MICROCHIP

dsPIC30F
Data Sheet

High Performance

Digital Signal Controllers

Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products 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 PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be 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 product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, FilterLab, KEELOQ, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microID, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rfPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (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.

Printed on recycled paper.

DNV Certification, Inc.
USA

ANSI★RAB
QMS
ACCREDITED

DNV

DNV MSC
The Netherlands
Accredited by the RvA

ISO 9001 / QS-9000
REGISTERED FIRM

Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company’s quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip’s quality system for the design and manufacture of development systems is ISO 9001 certified.

dsPIC30F Enhanced FLASH 16-bit Microcontrollers

High Performance Modified RISC CPU:

- Modified Harvard architecture
- C compiler optimized instruction set architecture
- 89 base instructions
- 24-bit wide instructions, 16-bit wide data path
- Linear program memory addressing up to 4M Instruction Words
- Linear data memory addressing up to 64 Kbytes
- Up to 144 Kbytes on-chip FLASH program space
  - Up to 48K Instruction Words
- Up to 8 Kbytes of on-chip data RAM
- Up to 4 Kbytes of non-volatile data EEPROM
- 16 x 16-bit working register array
- Three Address Generation Units that enable:
  - Dual data fetch
  - Accumulator write back for DSP operations
- Flexible addressing modes supporting:
  - Indirect, Modulo and Bit-Reversed modes
- Two, 40-bit wide accumulators with optional saturation logic
- 16-bit x 16-bit single cycle hardware fractional/ integer multiplier
- Single cycle Multiply-Accumulate (MAC) operation
- 40-stage Barrel Shifter
- Up to 30 MIPS operation:
  - DC to 40 MHz external clock input
  - 4 MHz - 10 MHz oscillator input with PLL active (4x, 8x, 16x)
- Up to 45 interrupt sources
  - 8 user selectable priority levels
- Vector table with up to 62 vectors
  - 54 interrupt vectors
  - 8 processor exceptions and software traps

Peripheral Features:

- High current sink/source I/O pins: 25 mA/25 mA
- Up to 5 external interrupt sources
- Timer module with programmable prescaler:
  - Up to five 16-bit timers/counters; optionally pair up 16-bit timers into 32-bit timer modules
- 16-bit Capture Input functions
- 16-bit Compare/PWM Output functions
  - Dual Compare mode available
- Data Converter Interface (DCI), supports common audio CODEC protocols including I^{2}S and AC’97

Peripheral Features (Continued):

- 3-wire SPI™ modules (supports 4 frame modes)
- I^{2}C™ module supports Multi-Master/Slave mode and 7-bit/10-bit addressing
- Addressable UART modules supporting:
  - Interrupt on address bit
  - Wake-up on START bit
  - 4 characters deep TX and RX FIFO buffers
- CAN bus modules

Motor Control PWM Module Features:

- Up to 8 PWM output channels
  - Complementary or Independent Output modes
  - Edge and Center Aligned modes
- 4 duty cycle generators
- Dedicated time-base with 4 modes
- 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

Input Capture Module Features:

- Captures 16-bit timer value
  - Capture every 1st, 4th or 16th rising edge
  - Capture every falling edge
  - Capture every rising and falling edge
- Resolution of 33 ns at 30 MIPS
- Timer2 or Timer3 time-base selection
- Input Capture during IDLE
- Interrupt on input capture event

Analog Features:

- 12-bit or 10-bit Analog-to-Digital Converter (A/D) with:
  - 100 Ksps (for 12-bit A/D) or 500 Ksps (for 10-bit A/D) conversion rate
  - Up to 16 input channels
  - Conversion available during SLEEP and IDLE
- Programmable Low Voltage Detection (PLVD)
- Programmable Brown-out Detection and RESET generation

Special Microcontroller Features:

- Enhanced FLASH program memory
  - 100,000 erase/write cycle (typical) for industrial temperature range
- Data EEPROM memory
  - 1,000,000 erase/write cycle (typical) industrial temperature range
  - Data EEPROM Retention > 20 years
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- 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™) via 3 pins and power/ground
- 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

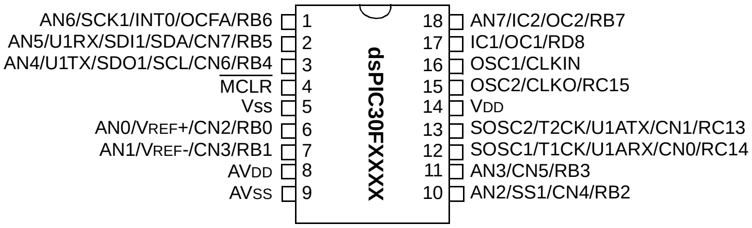
dsPIC30F Sensor Processor Family

Device	Pins	Program Memory		SRAM Bytes	EEPROM Bytes	Timer 16-bit	Input Cap	Output Comp/ Std PWM	A/D 12-bit 100 Ksps	UART	SPI™	I^{2}C™
		Bytes	Instructions									
dsPIC30F2011	18	12K	4K	1024	0	3	2	2	8 ch	1	1	1
dsPIC30F3012	18	24K	8K	2048	1024	3	2	2	8 ch	1	1	1
dsPIC30F2012	28	12K	4K	1024	0	3	2	2	10 ch	1	1	1
dsPIC30F3013	28	24K	8K	2048	1024	3	2	2	10 ch	2	1	1

Pin Diagrams

18-Pin SOIC and PDIP

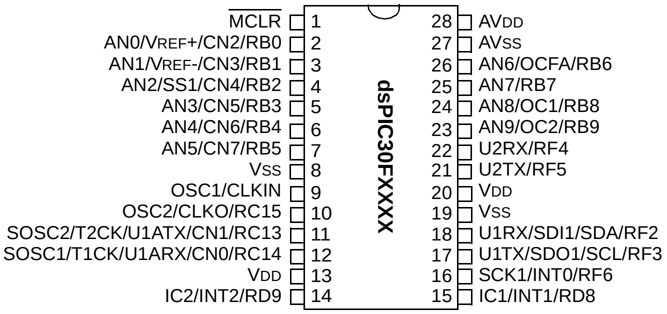
Part No.: 30F2011 / 30F3012



Note: Pinout subject to change.

28-Pin SDIP

Part No.: 30F2012 / 30F3013



Note: Pinout subject to change.

