

When SSRC<2:0> = 111 (Auto-Start mode), the conversion trigger is under A/D clock control. The SAMC bits select the number of A/D clocks between the start of acquisition and the start of conversion. This provides the fastest conversion rates on multiple channels. SAMC must always be at least 1 clock cycle.

Other trigger sources can come from timer modules, motor control PWM module or external interrupts.

Note: To operate the ADC at the maximum specified conversion speed, the auto-convert trigger option should be selected (SSRC = 111) and the auto-sample time bits should be set to '1' TAD (SAMC = 00001). This configuration gives a total conversion period (sample + convert) of 13 TAD.

The use of any other conversion trigger results in additional TAD cycles to synchronize the external event to the ADC.

20.5 Aborting a Conversion

Clearing the ADON bit during a conversion aborts the current conversion and stops the sampling sequencing. The ADCBUFx is not updated with the partially completed A/D conversion sample. That is, the ADCBUFx will continue to contain the value of the last completed conversion (or the last value written to the ADCBUFx register).

If the clearing of the ADON bit coincides with an auto-start, the clearing has a higher priority.

After the A/D conversion is aborted, a 2 TAD wait is required before the next sampling may be started by setting the SAMP bit.

If sequential sampling is specified, the A/D continues at the next sample pulse, which corresponds with the next channel converted. If simultaneous sampling is specified, the A/D continues with the next multichannel group conversion sequence.

20.6 Selecting the A/D Conversion Clock

The A/D conversion requires 12 TAD. The source of the A/D conversion clock is software selected using a 6-bit counter. There are 64 possible options for TAD.

EQUATION 20-1: A/D CONVERSION CLOCK

$$T_{AD} = T_{CY} * (0.5 * (ADCS<5:0> + 1))$$
$$ADCS<5:0> = 2 \frac{T_{AD}}{T_{CY}} - 1$$

The internal RC oscillator is selected by setting the ADRC bit.

For correct A/D conversions, the A/D conversion clock (TAD) must be selected to ensure a minimum TAD time of 83.33 nsec (for VDD = 5V). Refer to **Section 24.0 "Electrical Characteristics"** for minimum TAD under other operating conditions.

Example 20-1 shows a sample calculation for the ADCS<5:0> bits, assuming a device operating speed of 30 MIPS.

EXAMPLE 20-1: A/D CONVERSION CLOCK CALCULATION

$$T_{AD} = 154 \text{ nsec}$$
$$T_{CY} = 33 \text{ nsec (30 MIPS)}$$

$$ADCS<5:0> = 2 \frac{T_{AD}}{T_{CY}} - 1$$
$$= 2 \cdot \frac{154 \text{ nsec}}{33 \text{ nsec}} - 1$$
$$= 8.33$$

Therefore,
Set ADCS<5:0> = 9

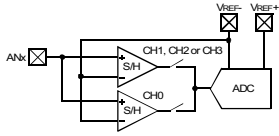
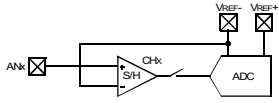
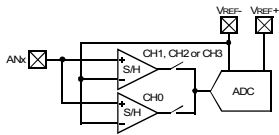
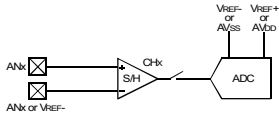
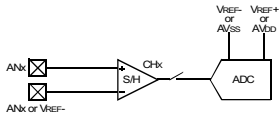
$$\text{Actual } T_{AD} = \frac{T_{CY}}{2} (ADCS<5:0> + 1)$$
$$= \frac{33 \text{ nsec}}{2} (9 + 1)$$
$$= 165 \text{ nsec}$$

dsPIC30F4011/4012

20.7 A/D Conversion Speeds

The dsPIC30F 10-bit ADC specifications permit a maximum 1 Msp/s sampling rate. Table 20-1 summarizes the conversion speeds for the dsPIC30F 10-bit ADC and the required operating conditions.

TABLE 20-1: 10-BIT A/D CONVERSION RATE PARAMETERS

| dsPIC30F 10-bit A/D Converter Conversion Rates | | | | | | |
|--|-------------|--------------------|---------|--------------|-----------------|---|
| A/D Speed | TAD Minimum | Sampling Time Min. | Rs Max. | VDD | Temperature | A/D Channels Configuration |
| Up to 1 Msp/s ⁽¹⁾ | 83.33 ns | 12 TAD | 500Ω | 4.5V to 5.5V | -40°C to +85°C |  |
| Up to 750 ksp/s ⁽¹⁾ | 95.24 ns | 2 TAD | 500Ω | 4.5V to 5.5V | -40°C to +85°C |  |
| Up to 600 ksp/s ⁽¹⁾ | 138.89 ns | 12 TAD | 500Ω | 3.0V to 5.5V | -40°C to +125°C |  |
| Up to 500 ksp/s | 153.85 ns | 1 TAD | 5.0 kΩ | 4.5V to 5.5V | -40°C to +125°C |  |
| Up to 300 ksp/s | 256.41 ns | 1 TAD | 5.0 kΩ | 3.0V to 5.5V | -40°C to +125°C |  |

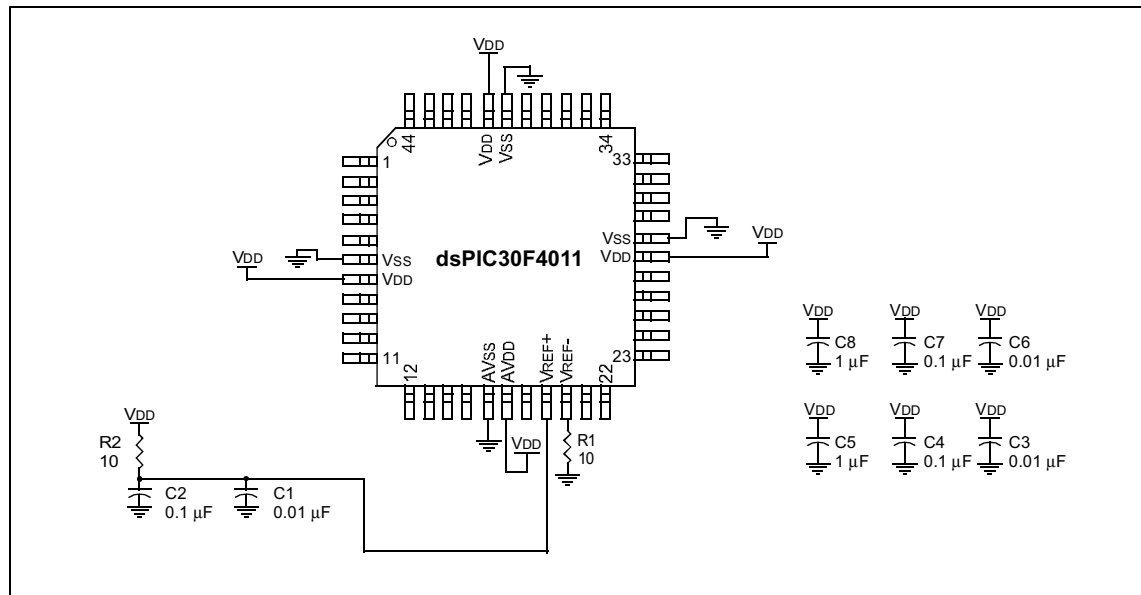
Note 1: External VREF- and VREF+ pins must be used for correct operation. See Figure 20-2 for recommended circuit.

dsPIC30F4011/4012

The configuration guidelines give the required setup values for the conversion speeds above 500 ksp/s, since they require external VREF pins usage and there are some differences in the configuration procedure. Configuration details that are not critical to the conversion speed have been omitted.

Figure 20-2 depicts the recommended circuit for the conversion rates above 500 ksp/s.

FIGURE 20-2: A/D CONVERTER VOLTAGE REFERENCE SCHEMATIC



20.7.1 1 Msp/s CONFIGURATION GUIDELINE

The configuration for 1 Msp/s operation is dependent on whether a single input pin is to be sampled or whether multiple pins are to be sampled.

20.7.1.1 Single Analog Input

For conversions at 1 Msp/s for a single analog input, at least two sample and hold channels must be enabled. The analog input multiplexer must be configured so that the same input pin is connected to both sample and hold channels. The A/D converts the value held on one S/H channel while the second S/H channel acquires a new input sample.

20.7.1.2 Multiple Analog Inputs

The ADC can also be used to sample multiple analog inputs using multiple sample and hold channels. In this case, the total 1 Msp/s conversion rate is divided among the different input signals. For example, four inputs can be sampled at a rate of 250 ksp/s for each signal, or two inputs could be sampled at a rate of 500 ksp/s for each signal. Sequential sampling must be used in this configuration to allow adequate sampling time on each input.

20.7.1.3 1 Msps Configuration Items

The following configuration items are required to achieve a 1 Msps conversion rate.

- Comply with conditions provided in Table 20-2
- Connect external VREF+ and VREF- pins following the recommended circuit shown in Figure 20-2
- Set SSRC<2:0> = 111 in the ADCON1 register to enable the auto-convert option
- Enable automatic sampling by setting the ASAM control bit in the ADCON1 register
- Enable sequential sampling by clearing the SIMSAM bit in the ADCON1 register
- Enable at least two sample and hold channels by writing the CHPS<1:0> control bits in the ADCON2 register
- Write the SMPI<3:0> control bits in the ADCON2 register for the desired number of conversions between interrupts. At a minimum, set SMPI<3:0> = 0001 since at least two sample and hold channels should be enabled
- Configure the A/D clock period to be:

$$\frac{1}{12 \times 1,000,000} = 83.33 \text{ ns}$$

by writing to the ADCS<5:0> control bits in the ADCON3 register

- Configure the sampling time to be 2 TAD by writing: SAMC<4:0> = 00010
- Select at least two channels per analog input pin by writing to the ADCHS register

20.7.2 750 ksps CONFIGURATION GUIDELINE

The following configuration items are required to achieve a 750 ksps conversion rate. This configuration assumes that a single analog input is to be sampled.

- Comply with conditions provided in Table 20-2
- Connect external VREF+ and VREF- pins following the recommended circuit shown in Figure 20-2
- Set SSRC<2:0> = 111 in the ADCON1 register to enable the auto-convert option
- Enable automatic sampling by setting the ASAM control bit in the ADCON1 register
- Enable one sample and hold channel by setting CHPS<1:0> = 00 in the ADCON2 register
- Write the SMPI<3:0> control bits in the ADCON2 register for the desired number of conversions between interrupts
- Configure the A/D clock period to be:

$$\frac{1}{(12 + 2) \times 750,000} = 95.24 \text{ ns}$$

by writing to the ADCS<5:0> control bits in the ADCON3 register

- Configure the sampling time to be 2 TAD by writing: SAMC<4:0> = 00010

20.7.3 600 ksps CONFIGURATION GUIDELINE

The configuration for 600 ksps operation is dependent on whether a single input pin is to be sampled or whether multiple pins are to be sampled.

20.7.3.1 Single Analog Input

When performing conversions at 600 ksps for a single analog input, at least two sample and hold channels must be enabled. The analog input multiplexer must be configured so that the same input pin is connected to both sample and hold channels. The ADC converts the value held on one S/H channel, while the second S/H channel acquires a new input sample.

20.7.3.2 Multiple Analog Input

The ADC can also be used to sample multiple analog inputs using multiple sample and hold channels. In this case, the total 600 ksps conversion rate is divided among the different input signals. For example, four inputs can be sampled at a rate of 150 ksps for each signal or two inputs can be sampled at a rate of 300 ksps for each signal. Sequential sampling must be used in this configuration to allow adequate sampling time on each input.

20.7.3.3 600 ksps Configuration Items

The following configuration items are required to achieve a 600 ksps conversion rate.

- Comply with conditions provided in Table 20-2
- Connect external VREF+ and VREF- pins following the recommended circuit shown in Figure 20-2
- Set SSRC<2:0> = 111 in the ADCON1 register to enable the auto-convert option
- Enable automatic sampling by setting the ASAM control bit in the ADCON1 register
- Enable sequential sampling by clearing the SIMSAM bit in the ADCON1 register
- Enable at least two sample and hold channels by writing the CHPS<1:0> control bits in the ADCON2 register
- Write the SMPI<3:0> control bits in the ADCON2 register for the desired number of conversions between interrupts. At a minimum, set SMPI<3:0> = 0001 since at least two sample and hold channels should be enabled
- Configure the A/D clock period to be:

$$\frac{1}{12 \times 600,000} = 138.89 \text{ ns}$$

by writing to the ADCS<5:0> control bits in the ADCON3 register

- Configure the sampling time to be 2 TAD by writing: SAMC<4:0> = 00010

Select at least two channels per analog input pin by writing to the ADCHS register.

dsPIC30F4011/4012

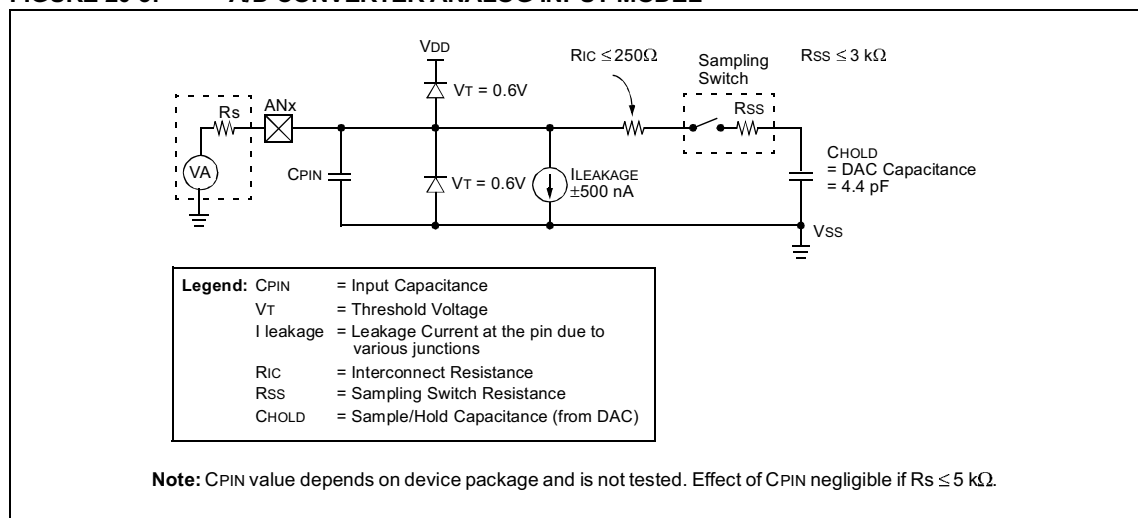
20.8 A/D Acquisition Requirements

The analog input model of the 10-bit ADC is shown in Figure 20-3. The total sampling time for the ADC is a function of the internal amplifier settling time, device V_{DD} and the holding capacitor charge time.