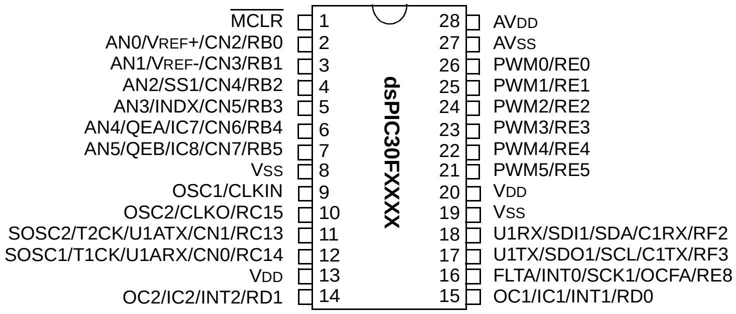
dsPIC30F Power Conversion and Motion Control Family

Device	Pins	Program Mem. Bytes/ Instructions	SRAM Bytes	EEPROM Bytes	Timer 16-bit	Input Cap	Output Comp/Std PWM	Motor Control PWM	A/D 10-bit 500 Ksps	Quad Enc	UART	SPI™	I^{2}C™	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
dsPIC30F4010	64	36K/12K	2048	1024	5	8	8	8 ch	16 ch	Yes	2	2	1	1
dsPIC30F5010	64	96K/32K	4096	2048	5	8	8	8 ch	16 ch	Yes	2	2	1	2
dsPIC30F6010	80	144K/48K	8192	4096	5	8	8	8 ch	16 ch	Yes	2	2	1	2

Pin Diagrams

28-Pin SDIP

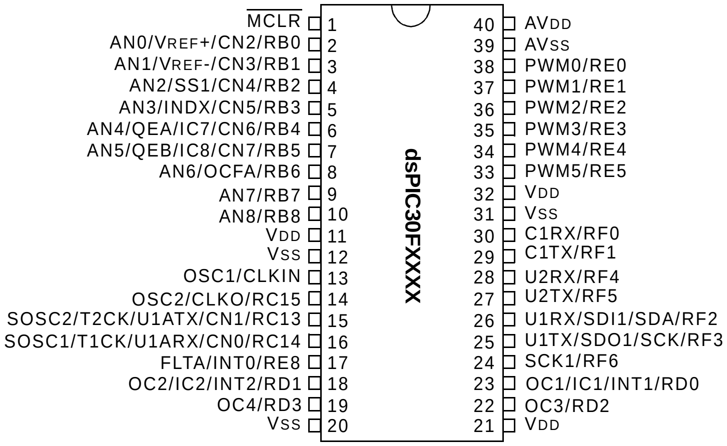
Part No.: 30F2010 / 30F3010 / 30F4012



Note: Pinout subject to change.

40-Pin PDIP

Part No.: 30F3011 / 30F4011

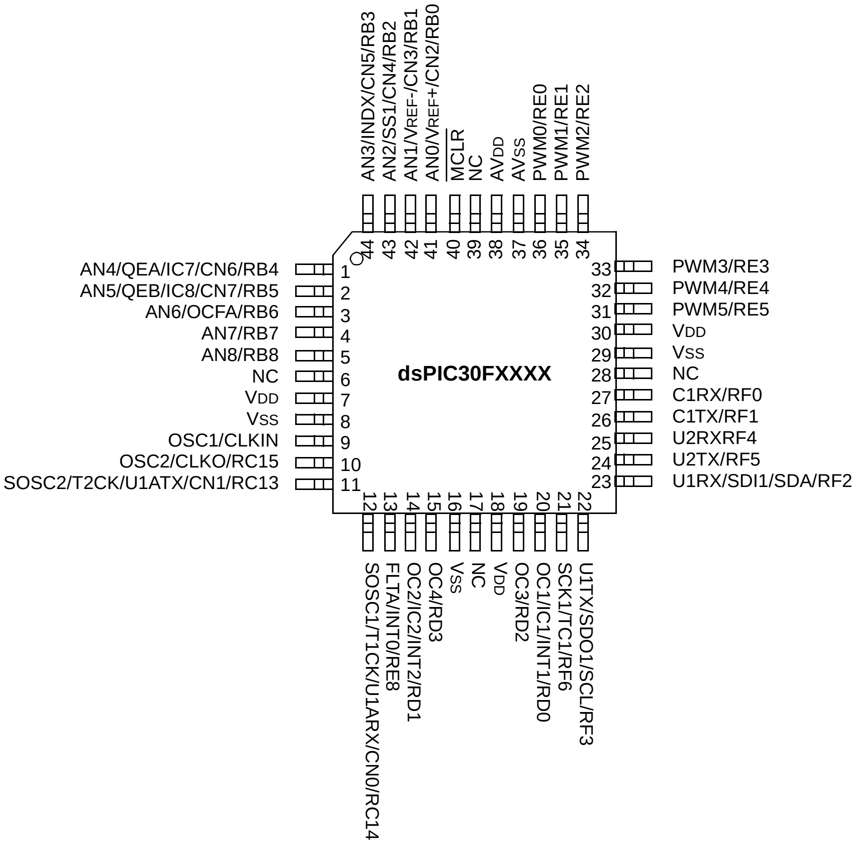


Note: Pinout subject to change.

Pin Diagrams (Cont.)

44-Pin TQFP

Part No.: 30F3011 / 30F4011

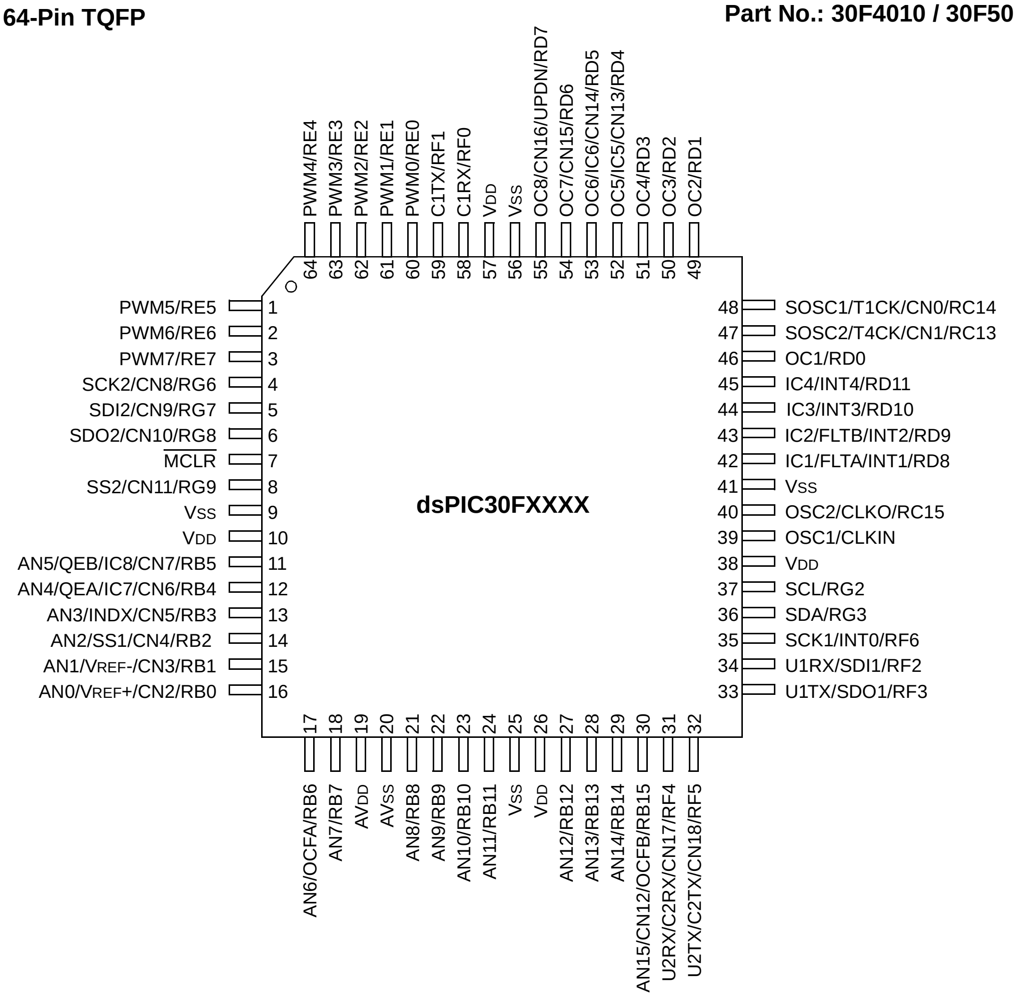


Note: Pinout subject to change.

Pin Diagrams (Cont.)

64-Pin TQFP

Part No.: 30F4010 / 30F5010

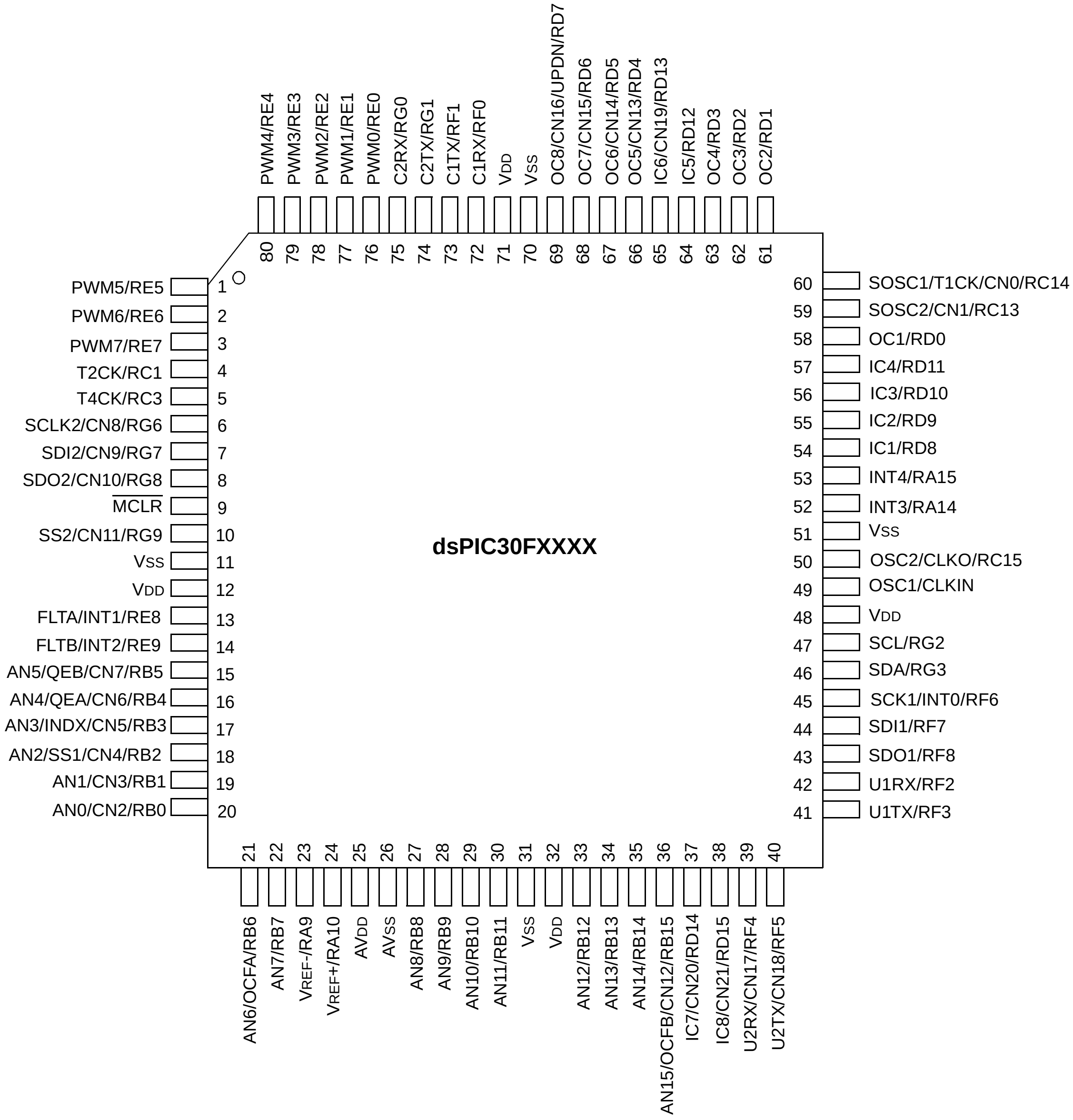


Note: Pinout subject to change.

Pin Diagrams (Cont.)

80-Pin TQFP

Part No.: 30F6010



Note: Pinout subject to change.

dsPIC30F General Purpose Controller Family

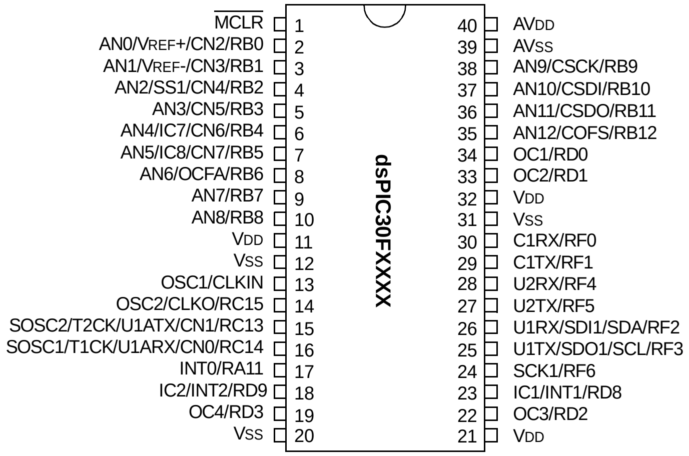
Device	Pins	Program Memory		SRAM Bytes	EEPROM Bytes	Timer 16-bit	Input Cap	Output Comp/Std PWM	CODEC Interface	A/D 12-bit 100 Ksps	UART	SPI™	I^{2}C™	CAN
		Bytes	Instructions											
dsPIC30F3014**	40/44	24K	8K	2048	1024	3	2	2	-	13 ch	2	1	1	-
dsPIC30F4013**	40/44	48K	16K	2048	1024	5	4	4	AC97, I^{2}S	13 ch	2	1	1	1
dsPIC30F4014**	64	36K	12K	2048	1024	5	8	8	AC97, I^{2}S	16 ch	2	2	1	1
dsPIC30F5011	64	66K	22K	4096	1024	5	8	8	-	16 ch	2	2	1	2
dsPIC30F5012	64	96K	32K	4096	2048	5	8	8	AC97, I^{2}S	16 ch	2	2	1	2
dsPIC30F6011	64	132K	44K	6144	2048	5	8	8	-	16 ch	2	2	1	2
dsPIC30F6012	64	144K	48K	8192	4096	5	8	8	AC97, I^{2}S	16 ch	2	2	1	2
dsPIC30F4015**	80	36K	12K	2048	1024	5	8	8	AC97, I^{2}S	16 ch	2	2	1	1
dsPIC30F5013	80	66K	22K	4096	1024	5	8	8	-	16 ch	2	2	1	2
dsPIC30F5014	80	96K	32K	4096	2048	5	8	8	AC97, I^{2}S	16 ch	2	2	1	2
dsPIC30F6013	80	132K	44K	6144	2048	5	8	8	-	16 ch	2	2	1	2
dsPIC30F6014	80	144K	48K	8192	4096	5	8	8	AC97, I^{2}S	16 ch	2	2	1	2