For the ADC to meet its specified accuracy, the charge holding capacitor (C_{HOLD}) must be allowed to fully charge to the voltage level on the analog input pin. The source impedance (R_S), the interconnect impedance (R_{IC}) and the internal sampling switch (R_{SS}) impedance combine to directly affect the time required to charge the capacitor C_{HOLD} . The combined impedance of the analog sources must therefore be small enough to fully charge the holding capacitor within the chosen sample time. To minimize the effects of pin leakage currents on the accuracy of the ADC, the maximum recommended source impedance, R_S , is $5\text{ k}\Omega$. After the analog input channel is selected (changed), this sampling function must be completed prior to starting the conversion. The internal holding capacitor will be in a discharged state prior to each sample operation.

The user must allow at least 1 T_{AD} period of sampling time, T_{SAMP} , between conversions to allow each sample to be acquired. This sample time may be controlled manually in software by setting/clearing the $SAMP$ bit, or it may be automatically controlled by the ADC. In an automatic configuration, the user must allow enough time between conversion triggers so that the minimum sample time can be satisfied. Refer to **Section 24.0 "Electrical Characteristics"** for T_{AD} and sample time requirements.

FIGURE 20-3: A/D CONVERTER ANALOG INPUT MODEL



20.9 Module Power-Down Modes

The module has 3 internal power modes. When the ADON bit is '1', the module is in Active mode; it is fully powered and functional. When ADON is '0', the module is in Off mode. The digital and analog portions of the circuit are disabled for maximum current savings. In order to return to the Active mode from Off mode, the user must wait for the ADC circuitry to stabilize.

20.10 A/D Operation During CPU Sleep and Idle Modes

20.10.1 A/D OPERATION DURING CPU SLEEP MODE

When the device enters Sleep mode, all clock sources to the module are shut down and stay at logic '0'.

If Sleep occurs in the middle of a conversion, the conversion is aborted. The converter will not continue with a partially completed conversion on exit from Sleep mode.

Register contents are not affected by the device entering or leaving Sleep mode.

The ADC module can operate during Sleep mode if the A/D clock source is set to RC (ADRC = 1). When the RC clock source is selected, the ADC module waits one instruction cycle before starting the conversion. This allows the SLEEP instruction to be executed, which eliminates all digital switching noise from the conversion. When the conversion is complete, the DONE bit is set and the result is loaded into the ADCBUFx register.

If the A/D interrupt is enabled, the device wakes up from Sleep. If the A/D interrupt is not enabled, the ADC module is then turned off, although the ADON bit remains set.

20.10.2 A/D OPERATION DURING CPU IDLE MODE

The ADSIDL bit selects if the module stops on Idle or continues on Idle. If ADSIDL = 0, the module continues operation on assertion of Idle mode. If ADSIDL = 1, the module stops on Idle.

20.11 Effects of a Reset

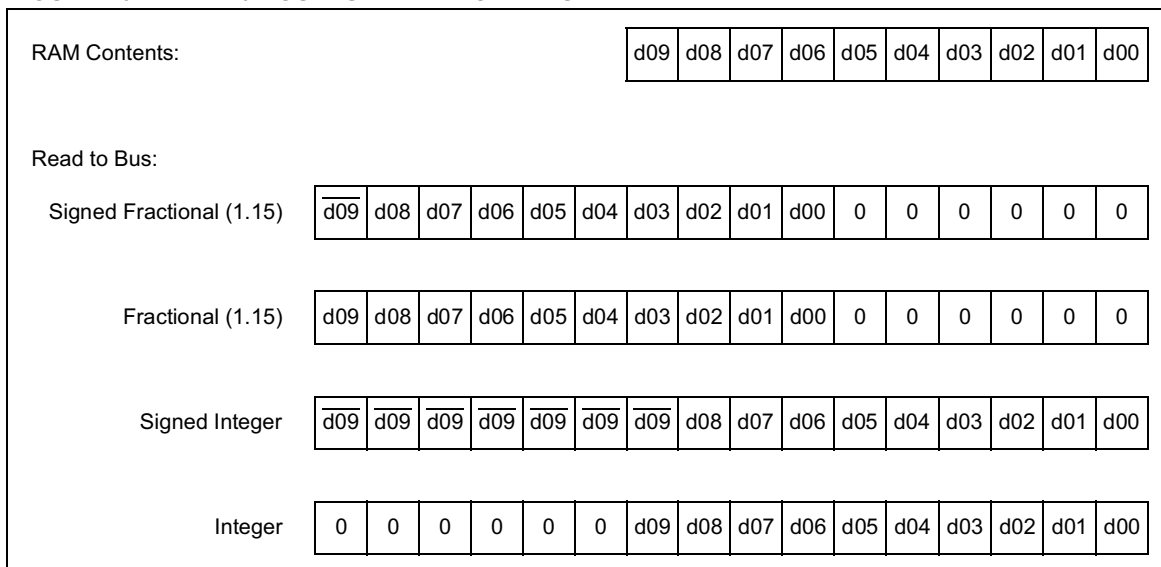
A device Reset forces all registers to their Reset state. This forces the ADC module to be turned off, and any conversion and acquisition sequence is aborted. The values that are in the ADCBUFx registers are not modified. The A/D Result register contains unknown data after a Power-on Reset.

20.12 Output Formats

The A/D result is 10 bits wide. The data buffer RAM is also 10 bits wide. The 10-bit data can be read in one of four different formats. The FORM<1:0> bits select the format. Each of the output formats translates to a 16-bit result on the data bus.

Write data will always be in right justified (integer) format.

FIGURE 20-4: A/D OUTPUT DATA FORMATS



dsPIC30F4011/4012

20.13 Configuring Analog Port Pins

The use of the ADPCFG and TRIS registers control the operation of the ADC port pins. The port pins that are desired as analog inputs must have their corresponding TRIS bit set (input). If the TRIS bit is cleared (output), the digital output level (V_{OH} or V_{OL}) is converted.

The A/D operation is independent of the state of the $CH0SA<3:0>/CH0SB<3:0>$ bits and the TRIS bits.

When reading the PORT register, all pins configured as analog input channels are read as cleared.

Pins configured as digital inputs will not convert an analog input. Analog levels on any pin that is defined as a digital input (including the ANx pins), may cause the input buffer to consume current that exceeds the device specifications.

20.14 Connection Considerations

The analog inputs have diodes to V_{DD} and V_{SS} as ESD protection. This requires that the analog input be between V_{DD} and V_{SS} . If the input voltage exceeds this range by greater than 0.3V (either direction), one of the diodes becomes forward biased and it may damage the device if the input current specification is exceeded.

An external RC filter is sometimes added for anti-aliasing of the input signal. The R component should be selected to ensure that the sampling time requirements are satisfied. Any external components connected (via high-impedance) to an analog input pin (capacitor, zener diode, etc.) should have very little leakage current at the pin.

TABLE 20-2: ADC REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|--------------|-----------|--------|------------|-----------|--------------|-----------|--------|-----------|--------|--------------------|-------|--------|-------|-------|-------|---------------------|
| ADCBUF0 | 0280 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 0 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUF1 | 0282 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 1 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUF2 | 0284 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 2 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUF3 | 0286 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 3 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUF4 | 0288 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 4 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUF5 | 028A | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 5 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUF6 | 028C | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 6 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUF7 | 028E | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 7 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUF8 | 0290 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 8 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUF9 | 0292 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 9 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUFA | 0294 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 10 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUFB | 0296 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 11 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUFC | 0298 | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 12 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUFD | 029A | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 13 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUFE | 029C | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 14 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCBUFF | 029E | — | — | — | — | — | — | — | — | — | — | ADC Data Buffer 15 | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCON1 | 02A0 | ADON | — | ADSIDL | — | — | — | FORM<1:0> | — | SSRC<2:0> | — | — | — | SIMSAM | ASAM | SAMP | DONE | 0000 0000 0000 0000 |
| ADCON2 | 02A2 | — | VCFG<2:0> | — | — | CSCNA | CHPS<1:0> | — | — | BUFS | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCON3 | 02A4 | — | — | — | — | SAMC<4:0> | — | — | — | ADRC | — | — | — | — | — | — | — | 0000 0000 0000 0000 |
| ADCHS | 02A6 | CH123NB<1:0> | CH123SB | CH0NB | CH0SB<3:0> | — | CH123NA<1:0> | CH0NA | — | — | — | CH123SA | CH0SA | — | — | — | — | 0000 0000 0000 0000 |
| ADPCFG | 02A8 | — | — | — | — | — | — | — | PCFG8* | PCFG7* | PCFG6* | PCFG5 | PCFG4 | PCFG3 | PCFG2 | PCFG1 | PCFG0 | 0000 0000 0000 0000 |
| ADCSSL | 02AA | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | 0000 0000 0000 0000 |

Legend: u = uninitialized bit

* Not available on dsPIC30F4012.

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

dsPIC30F4011/4012

NOTES:

21.0 SYSTEM INTEGRATION

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

There are several features intended to maximize system reliability, minimize cost through elimination of external components, provide power-saving operating modes and offer code protection:

- Oscillator Selection
- Reset
 - Power-on Reset (POR)
 - Power-up Timer (PWRT)
 - Oscillator Start-up Timer (OST)
 - Programmable Brown-out Reset (BOR)
- Watchdog Timer (WDT)
- Power-Saving Modes (Sleep and Idle)
- Code Protection
- Unit ID Locations
- In-Circuit Serial Programming (ICSP)

dsPIC30F devices have a Watchdog Timer which is permanently enabled via the Configuration bits, or can be software controlled. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable. The other is the Power-up Timer (PWRT), which provides a delay on power-up only, designed to keep the part in Reset while the power supply stabilizes. With these two timers on-chip, most applications need no external Reset circuitry.

Sleep mode is designed to offer a very low-current Power-Down mode. The user can wake-up from Sleep through external Reset, Watchdog Timer wake-up or through an interrupt. Several oscillator options are also made available to allow the part to fit a wide variety of applications. In the Idle mode, the clock sources are still active, but the CPU is shut off. The RC oscillator option saves system cost, while the LP crystal option saves power.

21.1 Oscillator System Overview

The dsPIC30F oscillator system has the following modules and features:

- Various external and internal oscillator options as clock sources
- An on-chip PLL to boost internal operating frequency
- A clock switching mechanism between various clock sources
- Programmable clock postscaler for system power savings
- A Fail-Safe Clock Monitor (FSCM) that detects clock failure and takes fail-safe measures
- Clock Control register (OSCCON)
- Configuration bits for main oscillator selection

Table 21-1 provides a summary of the dsPIC30F oscillator operating modes. A simplified diagram of the oscillator system is shown in Figure 21-1.

Configuration bits determine the clock source upon Power-on Reset (POR) and Brown-out Reset (BOR). Thereafter, the clock source can be changed between permissible clock sources. The OSCCON register controls the clock switching and reflects system clock related status bits.