**Proposed Products (others are committed)

Pin Diagrams

40-Pin PDIP

Part No.: 30F3014 / 30F4013

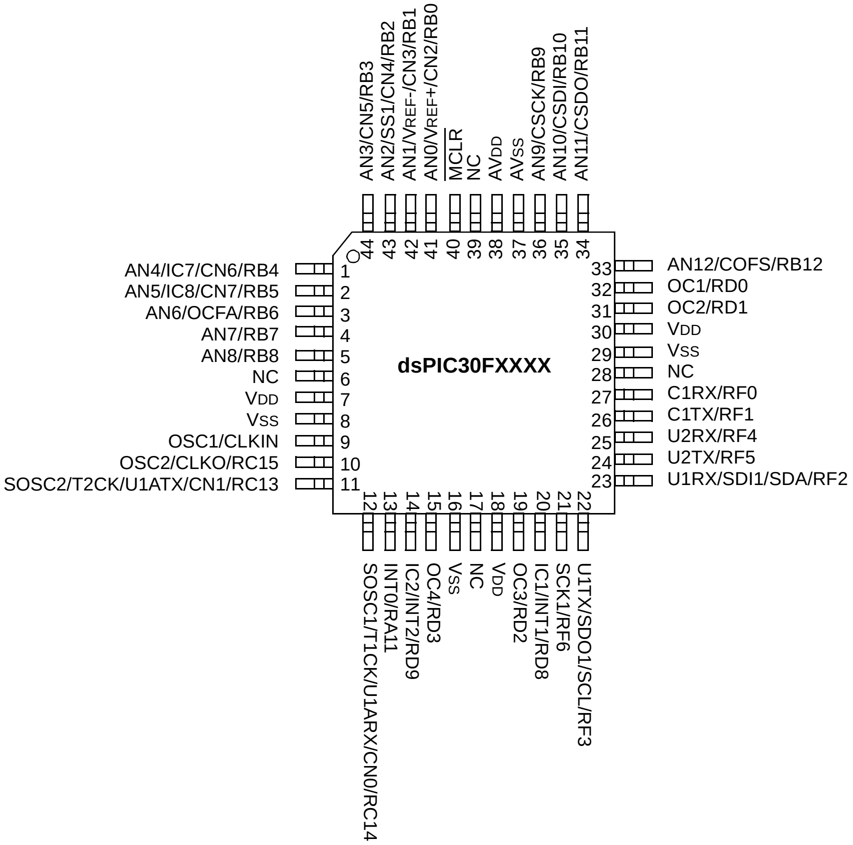


Note: Pinout subject to change.

Pin Diagrams (Cont.)

44-Pin TQFP

Part No.: 30F3014 / 30F4013

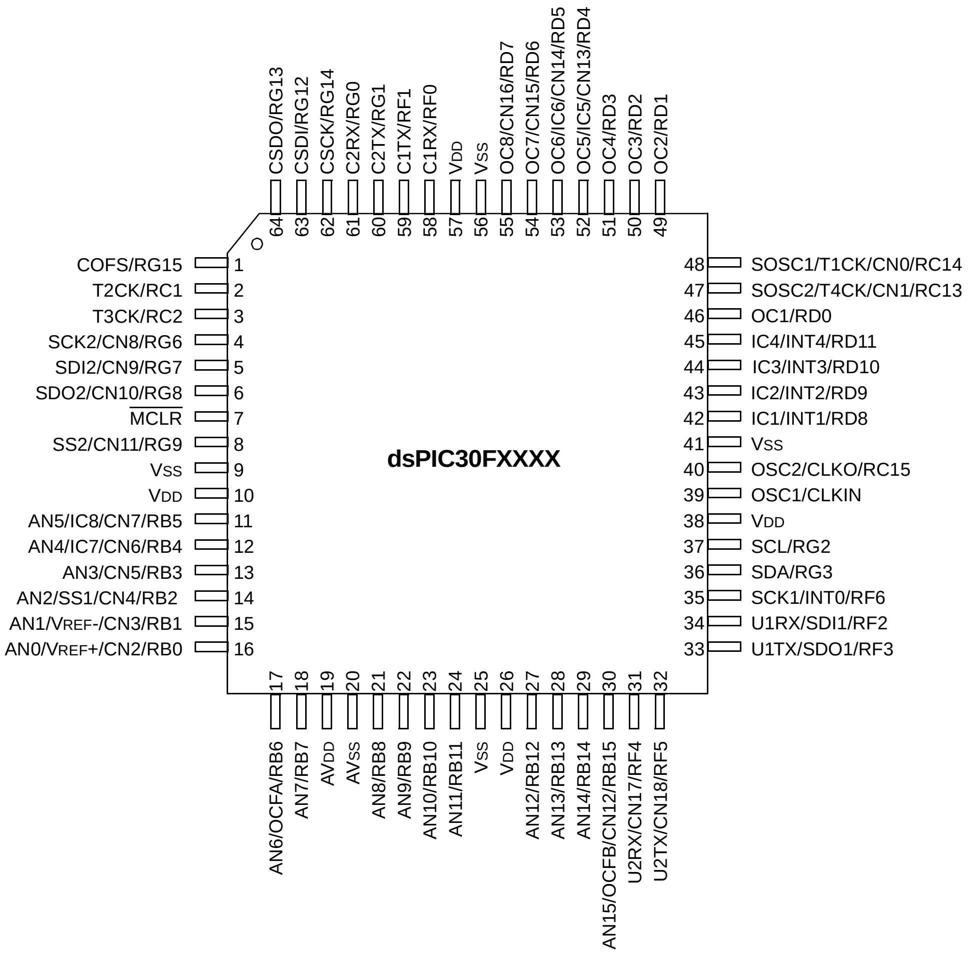


Note: Pinout subject to change.

Pin Diagrams (Cont.)

64-Pin TQFP

Part No.: 30F4014 / 30F5011 / 30F5012 / 30F6011 / 30F6012

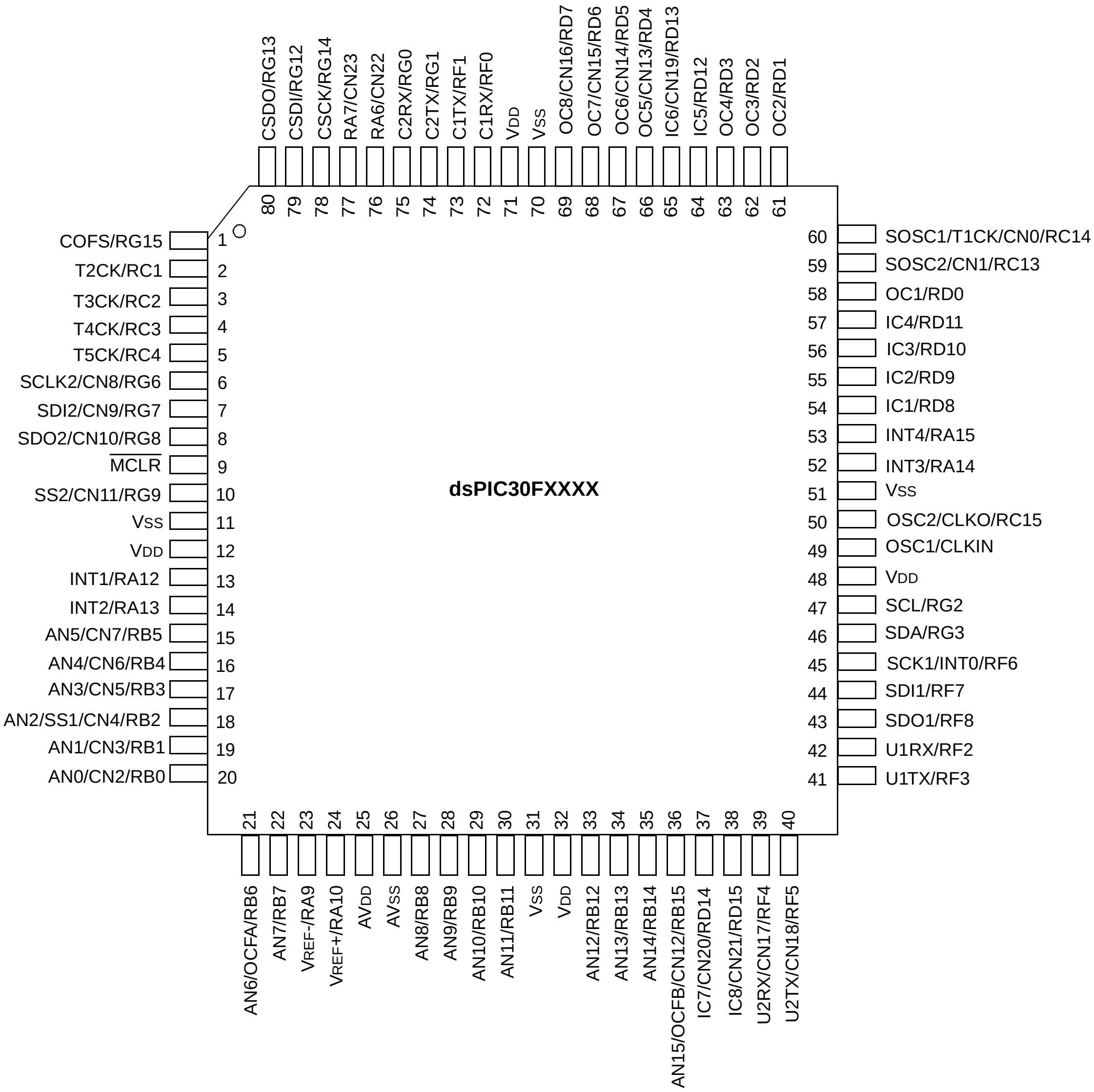


Note: Pinout subject to change.

Pin Diagrams (Cont.)

80-Pin TQFP

Part No.: 30F4015 / 30F5013 / 30F5014 / 30F6013 / 30F6014



Note: Pinout subject to change.

Table of Contents

TO OUR VALUED CUSTOMERS

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

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.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)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, 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/cn** to receive the most current information on all of our products.

NOTES:

1.0 DEVICE OVERVIEW

This document contains device family specific information for the dsPIC30F family of Digital Signal Controller (DSC) devices. The dsPIC30F devices contain extensive Digital Signal Processor (DSP) functionality within a high performance 16-bit Microcontroller (MCU) architecture.

Table 1-1 provides a brief description of device I/O pinouts and the functions that may be 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: PINOUT I/O DESCRIPTIONS

Pin Name	Pin Type	Buffer Type	Description
AN0 - AN15	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.
CLKIN	I	ST/CMOS	External clock source input. Always associated with OSC1 pin function.
CLKO	O	-	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKOUT in RC and EC modes. Always associated with OSC2 pin function.
CN0 - CN23	I	ST	Input change notification inputs. Can be software programmed for internal weak pull-ups on all inputs.
COFS	I/O	ST	Data Converter Interface frame synchronization pin.
CSCK	I/O	ST	Data Converter Interface serial clock input/output pin.
CSDI	I	ST	Data Converter Interface serial data input pin.
CSDO	O	-	Data Converter Interface serial data output pin.
C1RX	I	ST	CAN1 bus receive pin.
C1TX	O	-	CAN1 bus transmit pin.
C2RX	I	ST	CAN2 bus receive pin.
C2TX	O	-	CAN1 bus transmit pin
IC1 - IC8	I	ST	Capture inputs 1 through 8.
INDX	I	ST	Quadrature Encoder Index Pulse input.
QEA	I	ST	Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode.
QEB	I	ST	Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode.
UPDN	O	CMOS	Position Up/Down Counter Direction State.
INT0	I	ST	External Interrupt 0.
INT1	I	ST	External Interrupt 1.
INT2	I	ST	External Interrupt 2.
INT3	I	ST	External Interrupt 3.
INT4	I	ST	External Interrupt 4.