dsPIC30F4011/4012

TABLE 21-1: OSCILLATOR OPERATING MODES

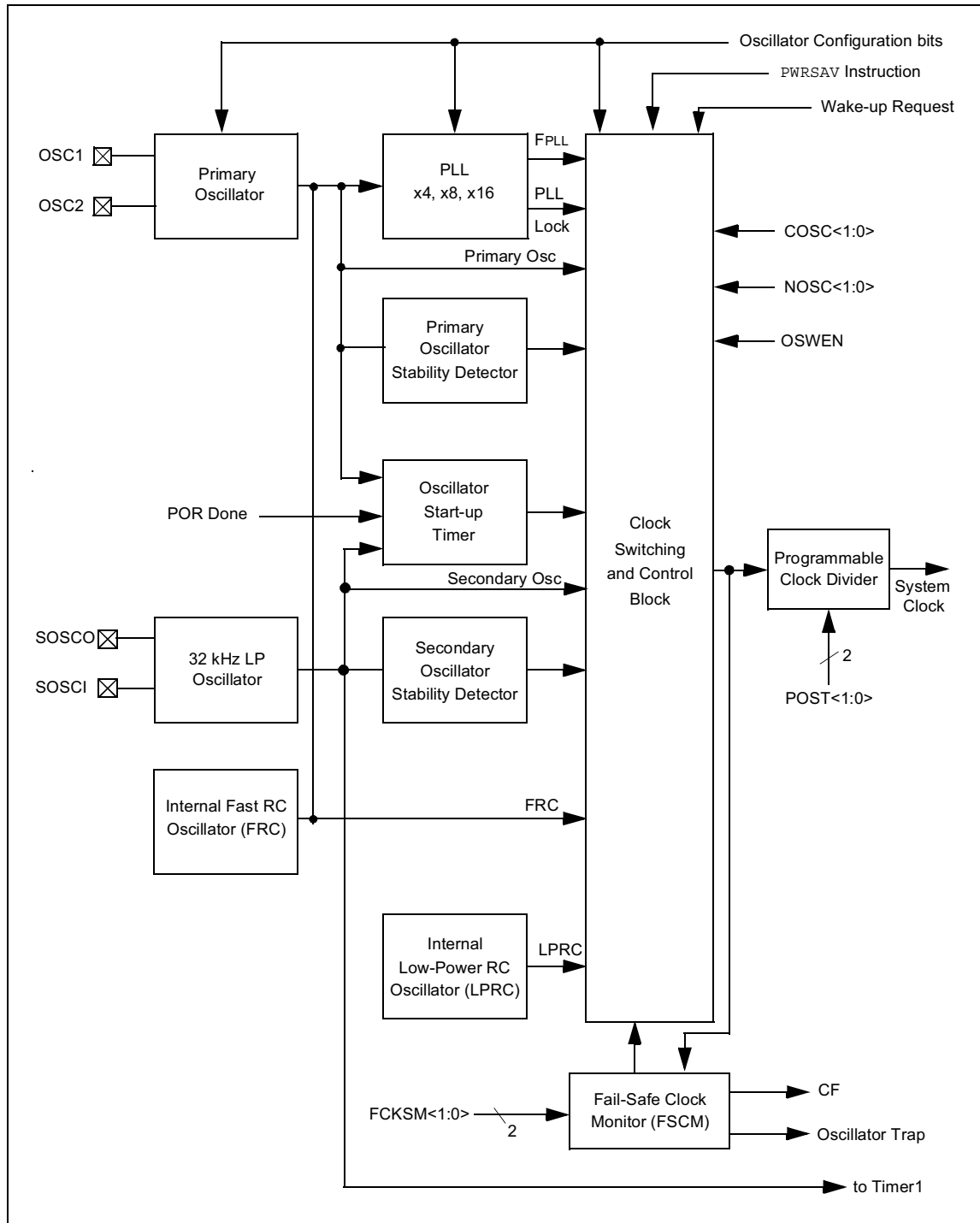
| Oscillator Mode | Description |
|-----------------|--|
| XTL | 200 kHz-4 MHz crystal on OSC1:OSC2 |
| XT | 4 MHz-10 MHz crystal on OSC1:OSC2 |
| XT w/PLL 4x | 4 MHz-10 MHz crystal on OSC1:OSC2, 4x PLL enabled |
| XT w/PLL 8x | 4 MHz-10 MHz crystal on OSC1:OSC2, 8x PLL enabled |
| XT w/PLL 16x | 4 MHz-10 MHz crystal on OSC1:OSC2, 16x PLL enabled ⁽¹⁾ |
| LP | 32 kHz crystal on SOSCO:SOSCI ⁽²⁾ |
| HS | 10 MHz-25 MHz crystal |
| EC | External clock input (0-40 MHz) |
| ECIO | External clock input (0-40 MHz), OSC2 pin is I/O |
| EC w/PLL 4x | External clock input (0-40 MHz), OSC2 pin is I/O, 4x PLL enabled ⁽¹⁾ |
| EC w/PLL 8x | External clock input (0-40 MHz), OSC2 pin is I/O, 8x PLL enabled ⁽¹⁾ |
| EC w/PLL 16x | External clock input (0-40 MHz), OSC2 pin is I/O, 16x PLL enabled ⁽¹⁾ |
| ERC | External RC oscillator, OSC2 pin is Fosc/4 output ⁽³⁾ |
| ERCIO | External RC oscillator, OSC2 pin is I/O ⁽³⁾ |
| FRC | 8 MHz internal RC oscillator |
| FRC w/PLL 4x | 7.37 MHz Internal RC oscillator, 4x PLL enabled |
| FRC w/PLL 8x | 7.37 MHz Internal RC oscillator, 8x PLL enabled |
| FRC w/PLL 16x | 7.37 MHz Internal RC oscillator, 16x PLL enabled |
| LPRC | 512 kHz internal RC oscillator |

Note 1: The dsPIC30F maximum operating frequency of 120 MHz must be met.

2: LP oscillator can be conveniently shared as a system clock, as well as a Real-Time Clock for Timer1.

3: Requires external R and C. Frequency operation up to 4 MHz.

FIGURE 21-1: OSCILLATOR SYSTEM BLOCK DIAGRAM



dsPIC30F4011/4012

21.2 Oscillator Configurations

21.2.1 INITIAL CLOCK SOURCE SELECTION

While coming out of Power-on Reset or Brown-out Reset, the device selects its clock source based on:

- The FOS<1:0> Configuration bits that select one of four oscillator groups.
- The FPR<3:0> Configuration bits that select one of 13 oscillator choices within the primary group.

The selection is as shown in Table 21-2.

21.2.2 OSCILLATOR START-UP TIMER (OST)

In order to ensure that a crystal oscillator (or ceramic resonator) has started and stabilized, an Oscillator Start-up Timer (OST) is included. It is a simple, 10-bit counter that counts 1024 T_{osc} cycles before releasing the oscillator clock to the rest of the system. The time-out period is designated as T_{OST}. The T_{OST} time is involved every time the oscillator has to restart (i.e., on POR, BOR and wake-up from Sleep). The Oscillator Start-up Timer is applied to the LP, XT, XTL and HS modes (upon wake-up from Sleep, POR and BOR) for the primary oscillator.

TABLE 21-2: CONFIGURATION BIT VALUES FOR CLOCK SELECTION

| Oscillator Mode | Oscillator Source | FOS1 | FOS0 | FPR3 | FPR2 | FPR1 | FPR0 | OSC2 Function |
|-----------------|-------------------|------|------|------|------|------|------|---------------|
| EC | Primary | 1 | 1 | 1 | 0 | 1 | 1 | CLKO |
| ECIO | Primary | 1 | 1 | 1 | 1 | 0 | 0 | I/O |
| EC w/PLL 4x | Primary | 1 | 1 | 1 | 1 | 0 | 1 | I/O |
| EC w/PLL 8x | Primary | 1 | 1 | 1 | 1 | 1 | 0 | I/O |
| EC w/PLL 16x | Primary | 1 | 1 | 1 | 1 | 1 | 1 | I/O |
| ERC | Primary | 1 | 1 | 1 | 0 | 0 | 1 | CLKO |
| ERCIO | Primary | 1 | 1 | 1 | 0 | 0 | 0 | I/O |
| XT | Primary | 1 | 1 | 0 | 1 | 0 | 0 | OSC2 |
| XT w/PLL 4x | Primary | 1 | 1 | 0 | 1 | 0 | 1 | OSC2 |
| XT w/PLL 8x | Primary | 1 | 1 | 0 | 1 | 1 | 0 | OSC2 |
| XT w/PLL 16x | Primary | 1 | 1 | 0 | 1 | 1 | 1 | OSC2 |
| XTL | Primary | 1 | 1 | 0 | 0 | 0 | 0 | OSC2 |
| HS | Primary | 1 | 1 | 0 | 0 | 1 | 0 | OSC2 |
| FRC w/PLL 4x | Primary | 1 | 1 | 0 | 0 | 0 | 1 | I/O |
| FRC w/PLL 8x | Primary | 1 | 1 | 1 | 0 | 1 | 0 | I/O |
| FRC w/PLL 16x | Primary | 1 | 1 | 0 | 0 | 1 | 1 | I/O |
| LP | Secondary | 0 | 0 | — | — | — | — | (Notes 1, 2) |
| FRC | Internal FRC | 0 | 1 | — | — | — | — | (Notes 1, 2) |
| LPRC | Internal LPRC | 1 | 0 | — | — | — | — | (Notes 1, 2) |

Note 1: OSC2 pin function is determined by the Primary Oscillator mode selection (FPR<3:0>).

Note 2: Note that the OSC1 pin cannot be used as an I/O pin, even if the secondary oscillator or an internal clock source is selected at all times.

21.2.3 LP OSCILLATOR CONTROL

Enabling the LP oscillator is controlled with two elements:

1. The current oscillator group bits, COSC<1:0>.
2. The LPOSCEN bit (OSCCON register).

The LP oscillator is on (even during Sleep mode) if LPOSCEN = 1. The LP oscillator is the device clock if:

- COSC<1:0> = 00 (LP selected as main oscillator) and
- LPOSCEN = 1

Keeping the LP oscillator on at all times allows for a fast switch to the 32 kHz system clock for lower power operation. Returning to the faster main oscillator still requires a start-up time.

21.2.4 PHASE LOCKED LOOP (PLL)

The PLL multiplies the clock which is generated by the primary oscillator. The PLL is selectable to have either gains of x4, x8 or x16. Input and output frequency ranges are summarized in Table 21-3.

TABLE 21-3: PLL FREQUENCY RANGE

| F _{IN} | PLL Multiplier | F _{OUT} |
|-----------------|----------------|------------------|
| 4 MHz-10 MHz | x4 | 16 MHz-40 MHz |
| 4 MHz-10 MHz | x8 | 32 MHz-80 MHz |
| 4 MHz-7.5 MHz | x16 | 64 MHz-120 MHz |

The PLL features a lock output which is asserted when the PLL enters a phase locked state. Should the loop fall out of lock (e.g., due to noise), the lock signal is rescinded. The state of this signal is reflected in the read-only LOCK bit in the OSCCON register.

21.2.5 FAST RC OSCILLATOR (FRC)

The FRC oscillator is a fast (7.37 MHz, +/-2% nominal), internal RC oscillator. This oscillator is intended to provide reasonable device operating speeds without the use of an external crystal, ceramic resonator or RC network. Using the x4, x8 and x16 PLL options, higher operational frequencies can be generated.

The dsPIC30F operates from the FRC oscillator whenever the Current Oscillator Selection (COSC<1:0>) control bits in the OSCCON register (OSCCON<13:12>) are set to '01'.

There are four tuning bits (TUN<3:0>) for the FRC oscillator in the OSCCON register. These tuning bits allow the FRC oscillator frequency to be adjusted as close to 7.37 MHz as possible, depending on the device operating conditions. The FRC oscillator frequency has been calibrated during factory testing. Table 21-4 describes the adjustment range of the TUN<3:0> bits.

TABLE 21-4: FRC TUNING

| TUN<3:0> Bits | FRC Frequency |
|---------------|--|
| 0111 | +10.5% |
| 0110 | +9.0% |
| 0101 | +7.5% |
| 0100 | +6.0% |
| 0011 | +4.5% |
| 0010 | +3.0% |
| 0001 | +1.5% |
| 0000 | Center Frequency (oscillator is running at calibrated frequency) |
| 1111 | -1.5% |
| 1110 | -3.0% |
| 1101 | -4.5% |
| 1100 | -6.0% |
| 1011 | -7.5% |
| 1010 | -9.0% |
| 1001 | -10.5% |
| 1000 | -12.0% |

21.2.6 LOW-POWER RC OSCILLATOR (LPRC)

The LPRC oscillator is a component of the Watchdog Timer (WDT) and oscillates at a nominal frequency of 512 kHz. The LPRC oscillator is the clock source for the Power-up Timer (PWRT) circuit, WDT and clock monitor circuits. It may also be used to provide a low-frequency clock source option for applications where power consumption is critical and timing accuracy is not required.

The LPRC oscillator is always enabled at a Power-on Reset because it is the clock source for the PWRT. After the PWRT expires, the LPRC oscillator remains on if one of the following is true:

- The Fail-Safe Clock Monitor is enabled
- The WDT is enabled
- The LPRC oscillator is selected as the system clock via the COSC<1:0> control bits in the OSCCON register

If one of the above conditions is not true, the LPRC shuts off after the PWRT expires.

Note 1: OSC2 pin function is determined by the Primary Oscillator mode selection (FPR<3:0>).

2: Note that OSC1 pin cannot be used as an I/O pin, even if the secondary oscillator or an internal clock source is selected at all times.

dsPIC30F4011/4012

21.2.7 FAIL-SAFE CLOCK MONITOR

The Fail-Safe Clock Monitor (FSCM) allows the device to continue to operate even in the event of an oscillator failure. The FSCM function is enabled by appropriately programming the FCKSM<1:0> Configuration bits (Clock Switch and Monitor Selection bits) in the Fosc device Configuration register. If the FSCM function is enabled, the LPRC internal oscillator runs at all times (except during Sleep mode) and is not subject to control by the SWDTEN bit.

In the event of an oscillator failure, the FSCM generates a clock failure trap event and switches the system clock over to the FRC oscillator. The user then has the option to either attempt to restart the oscillator or execute a controlled shutdown. The user may decide to treat the trap as a warm Reset by simply loading the Reset address into the oscillator fail trap vector. In this event, the CF (Clock Fail) status bit (OSCCON<3>) is also set whenever a clock failure is recognized.

In the event of a clock failure, the WDT is unaffected and continues to run on the LPRC clock.

If the oscillator has a very slow start-up time coming out of POR, BOR or Sleep, it is possible that the PWRT timer will expire before the oscillator has started. In such cases, the FSCM is activated. The FSCM initiates a clock failure trap, and the COSC<1:0> bits are loaded with the Fast RC (FRC) oscillator selection. This effectively shuts off the original oscillator that was trying to start.

The user may detect this situation and restart the oscillator in the clock fail trap Interrupt Service Routine (ISR).

Upon a clock failure detection, the FSCM module initiates a clock switch to the FRC oscillator as follows:

1. The COSC<1:0> bits (OSCCON<13:12>) are loaded with the FRC oscillator selection value.
2. CF bit is set (OSCCON<3>).
3. OSWEN control bit (OSCCON<0>) is cleared.

For the purpose of clock switching, the clock sources are sectioned into four groups:

1. Primary
2. Secondary
3. Internal FRC
4. Internal LPRC

The user can switch between these functional groups but cannot switch between options within a group. If the primary group is selected, then the choice within the group is always determined by the FPR<3:0> Configuration bits.

The OSCCON register holds the control and status bits related to clock switching.

- COSC<1:0>: Read-only status bits always reflect the current oscillator group in effect.
- NOSC<1:0>: Control bits which are written to indicate the new oscillator group of choice.
 - On POR and BOR, COSC<1:0> and NOSC<1:0> are both loaded with the Configuration bit values, FOS<1:0>.
- LOCK: The LOCK status bit indicates a PLL lock.
- CF: Read-only status bit indicating if a clock fail detect has occurred.
- OSWEN: Control bit changes from a '0' to a '1' when a clock transition sequence is initiated. Clearing the OSWEN control bit aborts a clock transition in progress (used for hang-up situations).

If Configuration bits, FCKSM<1:0> = 1x, then the clock switching and Fail-Safe Clock Monitor functions are disabled. This is the default Configuration bit setting.

If clock switching is disabled, then the FOS<1:0> and FPR<3:0> bits directly control the oscillator selection, and the COSC<1:0> bits do not control the clock selection. However, these bits do reflect the clock source selection.

Note: The application should not attempt to switch to a clock of frequency lower than 100 kHz when the Fail-Safe Clock Monitor is enabled. If such clock switching is performed, the device may generate an oscillator fail trap and switch to the Fast RC (FRC) oscillator.

21.2.8 PROTECTION AGAINST ACCIDENTAL WRITES TO OSCCON

A write to the OSCCON register is intentionally made difficult because it controls clock switching and clock scaling.

To write to the OSCCON low byte, the following code sequence must be executed without any other instructions in between:

- Byte Write "0x46" to OSCCON low
- Byte Write "0x57" to OSCCON low

Byte Write is allowed for one instruction cycle. Write the desired value or use bit manipulation instruction.

To write to the OSCCON high byte, the following instructions must be executed without any other instructions in between:

- Byte Write "0x78" to OSCCON high
- Byte Write "0x9A" to OSCCON high

Byte Write is allowed for one instruction cycle. Write the desired value or use bit manipulation instruction.

dsPIC30F4011/4012

21.3 Reset

The dsPIC30F4011/4012 devices differentiate between various kinds of Reset:

- Power-on Reset (POR)
- $\overline{\text{MCLR}}$ Reset during normal operation
- $\overline{\text{MCLR}}$ Reset during Sleep
- Watchdog Timer (WDT) Reset (during normal operation)
- Programmable Brown-out Reset (BOR)
- RESET Instruction
- Reset caused by trap lock-up (TRAPR)
- Reset caused by illegal opcode, or by using an uninitialized W register as an Address Pointer (IOPUWR)

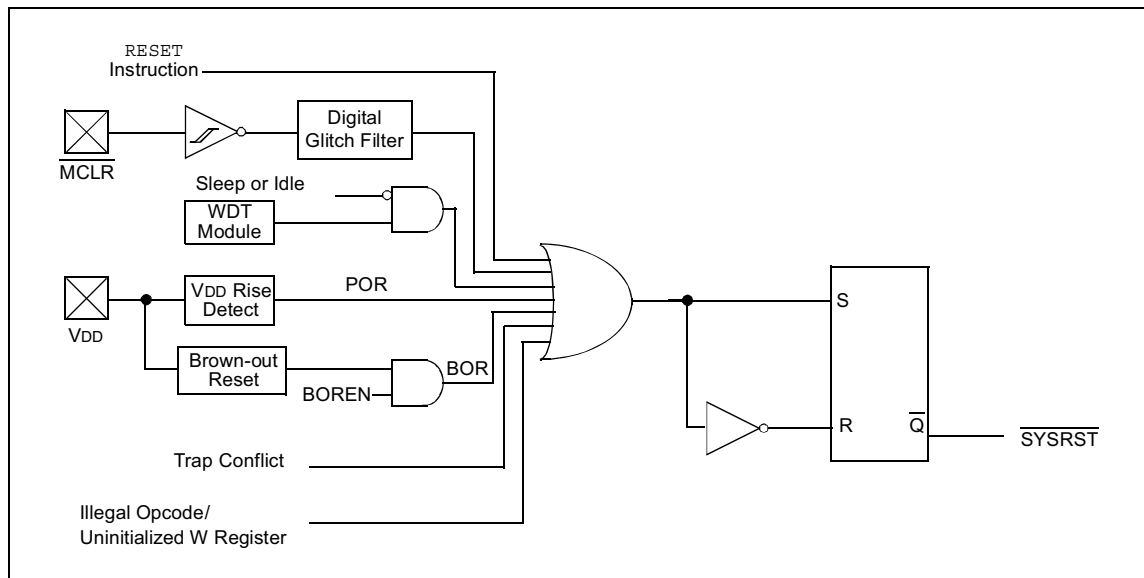
Different registers are affected in different ways by various Reset conditions. Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register are set or cleared differently in different Reset situations, as indicated in Table 21-5. These bits are used in software to determine the nature of the Reset.

A block diagram of the on-chip Reset circuit is shown in Figure 21-2.

A $\overline{\text{MCLR}}$ noise filter is provided in the $\overline{\text{MCLR}}$ Reset path. The filter detects and ignores small pulses.

Internally generated Resets do not drive $\overline{\text{MCLR}}$ pin low.

FIGURE 21-2: RESET SYSTEM BLOCK DIAGRAM



21.3.1 POR: POWER-ON RESET

A power-on event generates an internal POR pulse when a VDD rise is detected. The Reset pulse occurs at the POR circuit threshold voltage (VPOR) which is nominally 1.85V. The device supply voltage characteristics must meet specified starting voltage and rise rate requirements. The POR pulse resets a POR timer and places the device in the Reset state. The POR also selects the device clock source identified by the oscillator Configuration fuses.

The POR circuit inserts a small delay, TPOR, which is nominally 10 μs and ensures that the device bias circuits are stable. Furthermore, a user-selected power-up time-out (TPWRT) is applied. The TPWRT parameter is based on device Configuration bits and can be 0 ms (no delay), 4 ms, 16 ms or 64 ms. The total delay is at device power-up, TPOR + TPWRT. When these delays have expired, $\overline{\text{SYSRST}}$ will be negated on the next leading edge of the Q1 clock and the PC jumps to the Reset vector.

The timing for the $\overline{\text{SYSRST}}$ signal is shown in Figure 21-3 through Figure 21-5.

dsPIC30F4011/4012

FIGURE 21-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO V_{DD})

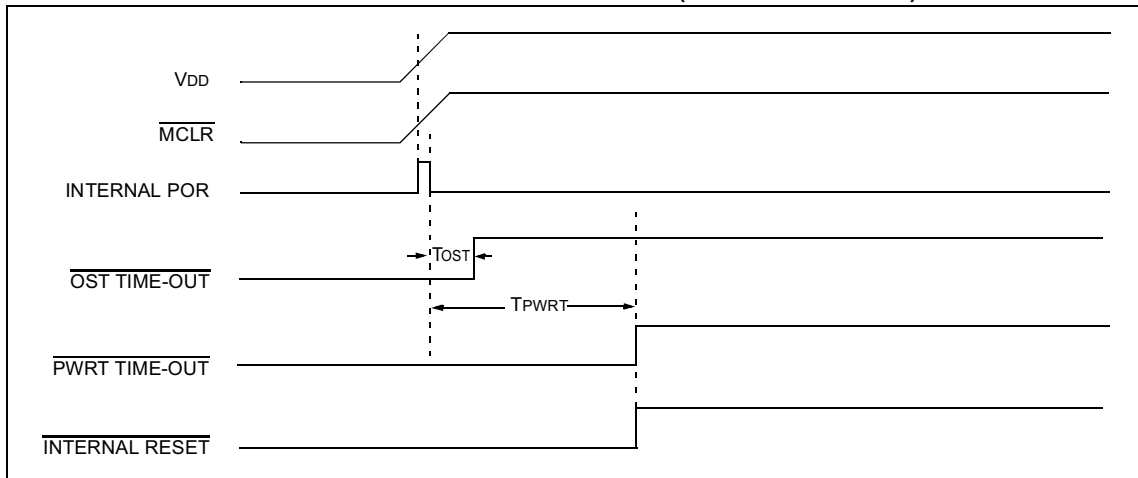


FIGURE 21-4: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO V_{DD}): CASE 1

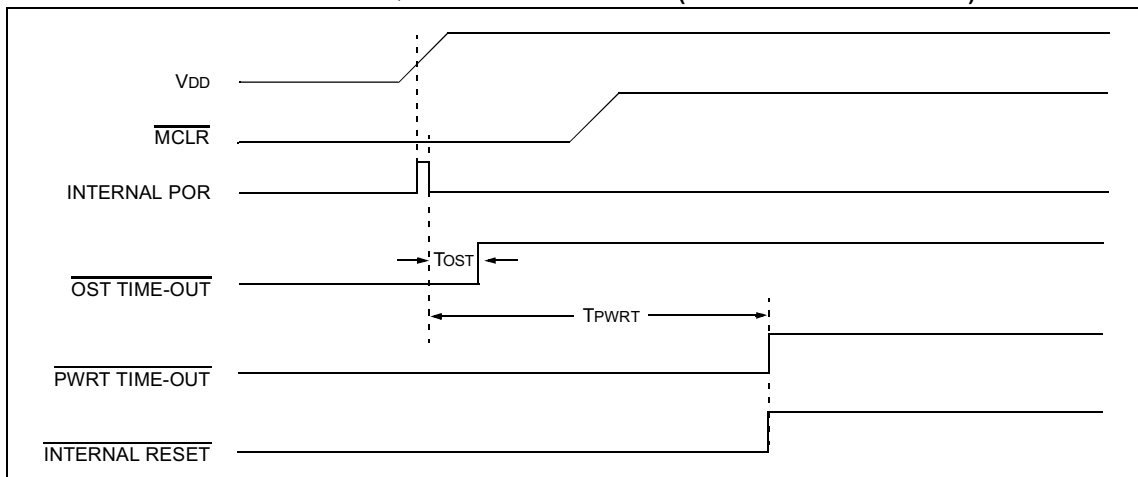
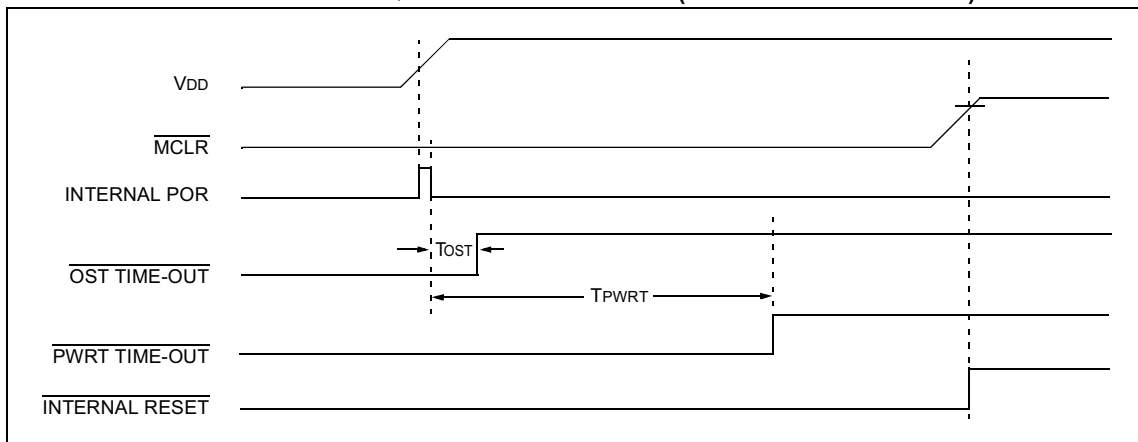


FIGURE 21-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO V_{DD}): CASE 2



dsPIC30F4011/4012

21.3.1.1 POR with Long Crystal Start-up Time (with FSCM Enabled)

The oscillator start-up circuitry is not linked to the POR circuitry. Some crystal circuits (especially low-frequency crystals) have a relatively long start-up time. Therefore, one or more of the following conditions is possible after the POR timer and the PWRT have expired:

- The oscillator circuit has not begun to oscillate.
- The Oscillator Start-up Timer has not expired (if a crystal oscillator is used).
- The PLL has not achieved a lock (if PLL is used).

If the FSCM is enabled and one of the above conditions is true, then a clock failure trap occurs. The device automatically switches to the FRC oscillator and the user can switch to the desired crystal oscillator in the trap ISR.

21.3.1.2 Operating without FSCM and PWRT

If the FSCM is disabled and the Power-up Timer (PWRT) is also disabled, then the device exits rapidly from Reset on power-up. If the clock source is FRC, LPRC, ERC or EC, it will be active immediately.

If the FSCM is disabled and the system clock has not started, the device will be in a frozen state at the Reset vector until the system clock starts. From the user's perspective, the device will appear to be in Reset until a system clock is available.

21.3.2 BOR: PROGRAMMABLE BROWN-OUT RESET

The BOR (Brown-out Reset) module is based on an internal voltage reference circuit. The main purpose of the BOR module is to generate a device Reset when a brown-out condition occurs. Brown-out conditions are generally caused by glitches on the AC mains (i.e., missing portions of the AC cycle waveform due to bad power transmission lines, or voltage sags due to excessive current draw when a large inductive load is turned on).

The BOR module allows selection of one of the following voltage trip points (see Table 24-10):

- 2.6V-2.71V
- 4.1V-4.4V
- 4.58V-4.73V

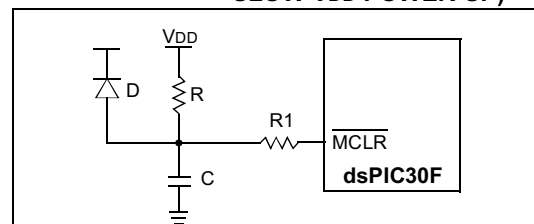
Note: The BOR voltage trip points indicated here are nominal values provided for design guidance only.