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

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

Pin Name	Pin Type	Buffer Type	Description
FLTA FLTB PWM0 PWM1 PWM2 PWM3 PWM4 PWM5 PWM6 PWM7	I I O O O O O O O O	ST ST - - - - - - - -	PWM Fault A input. PWM Fault B input. PWM output 0. PWM output 1. PWM output 2. PWM output 3. PWM output 4. PWM output 5. PWM output 6. PWM output 7.
$\overline{\text{MCLR}}$	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
OCFA OCFB OC1 OC2 OC3 OC4 OC5 OC6 OC7 OC8	I I O O O O O O O O	ST ST - - - - - - - -	Compare Fault A input (for Compare channels 1, 2, 3 and 4). Compare Fault B input (for Compare channels 5, 6, 7 and 8). Compare1 output. Compare2 output. Compare3 output. Compare4 output. Compare5 output. Compare6 output. Compare7 output. Compare8 output.
OSC1 OSC2	I I/O	ST/CMOS -	Oscillator crystal input. ST buffer when configured in RC mode; CMOS otherwise. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKOUT in RC and EC modes.
RA0 - RA7 RA9 - RA15	I/O I/O	ST ST	PORTA is a bi-directional I/O port.
RB0 - RB15	I/O	ST	PORTB is a bi-directional I/O port.
RC0 - RC4 RC13 - RC15	I/O I/O	ST ST	PORTC is a bi-directional I/O port.
RD0 - RD15	I/O	ST	PORTD is a bi-directional I/O port.
RE0 - RE9	I/O	ST	PORTE is a bi-directional I/O port.
RF0 - RF8 RF12 - RF13	I/O I/O	ST ST	PORTF is a bi-directional I/O port.
RG0 - RG3 RG6 - RG9 RG12 - RG15	I/O I/O I/O	ST ST ST	PORTG is a bi-directional I/O port.
SCK1 SDI1 SDO1 SS1 SCK2 SDI2 SDO2 SS2	I/O I O I I/O I O I	ST ST - ST ST ST - ST	Synchronous serial clock input/output for SPI1. SPI1 Data In. SPI1 Data Out. SPI1 Slave Synchronization. Synchronous serial clock input/output for SPI2. SPI2 Data In. SPI2 Data Out. SPI2 Slave Synchronization.
SCL SDA	I/O I/O	ST ST	Synchronous serial clock input/output for I^2C. Synchronous serial data input/output for I^2C.
SOSC1 SOSC2	I O	ST/CMOS -	32 kHz low power oscillator crystal input; CMOS otherwise. 32 kHz low power oscillator crystal output.

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

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

Pin Name	Pin Type	Buffer Type	Description
T1CK	I	ST	Timer1 external clock input.
T2CK	I	ST	Timer2 external clock input.
T3CK	I	ST	Timer3 external clock input.
T4CK	I	ST	Timer4 external clock input.
T5CK	I	ST	Timer5 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:	TTL	= TTL compatible input	CMOS	=	CMOS compatible input or output
	ST	= Schmitt Trigger input with CMOS levels	Analog	=	Analog input
	I	= Input	O	=	Output
	P	= Power	OD	=	Open Drain (no P diode to VDD)

NOTES:

2.0 CORE ARCHITECTURE OVERVIEW

2.1 Core Overview

The core has a 24-bit instruction word. The Program Counter (PC) is 24-bits wide with the Least Significant (LS) bit always clear (see Section 3.1), and the Most Significant (MS) bit 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 pre-fetch mechanism is used to help maintain throughput. Unconditional overhead free program loop constructs are supported using the `DO` and `REPEAT` instructions, both of which are interruptible at any point.

The working register array consists of 16 x 16-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). 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 optionally 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 the sole limitation that the access requires an additional cycle. Moreover, only the lower 16 bits of each instruction word can be accessed using this method.
- Linear 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 7.0 for details on modulo and bit-reversed addressing.

The core supports Inherent (no operand), Relative, Literal, Memory Direct, Register Direct, Register Indirect and Register 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.

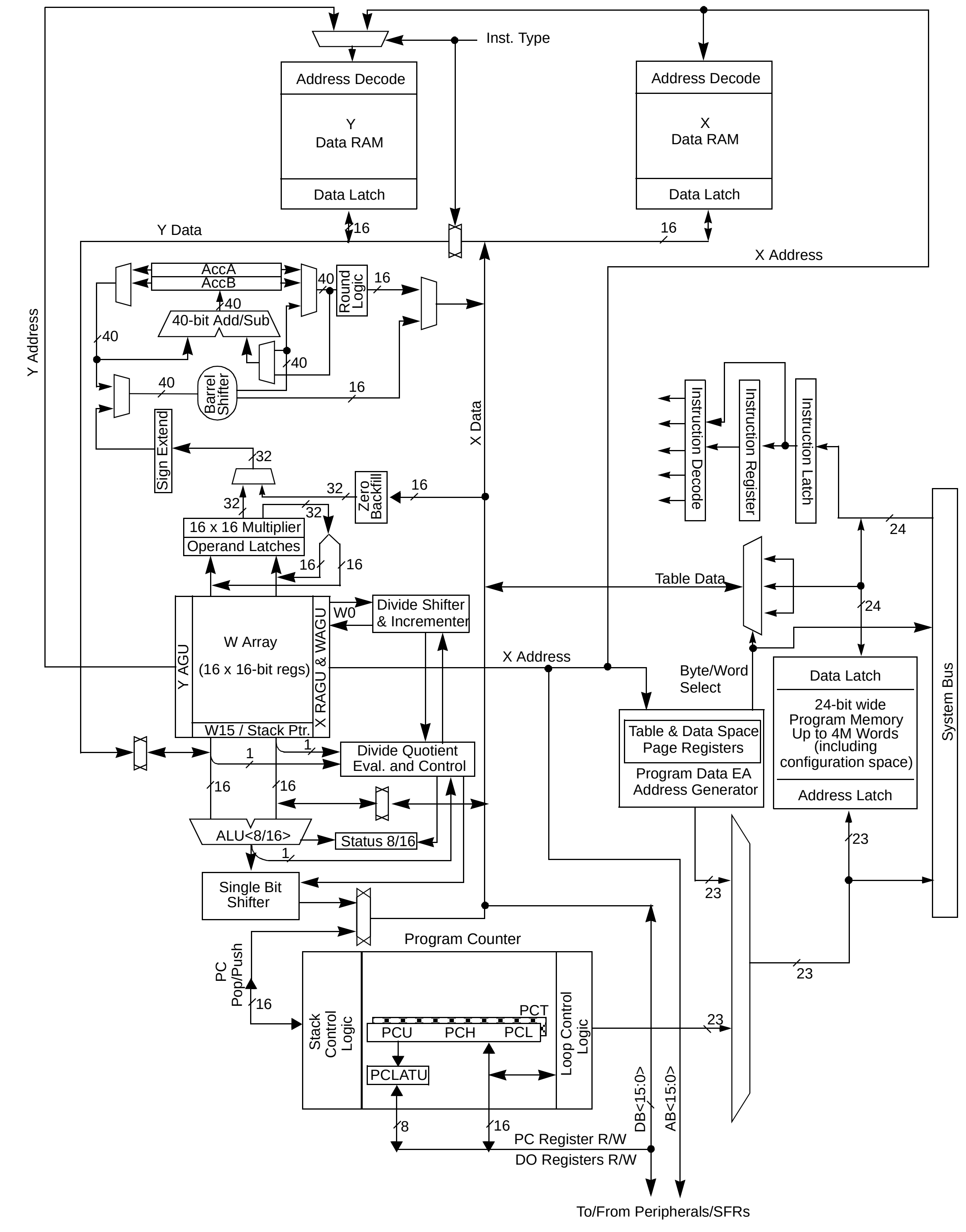
A DSP engine has been included to significantly enhance the core arithmetic capability and throughput. It features a high speed 16-bit by 16-bit multiplier, a 40-bit ALU, two 40-bit saturating accumulators and a 40-bit bi-directional barrel shifter. Data in the accumulator or any working register can be shifted up to 15 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 is split for these instructions and linear for all others. This is achieved in a transparent and flexible manner, through 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 pre-fetch mechanism is used, which accesses and partially predecodes instructions a cycle ahead to maximize available execution time. Most instructions execute in a single cycle, with certain exceptions as outlined in Section 2.3.2.