A BOR generates a Reset pulse, which resets the device. The BOR selects the clock source, based on the device Configuration bit values (FOS<1:0> and FPR<3:0>). Furthermore, if an oscillator mode is selected, the BOR activates the Oscillator Start-up Timer (OST). The system clock is held until OST expires. If the PLL is used, then the clock is held until the LOCK bit (OSCCON<5>) is '1'.

Concurrently, the POR time-out (TPOR) and the PWRT time-out (TPWRT) are applied before the internal Reset is released. If TPWRT = 0 and a crystal oscillator is being used, then a nominal delay of TFSCM = 100 μ s is applied. The total delay in this case is (TPOR + TFSCM).

The BOR status bit (RCON<1>) is set to indicate that a BOR has occurred. The BOR circuit, if enabled, continues to operate while in Sleep or Idle modes and resets the device if VDD falls below the BOR threshold voltage.

FIGURE 21-6: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)



- Note 1:** External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.
- 2:** R should be suitably chosen so as to make sure that the voltage drop across R does not violate the device's electrical specification.
- 3:** R1 should be suitably chosen so as to limit any current flowing into MCLR from external capacitor C, in the event of MCLR pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

Note: Dedicated supervisory devices, such as the MCP1XX and MCP8XX, may also be used as an external Power-on Reset circuit.

dsPIC30F4011/4012

Table 21-5 shows the Reset conditions for the RCON Register. Since the control bits within the RCON register are R/W, the information in the table implies that all the bits are negated prior to the action specified in the condition column.

TABLE 21-5: INITIALIZATION CONDITION FOR RCON REGISTER, CASE 1

| Condition | Program Counter | TRAPR | IOPUWR | EXTR | SWR | WDTO | IDLE | SLEEP | POR | BOR |
|--|-----------------------|-------|--------|------|-----|------|------|-------|-----|-----|
| Power-on Reset | 0x000000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Brown-out Reset | 0x000000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| MCLR Reset during normal operation | 0x000000 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Software Reset during normal operation | 0x000000 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| MCLR Reset during Sleep | 0x000000 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| MCLR Reset during Idle | 0x000000 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| WDT Time-out Reset | 0x000000 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| WDT Wake-up | PC + 2 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| Interrupt Wake-up from Sleep | PC + 2 ⁽¹⁾ | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Clock Failure Trap | 0x000004 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Trap Reset | 0x000000 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Illegal Operation Trap | 0x000000 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Note 1: When the wake-up is due to an enabled interrupt, the PC is loaded with the corresponding interrupt vector.

Table 21-6 shows a second example of the bit conditions for the RCON register. In this case, it is not assumed that the user has set/cleared specific bits prior to action specified in the condition column.

TABLE 21-6: INITIALIZATION CONDITION FOR RCON REGISTER, CASE 2

| Condition | Program Counter | TRAPR | IOPUWR | EXTR | SWR | WDTO | IDLE | SLEEP | POR | BOR |
|--|-----------------------|-------|--------|------|-----|------|------|-------|-----|-----|
| Power-on Reset | 0x000000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| Brown-out Reset | 0x000000 | u | u | u | u | u | u | u | 0 | 1 |
| MCLR Reset during normal operation | 0x000000 | u | u | 1 | 0 | 0 | 0 | 0 | u | u |
| Software Reset during normal operation | 0x000000 | u | u | 0 | 1 | 0 | 0 | 0 | u | u |
| MCLR Reset during Sleep | 0x000000 | u | u | 1 | u | 0 | 0 | 1 | u | u |
| MCLR Reset during Idle | 0x000000 | u | u | 1 | u | 0 | 1 | 0 | u | u |
| WDT Time-out Reset | 0x000000 | u | u | 0 | 0 | 1 | 0 | 0 | u | u |
| WDT Wake-up | PC + 2 | u | u | u | u | 1 | u | 1 | u | u |
| Interrupt Wake-up from Sleep | PC + 2 ⁽¹⁾ | u | u | u | u | u | u | 1 | u | u |
| Clock Failure Trap | 0x000004 | u | u | u | u | u | u | u | u | u |
| Trap Reset | 0x000000 | 1 | u | u | u | u | u | u | u | u |
| Illegal Operation Reset | 0x000000 | u | 1 | u | u | u | u | u | u | u |

Legend: u = unchanged

Note 1: When the wake-up is due to an enabled interrupt, the PC is loaded with the corresponding interrupt vector.

21.4 Watchdog Timer (WDT)

21.4.1 WATCHDOG TIMER OPERATION

The primary function of the Watchdog Timer (WDT) is to reset the processor in the event of a software malfunction. The WDT is a free-running timer that runs off an on-chip RC oscillator, requiring no external component. Therefore, the WDT timer continues to operate even if the main processor clock (e.g., the crystal oscillator) fails.

21.4.2 ENABLING AND DISABLING THE WDT

The Watchdog Timer can be “Enabled” or “Disabled” only through a Configuration bit (FWDTEN) in the Configuration register, FWDT.

Setting FWDTEN = 1 enables the Watchdog Timer. The enabling is done when programming the device. By default, after chip erase, FWDTEN bit = 1. Any device programmer capable of programming dsPIC30F devices allows programming of this and other Configuration bits.

If enabled, the WDT increments until it overflows or “times out”. A WDT time-out forces a device Reset (except during Sleep). To prevent a WDT time-out, the user must clear the Watchdog Timer using a CLRWDT instruction.

If a WDT times out during Sleep, the device wakes-up. The WDTO bit in the RCON register is cleared to indicate a wake-up resulting from a WDT time-out.

Setting FWDTEN = 0 allows user software to enable/disable the Watchdog Timer via the SWDTEN (RCON<5>) control bit.

21.5 Power-Saving Modes

There are two power-saving states that can be entered through the execution of a special instruction, PWRSAV. These are: Sleep and Idle.

The format of the PWRSAV instruction is as follows:

PWRSAV <parameter>, where ‘parameter’ defines Idle or Sleep mode.

21.5.1 SLEEP MODE

In Sleep mode, the clock to the CPU and peripherals is shut down. If an on-chip oscillator is being used, it is shut down.

The Fail-Safe Clock Monitor is not functional during Sleep, since there is no clock to monitor. However, the LPRC clock remains active if WDT is operational during Sleep.

The Brown-out Reset protection circuit and the Low-Voltage Detect (LVD) circuit, if enabled, remain functional during Sleep.

The processor wakes up from Sleep if at least one of the following conditions has occurred:

- any interrupt that is individually enabled and meets the required priority level
- any Reset (POR, BOR and MCLR)
- WDT time-out

On waking up from Sleep mode, the processor restarts the same clock that was active prior to entry into Sleep mode. When clock switching is enabled, bits COSC<1:0> determine the oscillator source to be used on wake-up. If clock switch is disabled, then there is only one system clock.

| |
|--|
| Note: If a POR or BOR occurred, the selection of the oscillator is based on the FOS<1:0> and FPR<3:0> Configuration bits. |
|--|

If the clock source is an oscillator, the clock to the device is held off until OST times out (indicating a stable oscillator). If PLL is used, the system clock is held off until LOCK = 1 (indicating that the PLL is stable). In either case, TPOR, TLOCK and TPWRT delays are applied.

If EC, FRC, LPRC or ERC oscillators are used, then a delay of TPOR (~10 µs) is applied. This is the smallest delay possible on wake-up from Sleep.

Moreover, if the LP oscillator was active during Sleep, and LP is the oscillator used on wake-up, then the start-up delay is equal to TPOR. PWRT and OST delays are not applied. In order to have the smallest possible start-up delay when waking up from Sleep, one of these faster wake-up options should be selected before entering Sleep.

dsPIC30F4011/4012

Any interrupt that is individually enabled (using the corresponding IE bit) and meets the prevailing priority level can wake-up the processor. The processor processes the interrupt and branches to the ISR. The SLEEP status bit in the RCON register is set upon wake-up.

Note: In spite of various delays applied (TPOR, TLOCK and TPWRT), the crystal oscillator (and PLL) may not be active at the end of the time-out (e.g., for low-frequency crystals). In such cases, if FSCM is enabled, the device detects this condition as a clock failure and processes the clock failure trap. The FRC oscillator is enabled, and the user must re-enable the crystal oscillator. If FSCM is not enabled, then the device simply suspends execution of code until the clock is stable and remains in Sleep until the oscillator clock has started.

All Resets wake-up the processor from Sleep mode. Any Reset, other than POR, sets the SLEEP status bit. In a POR, the SLEEP bit is cleared.

If Watchdog Timer is enabled, the processor wakes-up from Sleep mode upon WDT time-out. The SLEEP and WDTO status bits are both set.

21.5.2 IDLE MODE

In Idle mode, the clock to the CPU is shut down while peripherals keep running. Unlike Sleep mode, the clock source remains active.

Several peripherals have a control bit in each module, that allows them to operate during Idle.

LPRC Fail-Safe Clock Monitor remains active if clock failure detect is enabled.

The processor wakes up from Idle if at least one of the following conditions is true:

- on any interrupt that is individually enabled (IE bit is '1') and meets the required priority level
- on any Reset (POR, BOR, $\overline{\text{MCLR}}$)
- on WDT time-out

Upon wake-up from Idle mode, the clock is re-applied to the CPU and instruction execution begins immediately, starting with the instruction following the PWRSAV instruction.

Any interrupt that is individually enabled (using IE bit) and meets the prevailing priority level can wake-up the processor. The processor processes the interrupt and branches to the ISR. The IDLE status bit in RCON register is set upon wake-up.

Any Reset, other than POR, sets the IDLE status bit. On a POR, the IDLE bit is cleared.

If Watchdog Timer is enabled, then the processor wakes-up from Idle mode upon WDT time-out. The IDLE and WDTO status bits are both set.

Unlike wake-up from Sleep, there are no time delays involved in wake-up from Idle.

21.6 Device Configuration Registers

The Configuration bits in each device Configuration register specify some of the device modes and are programmed by a device programmer, or by using the In-Circuit Serial Programming™ (ICSP™) feature of the device. Each device Configuration register is a 24-bit register, but only the lower 16 bits of each register are used to hold configuration data. There are four device Configuration registers available to the user:

1. FOSC (0xF80000): Oscillator Configuration Register
2. FWDT (0xF80002): Watchdog Timer Configuration Register
3. FBORPOR (0xF80004): BOR and POR Configuration Register
4. FGS (0xF8000A): General Code Segment Configuration Register

The placement of the Configuration bits is automatically handled when you select the device in your device programmer. The desired state of the Configuration bits may be specified in the source code (dependent on the language tool used), or through the programming interface. After the device has been programmed, the application software may read the Configuration bit values through the table read instructions. For additional information, please refer to the programming specifications of the device.

Note: If the code protection Configuration fuse bits (FGS<GCP> and FGS<GWRP>) have been programmed, an erase of the entire code-protected device is only possible at voltages $V_{DD} \geq 4.5V$.

21.7 In-Circuit Debugger

When MPLAB® ICD 2 is selected as a debugger, the in-circuit debugging functionality is enabled. This function allows simple debugging functions when used with MPLAB IDE. When the device has this feature enabled, some of the resources are not available for general use. These resources include the first 80 bytes of data RAM and two I/O pins.

one of four pairs of debug I/O pins may be selected by the user using configuration options in MPLAB IDE. These pin pairs are named EMUD/EMUC, EMUD1/EMUC1, EMUD2/EMUC2 and EMUD3/EMUC3.

In each case, the selected EMUD pin is the emulation/debug data line and the EMUC pin is the emulation/debug clock line. These pins interface to the MPLAB ICD 2 module available from Microchip. The selected pair of debug I/O pins is used by MPLAB ICD 2 to send commands and receive responses, as well as to send and receive data. To use the in-circuit debugger function of the device, the design must implement ICSP connections to MCLR, VDD, VSS, PGC, PGD and the selected EMUDx/EMUCx pin pair.

This gives rise to two possibilities:

1. If EMUD/EMUC is selected as the debug I/O pin pair, then only a 5-pin interface is required as the EMUD and EMUC pin functions are multiplexed with the PGD and PGC pin functions in all dsPIC30F devices.
2. If EMUD1/EMUC1, EMUD2/EMUC2 or EMUD3/EMUC3 is selected as the debug I/O pin pair, then a 7-pin interface is required as the EMUDx/EMUCx pin functions (x = 1, 2 or 3) are not multiplexed with the PGD and PGC pin functions.

dsPIC30F4011/4012

TABLE 21-7: SYSTEM INTEGRATION REGISTER MAP

| SFR Name | Addr. | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset State |
|----------|-------|--------|--------|-----------|--------|--------|--------|-----------|-------|-----------|-----------|--------|-------|-------|-------|---------|-------|--------------------------------|
| RCON | 0740 | TRAPR | IOPUWR | BGST | — | TUN1 | TUN0 | NOSC<1:0> | EXTR | SWR | POST<1:0> | SWDTEN | WDTO | SLEEP | IDLE | BOR | POR | Depends on type of Reset. |
| OSCCON | 0742 | TUN3 | TUN2 | COSC<1:0> | — | TUN1 | TUN0 | NOSC<1:0> | — | POST<1:0> | — | LOCK | — | CF | — | LPOSCEN | OSWEN | Depends on Configuration bits. |

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

TABLE 21-8: DEVICE CONFIGURATION REGISTER MAP

| File Name | Addr. | Bits 23-16 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
|-----------|--------|------------|--------|--------|--------|--------|--------|--------|-----------|-------|-------|-------|------------|-------|-------|-------|-------|------------|------|
| FOSC | F80000 | — | — | — | — | — | — | — | FOSC<1:0> | — | — | — | — | — | — | — | — | FPR<3:0> | |
| FWDT | F80002 | — | — | — | — | — | — | — | — | — | — | — | FWPSA<1:0> | — | — | — | — | FWPSB<3:0> | |
| FBORPOR | F80004 | — | — | — | — | — | — | — | HPOL | LPOL | BOREN | — | BORV<1:0> | — | — | — | — | FPWRT<1:0> | |
| FGS | F8000A | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | — | GCP | GWRP |

Note: Refer to the "dsPIC30F Family Reference Manual" (DS70046) for descriptions of register bit fields.

22.0 INSTRUCTION SET SUMMARY

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the *dsPIC30F Family Reference Manual* (DS70046). For more information on the device instruction set and programming, refer to the *dsPIC30F/33F Programmer's Reference Manual* (DS70157).

The dsPIC30F instruction set adds many enhancements to the previous PIC[®] MCU instruction sets, while maintaining an easy migration from PIC MCU instruction sets.

Most instructions are a single program memory word (24 bits). Only three instructions require two program memory locations.

Each single-word instruction is a 24-bit word divided into an 8-bit 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 five basic categories:

- Word or byte-oriented operations
- Bit-oriented operations
- Literal operations
- DSP operations
- Control operations

Table 22-1 shows the general symbols used in describing the instructions.

The dsPIC30F instruction set summary in Table 22-2 lists all the instructions along with the status flags affected by each instruction.

Most word or byte-oriented W register instructions (including barrel shift instructions) have three operands:

- The first source operand, which is typically a register 'Wb' without any address modifier
- The second source operand, which is typically a register 'Ws' with or without an address modifier
- The destination of the result, which is typically a register 'Wd' with or without an address modifier

However, word or byte-oriented file register instructions have two operands:

- The file register specified by the value 'f'
- The destination, which could either be the file register 'f' or the W0 register, which is denoted as 'WREG'

Most bit-oriented instructions (including simple rotate/shift instructions) have two operands:

- The W register (with or without an address modifier) or file register (specified by the value of 'Ws' or 'f')
- The bit in the W register or file register (specified by a literal value or indirectly by the contents of register 'Wb')

The literal instructions that involve data movement may use some of the following operands:

- A literal value to be loaded into a W register or file register (specified by the value of 'k')
- The W register or file register where the literal value is to be loaded (specified by 'Wb' or 'f')

However, literal instructions that involve arithmetic or logical operations use some of the following operands:

- The first source operand which is a register 'Wb' without any address modifier
- The second source operand which is a literal value
- The destination of the result (only if not the same as the first source operand) which is typically a register 'Wd' with or without an address modifier

The MAC class of DSP instructions may use some of the following operands:

- The accumulator (A or B) to be used (required operand)
- The W registers to be used as the two operands
- The X and Y address space prefetch operations
- The X and Y address space prefetch destinations
- The accumulator write-back destination

The other DSP instructions do not involve any multiplication and may include:

- The accumulator to be used (required)
- The source or destination operand (designated as Wso or Wdo, respectively) with or without an address modifier
- The amount of shift, specified by a W register 'Wn' or a literal value

The control instructions may use some of the following operands:

- A program memory address
- The mode of the table read and table write instructions

All instructions are a single word, except for certain double-word instructions, which were made double-word instructions so that all the required information is available in these 48 bits. In the second word, the 8 MSBs are '0's. If this second word is executed as an instruction (by itself), it will execute as a NOP.

dsPIC30F4011/4012

Most 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. Notable exceptions are the BRA (unconditional/computed branch), indirect CALL/GOTO, all table reads and writes and RETURN/RETFIE instructions, which are single-word instructions but take two or three cycles. Certain instructions that involve skipping over the subsequent instruction, require either two or three

cycles if the skip is performed, depending on whether the instruction being skipped is a single-word or two-word instruction. Moreover, double-word moves require two cycles. The double-word instructions execute in two instruction cycles.

Note: For more details on the instruction set, refer to the “dsPIC30F/33F Programmer’s Reference Manual” (DS70157).

TABLE 22-1: SYMBOLS USED IN OPCODE DESCRIPTIONS

| Field | Description |
|-----------------|---|
| #text | Means literal defined by “text” |
| (text) | Means “content of text” |
| [text] | Means “the location addressed by text” |
| { } | Optional field or operation |
| <n:m> | Register bit field |
| .b | Byte mode selection |
| .d | Double-Word mode selection |
| .s | Shadow register select |
| .w | Word mode selection (default) |
| Acc | One of two accumulators {A, B} |
| AWB | Accumulator Write-Back Destination Address register $\in \{W13, [W13] \pm 2\}$ |
| bit4 | 4-bit bit selection field (used in word addressed instructions) $\in \{0 \dots 15\}$ |
| C, DC, N, OV, Z | MCU Status bits: Carry, Digit Carry, Negative, Overflow, Zero |
| Expr | Absolute address, label or expression (resolved by the linker) |
| f | File register address $\in \{0x0000 \dots 0x1FFF\}$ |
| lit1 | 1-bit unsigned literal $\in \{0, 1\}$ |
| lit4 | 4-bit unsigned literal $\in \{0 \dots 15\}$ |
| lit5 | 5-bit unsigned literal $\in \{0 \dots 31\}$ |
| lit8 | 8-bit unsigned literal $\in \{0 \dots 255\}$ |
| lit10 | 10-bit unsigned literal $\in \{0 \dots 255\}$ for Byte mode, $\{0:1023\}$ for Word mode |
| lit14 | 14-bit unsigned literal $\in \{0 \dots 16384\}$ |
| lit16 | 16-bit unsigned literal $\in \{0 \dots 65535\}$ |
| lit23 | 23-bit unsigned literal $\in \{0 \dots 8388608\}$; LSB must be 0 |
| None | Field does not require an entry, may be blank |
| OA, OB, SA, SB | DSP Status bits: ACCA Overflow, ACCB Overflow, ACCA Saturate, ACCB Saturate |
| PC | Program Counter |
| Slit10 | 10-bit signed literal $\in \{-512 \dots 511\}$ |
| Slit16 | 16-bit signed literal $\in \{-32768 \dots 32767\}$ |
| Slit6 | 6-bit signed literal $\in \{-16 \dots 16\}$ |

dsPIC30F4011/4012

TABLE 22-1: SYMBOLS USED IN OPCODE DESCRIPTIONS (CONTINUED)

| Field | Description |
|--------|---|
| wb | Base W register $\in \{W0..W15\}$ |
| wd | Destination W register $\in \{Wd, [Wd], [Wd++] , [Wd--], [++Wd], [--Wd]\}$ |
| wdo | Destination W register $\in \{Wnd, [Wnd], [Wnd++] , [Wnd--], [++Wnd], [--Wnd], [Wnd+Wb]\}$ |
| wm, wn | Dividend, Divisor working register pair (Direct Addressing) |
| wm*wm | Multiplicand and Multiplier working register pair for Square instructions $\in \{W4*W4, W5*W5, W6*W6, W7*W7\}$ |
| wm*wn | Multiplicand and Multiplier working register pair for DSP instructions $\in \{W4*W5, W4*W6, W4*W7, W5*W6, W5*W7, W6*W7\}$ |
| wn | One of 16 working registers $\in \{W0..W15\}$ |
| wnd | One of 16 destination working registers $\in \{W0..W15\}$ |
| wns | One of 16 source working registers $\in \{W0..W15\}$ |
| WREG | W0 (working register used in file register instructions) |
| ws | Source W register $\in \{Ws, [Ws], [Ws++] , [Ws--], [++Ws], [--Ws]\}$ |
| wso | Source W register $\in \{Wns, [Wns], [Wns++] , [Wns--], [++Wns], [--Wns], [Wns+Wb]\}$ |
| wx | X Data Space Prefetch Address register for DSP instructions $\in \{[W8] += 6, [W8] += 4, [W8] += 2, [W8], [W8] -= 6, [W8] -= 4, [W8] -= 2, [W9] += 6, [W9] += 4, [W9] += 2, [W9], [W9] -= 6, [W9] -= 4, [W9] -= 2, [W9+W12], \text{none}\}$ |
| wxd | X Data Space Prefetch Destination register for DSP instructions $\in \{W4..W7\}$ |
| wy | Y Data Space Prefetch Address register for DSP instructions $\in \{[W10] += 6, [W10] += 4, [W10] += 2, [W10], [W10] -= 6, [W10] -= 4, [W10] -= 2, [W11] += 6, [W11] += 4, [W11] += 2, [W11], [W11] -= 6, [W11] -= 4, [W11] -= 2, [W11+W12], \text{none}\}$ |
| wyd | Y Data Space Prefetch Destination register for DSP instructions $\in \{W4..W7\}$ |

dsPIC30F4011/4012

TABLE 22-2: INSTRUCTION SET OVERVIEW

| Base Instr # | Assembly Mnemonic | Assembly Syntax | Description | # of words | # of cycles | Status Flags Affected |
|--------------------|------------------------|-----------------------------|---|------------|---------------|-----------------------|
| 1 | ADD | ADD <i>Acc</i> | Add Accumulators | 1 | 1 | OA, OB, SA, SB |
| | | ADD <i>f</i> | $f = f + \text{WREG}$ | 1 | 1 | C, DC, N, OV, Z |
| | | ADD <i>f, WREG</i> | $\text{WREG} = f + \text{WREG}$ | 1 | 1 | C, DC, N, OV, Z |
| | | ADD <i>#lit10, Wn</i> | $\text{Wd} = \text{lit10} + \text{Wd}$ | 1 | 1 | C, DC, N, OV, Z |
| | | ADD <i>Wb, Ws, Wd</i> | $\text{Wd} = \text{Wb} + \text{Ws}$ | 1 | 1 | C, DC, N, OV, Z |
| | | ADD <i>Wb, #lit5, Wd</i> | $\text{Wd} = \text{Wb} + \text{lit5}$ | 1 | 1 | C, DC, N, OV, Z |
| 2 | ADDC | ADDC <i>f</i> | $f = f + \text{WREG} + (C)$ | 1 | 1 | C, DC, N, OV, Z |
| | | ADDC <i>f, WREG</i> | $\text{WREG} = f + \text{WREG} + (C)$ | 1 | 1 | C, DC, N, OV, Z |
| | | ADDC <i>#lit10, Wn</i> | $\text{Wd} = \text{lit10} + \text{Wd} + (C)$ | 1 | 1 | C, DC, N, OV, Z |
| | | ADDC <i>Wb, Ws, Wd</i> | $\text{Wd} = \text{Wb} + \text{Ws} + (C)$ | 1 | 1 | C, DC, N, OV, Z |
| | | ADDC <i>Wb, #lit5, Wd</i> | $\text{Wd} = \text{Wb} + \text{lit5} + (C)$ | 1 | 1 | C, DC, N, OV, Z |
| 3 | AND | AND <i>f</i> | $f = f \text{ .AND. } \text{WREG}$ | 1 | 1 | N, Z |
| | | AND <i>f, WREG</i> | $\text{WREG} = f \text{ .AND. } \text{WREG}$ | 1 | 1 | N, Z |
| | | AND <i>#lit10, Wn</i> | $\text{Wd} = \text{lit10} \text{ .AND. } \text{Wd}$ | 1 | 1 | N, Z |
| | | AND <i>Wb, Ws, Wd</i> | $\text{Wd} = \text{Wb} \text{ .AND. } \text{Ws}$ | 1 | 1 | N, Z |
| | | AND <i>Wb, #lit5, Wd</i> | $\text{Wd} = \text{Wb} \text{ .AND. } \text{lit5}$ | 1 | 1 | N, Z |
| 4 | ASR | ASR <i>f</i> | $f = \text{Arithmetic Right Shift } f$ | 1 | 1 | C, N, OV, Z |
| | | ASR <i>f, WREG</i> | $\text{WREG} = \text{Arithmetic Right Shift } f$ | 1 | 1 | C, N, OV, Z |
| | | ASR <i>Ws, Wd</i> | $\text{Wd} = \text{Arithmetic Right Shift } \text{Ws}$ | 1 | 1 | C, N, OV, Z |
| | | ASR <i>Wb, Wns, Wnd</i> | $\text{Wnd} = \text{Arithmetic Right Shift } \text{Wb} \text{ by } \text{Wns}$ | 1 | 1 | N, Z |
| | | ASR <i>Wb, #lit5, Wnd</i> | $\text{Wnd} = \text{Arithmetic Right Shift } \text{Wb} \text{ by } \text{lit5}$ | 1 | 1 | N, Z |
| 5 | BCLR | BCLR <i>f, #bit4</i> | Bit Clear <i>f</i> | 1 | 1 | None |
| | | BCLR <i>Ws, #bit4</i> | Bit Clear <i>Ws</i> | 1 | 1 | None |
| 6 | BRA | BRA <i>C, Expr</i> | Branch if Carry | 1 | 1 (2) | None |
| | | BRA <i>GE, Expr</i> | Branch if greater than or equal | 1 | 1 (2) | None |
| | | BRA <i>GEU, Expr</i> | Branch if unsigned greater than or equal | 1 | 1 (2) | None |
| | | BRA <i>GT, Expr</i> | Branch if greater than | 1 | 1 (2) | None |
| | | BRA <i>GTU, Expr</i> | Branch if unsigned greater than | 1 | 1 (2) | None |
| | | BRA <i>LE, Expr</i> | Branch if less than or equal | 1 | 1 (2) | None |
| | | BRA <i>LEU, Expr</i> | Branch if unsigned less than or equal | 1 | 1 (2) | None |
| | | BRA <i>LT, Expr</i> | Branch if less than | 1 | 1 (2) | None |
| | | BRA <i>LTU, Expr</i> | Branch if unsigned less than | 1 | 1 (2) | None |
| | | BRA <i>N, Expr</i> | Branch if Negative | 1 | 1 (2) | None |
| | | BRA <i>NC, Expr</i> | Branch if Not Carry | 1 | 1 (2) | None |
| | | BRA <i>NN, Expr</i> | Branch if Not Negative | 1 | 1 (2) | None |
| | | BRA <i>NOV, Expr</i> | Branch if Not Overflow | 1 | 1 (2) | None |
| | | BRA <i>NZ, Expr</i> | Branch if Not Zero | 1 | 1 (2) | None |
| | | BRA <i>OA, Expr</i> | Branch if Accumulator A overflow | 1 | 1 (2) | None |
| | | BRA <i>OB, Expr</i> | Branch if Accumulator B overflow | 1 | 1 (2) | None |
| | | BRA <i>OV, Expr</i> | Branch if Overflow | 1 | 1 (2) | None |
| | | BRA <i>SA, Expr</i> | Branch if Accumulator A saturated | 1 | 1 (2) | None |
| | | BRA <i>SB, Expr</i> | Branch if Accumulator B saturated | 1 | 1 (2) | None |
| BRA <i>Expr</i> | Branch Unconditionally | 1 | 2 | None | | |
| BRA <i>Z, Expr</i> | Branch if Zero | 1 | 1 (2) | None | | |
| BRA <i>Wn</i> | Computed Branch | 1 | 2 | None | | |
| 7 | BSET | BSET <i>f, #bit4</i> | Bit Set <i>f</i> | 1 | 1 | None |
| | | BSET <i>Ws, #bit4</i> | Bit Set <i>Ws</i> | 1 | 1 | None |
| 8 | BSW | BSW. <i>C</i> <i>Ws, Wb</i> | Write <i>C</i> bit to <i>Ws</i> < <i>Wb</i> > | 1 | 1 | None |
| | | BSW. <i>Z</i> <i>Ws, Wb</i> | Write <i>Z</i> bit to <i>Ws</i> < <i>Wb</i> > | 1 | 1 | None |
| 9 | BTG | BTG <i>f, #bit4</i> | Bit Toggle <i>f</i> | 1 | 1 | None |
| | | BTG <i>Ws, #bit4</i> | Bit Toggle <i>Ws</i> | 1 | 1 | None |
| 10 | BTSC | BTSC <i>f, #bit4</i> | Bit Test <i>f</i> , Skip if Clear | 1 | 1 (2 or 3) | None |
| | | BTSC <i>Ws, #bit4</i> | Bit Test <i>Ws</i> , Skip if Clear | 1 | 1 (2 or 3) | None |