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 3 are reserved) and 54 interrupts. Each interrupt is prioritized based on a user assigned priority between 0 and 7 (0 being the lowest priority and 7 being the highest) in conjunction with a predetermined ‘natural order’.

A block diagram of the core is shown in Figure 2-1.

FIGURE 2-1: dsPIC30F CPU CORE BLOCK DIAGRAM



2.2 Programmer's Model

The programmer's model is shown in Figure 2-3 and consists of 16 x 16-bit working registers (W0 through W15), 2 x 40-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.

Most of these registers have a shadow register associated with them, as shown in Figure 2-3. The shadow register is used as a temporary holding register and can transfer its contents to or from its host register upon some event occurring. 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...W14, TBLPAG, PSVPAG, SR (DC, N, OV, SZ and C bits only) transferred
- `DO` instruction
  DA bit, DOSTART, DOEND, DCOUNT shadows pushed on loop start, 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

W15 is the dedicated software stack pointer (SP), and will be 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.

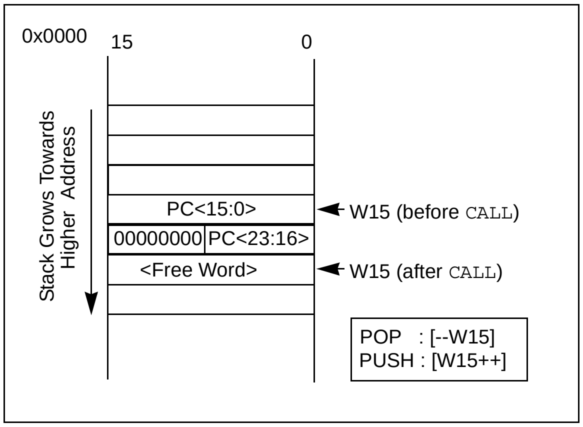
The stack pointer always points to the first available free word and grows from lower addresses towards higher addresses. It pre-decrements for stack pops (reads) and post-increments for stack pushes (writes), as shown in Figure 2-2. Note that for a PC push during any `CALL` instruction, the MSB of the PC is zero extended before the push, ensuring that the MSB is always clear.

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

There is a Stack 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 EA is found to be greater than the contents of SPLIM, then a Stack Pointer Overflow trap is generated.

Similarly, a Stack Pointer Underflow 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.

FIGURE 2-2: `CALL` STACK FRAME



2.2.2 STATUS REGISTER

The dsPIC™ core has a 16-bit status register (SR), the LS Byte of which is referred to as the STATUS register low byte (SRL). A detailed description is shown in Register 2-1.

> **Note:** When the memory mapped STATUS register is the destination address for an operation which affects any of the SR bits, data writes are disabled to all bits.

SRL contains all the MCU ALU operation status flags (including the ‘sticky Z’ bit, SZ), as well as the CPU Interrupt Priority status bits, IPL<2:0>, and the REPEAT active status bit, RA. During exception processing, SRL is concatenated with the MS Byte of the PC to form a complete word value which is then stacked.

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

Most SR bits are read/write. Exceptions are:

1. The DA bit: DA is read and clear only, because accidentally setting it could cause erroneous operation.
2. The RA bit: RA is a read only bit, because accidentally setting it could cause erroneous operation. RA is only set on entry into a repeat loop, and cannot be directly cleared by software.
3. The OV, OA, OB and OAB bits: These bits are read only and can only be set by the DSP engine overflow logic.
4. The SA, SB and SAB bits: These are read and clear only and can only be set by the DSP engine saturation logic. They are ‘sticky’, i.e., once set, they remain set until cleared by the user, irrespective of the results from any subsequent DSP operations.

> **Note:** Clearing the SAB bit will also clear both the SA and SB bits.

2.2.2.1 Sticky Z (SZ) Status Bit

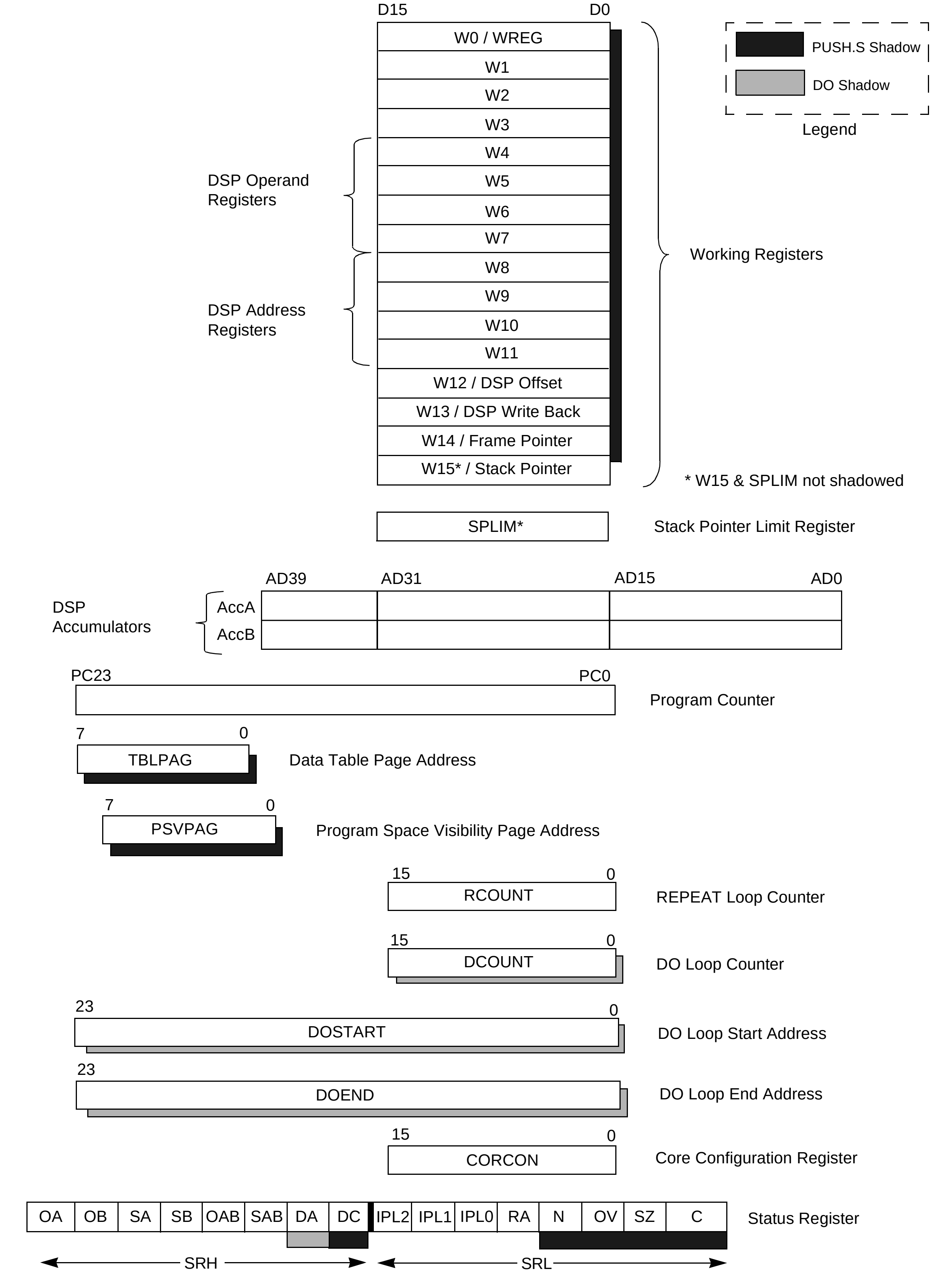
For most instructions, the SZ status bit is not sticky. Instructions that use a carry/borrow input will only be able to clear SZ (for a non-zero result) and can never set it. A multi-precision sequence of instructions (starting with an instruction with no carry/borrow input) will thus, automatically logically AND the successive results of the zero test. All results must be zero for the SZ flag to remain set by the end of the sequence.