dsPIC30F4011/4012

TABLE 22-2: INSTRUCTION SET OVERVIEW (CONTINUED)

| Base Instr # | Assembly Mnemonic | Assembly Syntax | Description | # of words | # of cycles | Status Flags Affected |
|--------------|-------------------|----------------------------------|--|------------|---------------|--------------------------|
| 11 | BTSS | BTSS $f, \#bit4$ | Bit Test f , Skip if Set | 1 | 1 (2 or 3) | None |
| | | BTSS $Ws, \#bit4$ | Bit Test Ws , Skip if Set | 1 | 1 (2 or 3) | None |
| 12 | BTST | BTST $f, \#bit4$ | Bit Test f | 1 | 1 | Z |
| | | BTST.C $Ws, \#bit4$ | Bit Test Ws to C | 1 | 1 | C |
| | | BTST.Z $Ws, \#bit4$ | Bit Test Ws to Z | 1 | 1 | Z |
| | | BTST.C Ws, Wb | Bit Test $Ws < Wb >$ to C | 1 | 1 | C |
| | | BTST.Z Ws, Wb | Bit Test $Ws < Wb >$ to Z | 1 | 1 | Z |
| 13 | BTSTS | BTSTS $f, \#bit4$ | Bit Test then Set f | 1 | 1 | Z |
| | | BTSTS.C $Ws, \#bit4$ | Bit Test Ws to C, then Set | 1 | 1 | C |
| | | BTSTS.Z $Ws, \#bit4$ | Bit Test Ws to Z, then Set | 1 | 1 | Z |
| 14 | CALL | CALL $lit23$ | Call subroutine | 2 | 2 | None |
| | | CALL Wn | Call indirect subroutine | 1 | 2 | None |
| 15 | CLR | CLR f | $f = 0x0000$ | 1 | 1 | None |
| | | CLR $WREG$ | $WREG = 0x0000$ | 1 | 1 | None |
| | | CLR Ws | $Ws = 0x0000$ | 1 | 1 | None |
| | | CLR $Acc, Wx, Wxd, Wy, Wyd, AWB$ | Clear Accumulator | 1 | 1 | OA, OB, SA, SB |
| 16 | CLRWDT | CLRWDT | Clear Watchdog Timer | 1 | 1 | WDTO, Sleep |
| 17 | COM | COM f | $f = \bar{f}$ | 1 | 1 | N, Z |
| | | COM $f, WREG$ | $WREG = \bar{f}$ | 1 | 1 | N, Z |
| | | COM Ws, Wd | $Wd = \bar{Ws}$ | 1 | 1 | N, Z |
| 18 | CP | CP f | Compare f with $WREG$ | 1 | 1 | C, DC, N, OV, Z |
| | | CP $Wb, \#lit5$ | Compare Wb with $lit5$ | 1 | 1 | C, DC, N, OV, Z |
| | | CP Wb, Ws | Compare Wb with Ws ($Wb - Ws$) | 1 | 1 | C, DC, N, OV, Z |
| 19 | CPO | CPO f | Compare f with $0x0000$ | 1 | 1 | C, DC, N, OV, Z |
| | | CPO Ws | Compare Ws with $0x0000$ | 1 | 1 | C, DC, N, OV, Z |
| 20 | CPB | CPB f | Compare f with $WREG$, with Borrow | 1 | 1 | C, DC, N, OV, Z |
| | | CPB $Wb, \#lit5$ | Compare Wb with $lit5$, with Borrow | 1 | 1 | C, DC, N, OV, Z |
| | | CPB Wb, Ws | Compare Wb with Ws , with Borrow ($Wb - Ws - C$) | 1 | 1 | C, DC, N, OV, Z |
| 21 | CPSEQ | CPSEQ Wb, Wn | Compare Wb with Wn , skip if = | 1 | 1 (2 or 3) | None |
| 22 | CPSGT | CPSGT Wb, Wn | Compare Wb with Wn , skip if > | 1 | 1 (2 or 3) | None |
| 23 | CPSLT | CPSLT Wb, Wn | Compare Wb with Wn , skip if < | 1 | 1 (2 or 3) | None |
| 24 | CPSNE | CPSNE Wb, Wn | Compare Wb with Wn , skip if \neq | 1 | 1 (2 or 3) | None |
| 25 | DAW | DAW Wn | $Wn =$ decimal adjust Wn | 1 | 1 | C |
| 26 | DEC | DEC f | $f = f - 1$ | 1 | 1 | C, DC, N, OV, Z |
| | | DEC $f, WREG$ | $WREG = f - 1$ | 1 | 1 | C, DC, N, OV, Z |
| | | DEC Ws, Wd | $Wd = Ws - 1$ | 1 | 1 | C, DC, N, OV, Z |
| 27 | DEC2 | DEC2 f | $f = f - 2$ | 1 | 1 | C, DC, N, OV, Z |
| | | DEC2 $f, WREG$ | $WREG = f - 2$ | 1 | 1 | C, DC, N, OV, Z |
| | | DEC2 Ws, Wd | $Wd = Ws - 2$ | 1 | 1 | C, DC, N, OV, Z |
| 28 | DISI | DISI $\#lit14$ | Disable Interrupts for k instruction cycles | 1 | 1 | None |
| 29 | DIV | DIV.S Wm, Wn | Signed 16/16-bit Integer Divide | 1 | 18 | N, Z, C, OV |
| | | DIV.SD Wm, Wn | Signed 32/16-bit Integer Divide | 1 | 18 | N, Z, C, OV |
| | | DIV.U Wm, Wn | Unsigned 16/16-bit Integer Divide | 1 | 18 | N, Z, C, OV |
| | | DIV.UD Wm, Wn | Unsigned 32/16-bit Integer Divide | 1 | 18 | N, Z, C, OV |
| 30 | DIVF | DIVF Wm, Wn | Signed 16/16-bit Fractional Divide | 1 | 18 | N, Z, C, OV |
| 31 | DO | DO $\#lit14, Expr$ | Do code to PC + $Expr$, $lit14 + 1$ time | 2 | 2 | None |
| | | DO $Wn, Expr$ | Do code to PC + $Expr$, (Wn) + 1 time | 2 | 2 | None |
| 32 | ED | ED $Wm * Wm, Acc, Wx, Wy, Wxd$ | Euclidean Distance (no accumulate) | 1 | 1 | OA, OB, OAB, SA, SB, SAB |

dsPIC30F4011/4012

TABLE 22-2: INSTRUCTION SET OVERVIEW (CONTINUED)

| Base Instr # | Assembly Mnemonic | Assembly Syntax | Description | # of words | # of cycles | Status Flags Affected |
|---------------|--|---------------------------------------|--|------------|-------------|--------------------------|
| 33 | EDAC | EDAC Wm*Wm, Acc, Wx, Wy, Wxd | Euclidean Distance | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| 34 | EXCH | EXCH Wns, Wnd | Swap Wns with Wnd | 1 | 1 | None |
| 35 | FBCL | FBCL Ws, Wnd | Find Bit Change from Left (MSb) Side | 1 | 1 | C |
| 36 | FF1L | FF1L Ws, Wnd | Find First One from Left (MSb) Side | 1 | 1 | C |
| 37 | FF1R | FF1R Ws, Wnd | Find First One from Right (LSb) Side | 1 | 1 | C |
| 38 | GOTO | GOTO Expr | Go to address | 2 | 2 | None |
| | | GOTO Wn | Go to indirect | 1 | 2 | None |
| 39 | INC | INC f | f = f + 1 | 1 | 1 | C, DC, N, OV, Z |
| | | INC f, WREG | WREG = f + 1 | 1 | 1 | C, DC, N, OV, Z |
| | | INC Ws, Wd | Wd = Ws + 1 | 1 | 1 | C, DC, N, OV, Z |
| 40 | INC2 | INC2 f | f = f + 2 | 1 | 1 | C, DC, N, OV, Z |
| | | INC2 f, WREG | WREG = f + 2 | 1 | 1 | C, DC, N, OV, Z |
| | | INC2 Ws, Wd | Wd = Ws + 2 | 1 | 1 | C, DC, N, OV, Z |
| 41 | IOR | IOR f | f = f .IOR. WREG | 1 | 1 | N, Z |
| | | IOR f, WREG | WREG = f .IOR. WREG | 1 | 1 | N, Z |
| | | IOR #lit10, Wn | Wd = lit10 .IOR. Wd | 1 | 1 | N, Z |
| | | IOR Wb, Ws, Wd | Wd = Wb .IOR. Ws | 1 | 1 | N, Z |
| | | IOR Wb, #lit5, Wd | Wd = Wb .IOR. lit5 | 1 | 1 | N, Z |
| 42 | LAC | LAC Wso, #Slit4, Acc | Load Accumulator | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| 43 | LNK | LNK #lit14 | Link Frame Pointer | 1 | 1 | None |
| 44 | LSR | LSR f | f = Logical Right Shift f | 1 | 1 | C, N, OV, Z |
| | | LSR f, WREG | WREG = Logical Right Shift f | 1 | 1 | C, N, OV, Z |
| | | LSR Ws, Wd | Wd = Logical Right Shift Ws | 1 | 1 | C, N, OV, Z |
| | | LSR Wb, Wns, Wnd | Wnd = Logical Right Shift Wb by Wns | 1 | 1 | N, Z |
| | | LSR Wb, #lit5, Wnd | Wnd = Logical Right Shift Wb by lit5 | 1 | 1 | N, Z |
| 45 | MAC | MAC Wm*Wn, Acc, Wx, Wxd, Wy, Wyd, AWB | Multiply and Accumulate | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| | | MAC Wm*Wm, Acc, Wx, Wxd, Wy, Wyd | Square and Accumulate | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| 46 | MOV | MOV f, Wn | Move f to Wn | 1 | 1 | None |
| | | MOV f | Move f to f | 1 | 1 | N, Z |
| | | MOV f, WREG | Move f to WREG | 1 | 1 | N, Z |
| | | MOV #lit16, Wn | Move 16-bit literal to Wn | 1 | 1 | None |
| | | MOV.b #lit8, Wn | Move 8-bit literal to Wn | 1 | 1 | None |
| | | MOV Wn, f | Move Wn to f | 1 | 1 | None |
| | | MOV Wso, Wdo | Move Ws to Wd | 1 | 1 | None |
| | | MOV WREG, f | Move WREG to f | 1 | 1 | N, Z |
| | | MOV.D Wns, Wd | Move Double from W(ns):W(ns + 1) to Wd | 1 | 2 | None |
| MOV.D Ws, Wnd | Move Double from Ws to W(nd + 1):W(nd) | 1 | 2 | None | | |
| 47 | MOVSAC | MOVSAC Acc, Wx, Wxd, Wy, Wyd, AWB | Prefetch and store accumulator | 1 | 1 | None |
| 48 | MPY | MPY Wm*Wn, Acc, Wx, Wxd, Wy, Wyd | Multiply Wm by Wn to Accumulator | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| | | MPY Wm*Wm, Acc, Wx, Wxd, Wy, Wyd | Square Wm to Accumulator | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| 49 | MPY.N | MPY.N Wm*Wn, Acc, Wx, Wxd, Wy, Wyd | -(Multiply Wm by Wn) to Accumulator | 1 | 1 | None |
| 50 | MSC | MSC Wm*Wm, Acc, Wx, Wxd, Wy, Wyd, AWB | Multiply and Subtract from Accumulator | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| 51 | MUL | MUL.SS Wb, Ws, Wnd | {Wnd+1, Wnd} = signed(Wb) * signed(Ws) | 1 | 1 | None |
| | | MUL.SU Wb, Ws, Wnd | {Wnd+1, Wnd} = signed(Wb) * unsigned(Ws) | 1 | 1 | None |
| | | MUL.US Wb, Ws, Wnd | {Wnd+1, Wnd} = unsigned(Wb) * signed(Ws) | 1 | 1 | None |
| | | MUL.UU Wb, Ws, Wnd | {Wnd+1, Wnd} = unsigned(Wb) * unsigned(Ws) | 1 | 1 | None |
| | | MUL.SU Wb, #lit5, Wnd | {Wnd+1, Wnd} = signed(Wb) * unsigned(lit5) | 1 | 1 | None |
| | | MUL.UU Wb, #lit5, Wnd | {Wnd+1, Wnd} = unsigned(Wb) * unsigned(lit5) | 1 | 1 | None |
| | | MUL f | W3:W2 = f * WREG | 1 | 1 | None |

dsPIC30F4011/4012

TABLE 22-2: INSTRUCTION SET OVERVIEW (CONTINUED)

| Base Instr # | Assembly Mnemonic | Assembly Syntax | Description | # of words | # of cycles | Status Flags Affected |
|--------------|-------------------|-------------------------------|--|------------|-------------|--------------------------|
| 52 | NEG | NEG <i>Acc</i> | Negate Accumulator | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| | | NEG <i>f</i> | $f = \bar{f} + 1$ | 1 | 1 | C, DC, N, OV, Z |
| | | NEG <i>f, WREG</i> | $WREG = \bar{f} + 1$ | 1 | 1 | C, DC, N, OV, Z |
| | | NEG <i>Ws, Wd</i> | $Wd = \bar{Ws} + 1$ | 1 | 1 | C, DC, N, OV, Z |
| 53 | NOP | NOP | No Operation | 1 | 1 | None |
| | | NOPR | No Operation | 1 | 1 | None |
| 54 | POP | POP <i>f</i> | Pop <i>f</i> from Top-of-Stack (TOS) | 1 | 1 | None |
| | | POP <i>Wdo</i> | Pop from Top-of-Stack (TOS) to <i>Wdo</i> | 1 | 1 | None |
| | | POP.D <i>Wnd</i> | Pop from Top-of-Stack (TOS) to $W(nd):W(nd+1)$ | 1 | 2 | None |
| | | POP.S | Pop Shadow Registers | 1 | 1 | All |
| 55 | PUSH | PUSH <i>f</i> | Push <i>f</i> to Top-of-Stack (TOS) | 1 | 1 | None |
| | | PUSH <i>Wso</i> | Push <i>Wso</i> to Top-of-Stack (TOS) | 1 | 1 | None |
| | | PUSH.D <i>Wns</i> | Push $W(ns):W(ns+1)$ to Top-of-Stack (TOS) | 1 | 2 | None |
| | | PUSH.S | Push Shadow Registers | 1 | 1 | None |
| 56 | PWRSVAV | PWRSVAV #lit1 | Go into Sleep or Idle mode | 1 | 1 | WDTO, Sleep |
| 57 | RCALL | RCALL <i>Expr</i> | Relative Call | 1 | 2 | None |
| | | RCALL <i>Wn</i> | Computed Call | 1 | 2 | None |
| 58 | REPEAT | REPEAT #lit14 | Repeat Next Instruction lit14 + 1 time | 1 | 1 | None |
| | | REPEAT <i>Wn</i> | Repeat Next Instruction (<i>Wn</i>) + 1 time | 1 | 1 | None |
| 59 | RESET | RESET | Software device Reset | 1 | 1 | None |
| 60 | RETFIE | RETFIE | Return from interrupt | 1 | 3 (2) | None |
| 61 | RETLW | RETLW #lit10, <i>Wn</i> | Return with literal in <i>Wn</i> | 1 | 3 (2) | None |
| 62 | RETURN | RETURN | Return from Subroutine | 1 | 3 (2) | None |
| 63 | RLC | RLC <i>f</i> | <i>f</i> = Rotate Left through Carry <i>f</i> | 1 | 1 | C, N, Z |
| | | RLC <i>f, WREG</i> | $WREG = \text{Rotate Left through Carry } f$ | 1 | 1 | C, N, Z |
| | | RLC <i>Ws, Wd</i> | $Wd = \text{Rotate Left through Carry } Ws$ | 1 | 1 | C, N, Z |
| 64 | RLNC | RLNC <i>f</i> | <i>f</i> = Rotate Left (No Carry) <i>f</i> | 1 | 1 | N, Z |
| | | RLNC <i>f, WREG</i> | $WREG = \text{Rotate Left (No Carry) } f$ | 1 | 1 | N, Z |
| | | RLNC <i>Ws, Wd</i> | $Wd = \text{Rotate Left (No Carry) } Ws$ | 1 | 1 | N, Z |
| 65 | RRC | RRC <i>f</i> | <i>f</i> = Rotate Right through Carry <i>f</i> | 1 | 1 | C, N, Z |
| | | RRC <i>f, WREG</i> | $WREG = \text{Rotate Right through Carry } f$ | 1 | 1 | C, N, Z |
| | | RRC <i>Ws, Wd</i> | $Wd = \text{Rotate Right through Carry } Ws$ | 1 | 1 | C, N, Z |
| 66 | RRNC | RRNC <i>f</i> | <i>f</i> = Rotate Right (No Carry) <i>f</i> | 1 | 1 | N, Z |
| | | RRNC <i>f, WREG</i> | $WREG = \text{Rotate Right (No Carry) } f$ | 1 | 1 | N, Z |
| | | RRNC <i>Ws, Wd</i> | $Wd = \text{Rotate Right (No Carry) } Ws$ | 1 | 1 | N, Z |
| 67 | SAC | SAC <i>Acc, #Slit4, Wdo</i> | Store Accumulator | 1 | 1 | None |
| | | SAC.R <i>Acc, #Slit4, Wdo</i> | Store Rounded Accumulator | 1 | 1 | None |
| 68 | SE | SE <i>Ws, Wnd</i> | $Wnd = \text{sign-extended } Ws$ | 1 | 1 | C, N, Z |
| 69 | SETM | SETM <i>f</i> | $f = 0xFFFF$ | 1 | 1 | None |
| | | SETM <i>WREG</i> | $WREG = 0xFFFF$ | 1 | 1 | None |
| | | SETM <i>Ws</i> | $Ws = 0xFFFF$ | 1 | 1 | None |
| 70 | SFTAC | SFTAC <i>Acc, Wn</i> | Arithmetic Shift Accumulator by (<i>Wn</i>) | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| | | SFTAC <i>Acc, #Slit6</i> | Arithmetic Shift Accumulator by <i>Slit6</i> | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| 71 | SL | SL <i>f</i> | <i>f</i> = Left Shift <i>f</i> | 1 | 1 | C,N,OV,Z |
| | | SL <i>f, WREG</i> | $WREG = \text{Left Shift } f$ | 1 | 1 | C,N,OV,Z |
| | | SL <i>Ws, Wd</i> | $Wd = \text{Left Shift } Ws$ | 1 | 1 | C,N,OV,Z |
| | | SL <i>Wb, Wns, Wnd</i> | $Wnd = \text{Left Shift } Wb \text{ by } Wns$ | 1 | 1 | N, Z |
| | | SL <i>Wb, #lit5, Wnd</i> | $Wnd = \text{Left Shift } Wb \text{ by lit5}$ | 1 | 1 | N, Z |

dsPIC30F4011/4012

TABLE 22-2: INSTRUCTION SET OVERVIEW (CONTINUED)

| Base Instr # | Assembly Mnemonic | Assembly Syntax | Description | # of words | # of cycles | Status Flags Affected |
|--------------|-------------------|----------------------------|------------------------------------|------------|-------------|--------------------------|
| 72 | SUB | SUB <i>Acc</i> | Subtract Accumulators | 1 | 1 | OA, OB, OAB, SA, SB, SAB |
| | | SUB <i>f</i> | $f = f - WREG$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUB <i>f, WREG</i> | $WREG = f - WREG$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUB <i>#lit10, Wn</i> | $Wn = Wn - lit10$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUB <i>Wb, Ws, Wd</i> | $Wd = Wb - Ws$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUB <i>Wb, #lit5, Wd</i> | $Wd = Wb - lit5$ | 1 | 1 | C, DC, N, OV, Z |
| 73 | SUBB | SUBB <i>f</i> | $f = f - WREG - (\overline{C})$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBB <i>f, WREG</i> | $WREG = f - WREG - (\overline{C})$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBB <i>#lit10, Wn</i> | $Wn = Wn - lit10 - (\overline{C})$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBB <i>Wb, Ws, Wd</i> | $Wd = Wb - Ws - (\overline{C})$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBB <i>Wb, #lit5, Wd</i> | $Wd = Wb - lit5 - (\overline{C})$ | 1 | 1 | C, DC, N, OV, Z |
| 74 | SUBR | SUBR <i>f</i> | $f = WREG - f$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBR <i>f, WREG</i> | $WREG = WREG - f$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBR <i>Wb, Ws, Wd</i> | $Wd = Ws - Wb$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBR <i>Wb, #lit5, Wd</i> | $Wd = lit5 - Wb$ | 1 | 1 | C, DC, N, OV, Z |
| 75 | SUBBR | SUBBR <i>f</i> | $f = WREG - f - (\overline{C})$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBBR <i>f, WREG</i> | $WREG = WREG - f - (\overline{C})$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBBR <i>Wb, Ws, Wd</i> | $Wd = Ws - Wb - (\overline{C})$ | 1 | 1 | C, DC, N, OV, Z |
| | | SUBBR <i>Wb, #lit5, Wd</i> | $Wd = lit5 - Wb - (\overline{C})$ | 1 | 1 | C, DC, N, OV, Z |
| 76 | SWAP | SWAP.b <i>Wn</i> | $Wn = \text{nibble swap } Wn$ | 1 | 1 | None |
| | | SWAP <i>Wn</i> | $Wn = \text{byte swap } Wn$ | 1 | 1 | None |
| 77 | TBLRDH | TBLRDH <i>Ws, Wd</i> | Read Prog<23:16> to Wd<7:0> | 1 | 2 | None |
| 78 | TBLRDL | TBLRDL <i>Ws, Wd</i> | Read Prog<15:0> to Wd | 1 | 2 | None |
| 79 | TBLWTH | TBLWTH <i>Ws, Wd</i> | Write Ws<7:0> to Prog<23:16> | 1 | 2 | None |
| 80 | TBLWTL | TBLWTL <i>Ws, Wd</i> | Write Ws to Prog<15:0> | 1 | 2 | None |
| 81 | ULNK | ULNK | Unlink Frame Pointer | 1 | 1 | None |
| 82 | XOR | XOR <i>f</i> | $f = f .XOR. WREG$ | 1 | 1 | N, Z |
| | | XOR <i>f, WREG</i> | $WREG = f .XOR. WREG$ | 1 | 1 | N, Z |
| | | XOR <i>#lit10, Wn</i> | $Wd = lit10 .XOR. Wd$ | 1 | 1 | N, Z |
| | | XOR <i>Wb, Ws, Wd</i> | $Wd = Wb .XOR. Ws$ | 1 | 1 | N, Z |
| | | XOR <i>Wb, #lit5, Wd</i> | $Wd = Wb .XOR. lit5$ | 1 | 1 | N, Z |
| 83 | ZE | ZE <i>Ws, Wnd</i> | $Wnd = \text{Zero-Extend } Ws$ | 1 | 1 | C, N, Z |

23.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
 - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
 - MPASM™ Assembler
 - MPLAB C18 and MPLAB C30 C Compilers
 - MPLINK™ Object Linker/
MPLIB™ Object Librarian
 - MPLAB ASM30 Assembler/Linker/Library
- Simulators
 - MPLAB SIM Software Simulator
- Emulators
 - MPLAB ICE 2000 In-Circuit Emulator
 - MPLAB ICE 4000 In-Circuit Emulator
- In-Circuit Debugger
 - MPLAB ICD 2
- Device Programmers
 - PICSTART® Plus Development Programmer
 - MPLAB PM3 Device Programmer
 - PICKit™ 2 Development Programmer
- Low-Cost Demonstration and Development Boards and Evaluation Kits

23.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16-bit microcontroller market. The MPLAB IDE is a Windows® operating system-based application that contains:

- A single graphical interface to all debugging tools
 - Simulator
 - Programmer (sold separately)
 - Emulator (sold separately)
 - In-Circuit Debugger (sold separately)
- A full-featured editor with color-coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Visual device initializer for easy register initialization
- Mouse over variable inspection
- Drag and drop variables from source to watch windows
- Extensive on-line help
- Integration of select third party tools, such as HI-TECH Software C Compilers and IAR C Compilers

The MPLAB IDE allows you to:

- Edit your source files (either assembly or C)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
 - Source files (assembly or C)
 - Mixed assembly and C
 - Machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost-effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increased flexibility and power.

dsPIC30F4011/4012

23.2 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for all PIC MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

23.3 MPLAB C18 and MPLAB C30 C Compilers

The MPLAB C18 and MPLAB C30 Code Development Systems are complete ANSI C compilers for Microchip's PIC18 family of microcontrollers and the dsPIC30, dsPIC33 and PIC24 family of digital signal controllers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

23.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler and the MPLAB C18 C Compiler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

23.5 MPLAB ASM30 Assembler, Linker and Librarian

MPLAB ASM30 Assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 C Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

23.6 MPLAB SIM Software Simulator

The MPLAB SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC® DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB SIM Software Simulator fully supports symbolic debugging using the MPLAB C18 and MPLAB C30 C Compilers, and the MPASM and MPLAB ASM30 Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.