The following instructions feature sticky operation of the SZ flag: `ADDC, CPB, SUBB` and `SUBBR`.

2.2.3 PROGRAM COUNTER

The Program Counter is 24-bits wide. Bit 0 is always clear. PC<23> may be used to access configuration fuse settings, using table read and table write instructions. Bit 23 is also clear for normal user program memory access. Therefore, the PC can address up to 4M instruction words.

FIGURE 2-3: PROGRAMMER'S MODEL



REGISTER 2-1: STATUS REGISTER (SR)

Upper Half (SRH):

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0
OA	OB	SA	SB	OAB	SAB	DA	DC
bit 15							bit 8

Lower Half (SRL):

R/W-1	R/W-1	R/W-1	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL2	IPL1	IPL0	RA	N	OV	SZ	C
bit 7							bit 0

bit 15 **OA:** Accumulator A Overflow Status bit
1 = Accumulator A overflowed
0 = Accumulator A not overflowed

bit 14 **OB:** Accumulator B Overflow Status bit
1 = Accumulator B overflowed
0 = Accumulator B not overflowed

bit 13 **SA:** Accumulator A Saturation Status bit
1 = Accumulator A is saturated or has been saturated at some time
0 = Accumulator A is not saturated

Note: This bit may be read or cleared, but not set.

bit 12 **SB:** Accumulator B Saturation Status bit
1 = Accumulator B is saturated or has been saturated at some time
0 = Accumulator B is not saturated

Note: This bit may be read or cleared, but not set.

bit 11 **OAB:** OA || OB Combined Accumulator Overflow Status bit
1 = Accumulators A or B have overflowed
0 = Neither Accumulators A or B have overflowed

bit 10 **SAB:** SA || SB Combined Accumulator Status bit
1 = Accumulators A or B are saturated or have been saturated at some time in the past
0 = Neither Accumulator A or B are saturated

Note: This bit may be cleared or read, but not set. Clearing this bit will clear SA and SB.

bit 9 **DA:** DO Loop Active bit
1 = DO loop in progress
0 = DO loop not in progress

Note: This bit may be read or cleared, but not set.

bit 8 **DC:** MCU ALU Half-Carry bit
1 = A carry-out from the 4th low order bit (for byte sized data) or 8th low order bit (for word sized data) of the result occurred
0 = No carry-out from the 4th low order bit (for byte sized data) or 8th low order bit (for word sized data) of the result occurred

bit 7-5 **IPL<2:0>:** CPU Interrupt Priority Status bits
111 = Level 7 interrupts enabled
110 = Level 6 to 7 interrupts enabled
101 = Level 5 to 7 interrupts enabled
100 = Level 4 to 7 interrupts enabled
011 = Level 3 to 7 interrupts enabled
010 = Level 2 to 7 interrupts enabled
001 = Level 1 to 7 interrupts enabled
000 = Level 0 to 7 interrupts enabled

bit 4 **RA:** REPEAT Loop Active Status bit
1 = REPEAT loop in progress
0 = REPEAT loop not in progress

REGISTER 2-1: STATUS REGISTER (SR) (Continued)

bit 3 **N:** MCU ALU Negative bit
This bit is used for signed arithmetic (2’s complement). It indicates whether the result of the ALU operation was negative (ALU MSb = 1).
`1` = Result was negative
`0` = Result was non-negative (zero or positive)

bit 2 **OV:** MCU ALU Overflow bit
This bit is used for signed arithmetic (2’s complement). It indicates an overflow of the magnitude, which causes the sign bit to change state.
`1` = Overflow occurred for signed arithmetic (in this arithmetic operation)
`0` = No overflow occurred
Example 1: w1 before instruction = 0x7fff. OV = 0.
Instruction executed: ADD #1,w1
w1 after instruction = 0x8000. OV = 1.
Example 2: w1 before instruction = 0xffff. OV = 0.
Instruction executed: ADD #1,w1
w1 after instruction = 0x0000. OV = 1.

bit 1 **SZ:** MCU ALU ‘sticky’ Zero bit
`1` = An `ADDC, CPB, SUBB` or `SUBBR` operation which affects the SZ bit has set it at some time in the past
`0` = The most recent operation which affects the SZ bit has cleared it (i.e., generated a non-zero result)
For all other operations, this bit indicates whether the most recent operation has produced a zero result.

bit 0 **C:** MCU ALU Carry/$\overline{\text{Borrow}}$ bit
`1` = A carry-out from the Most Significant bit of the result occurred
`0` = No carry-out from the Most Significant bit of the result occurred
For $\overline{\text{Borrow}}$, the polarity is reversed.

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

2.3 Instruction Flow

2.3.1 CLOCKING SCHEME

Each instruction cycle (TCY) is comprised of four Q cycles (Q1-Q4). The four-phase Q cycles provide the timing/delineation for the phases of the instruction cycle, such as Decode, Read, Process Data, and Write. Figure 2-4 shows the relationship of the Q cycles to the instruction cycle for both MCU and DSP instructions. The four Q cycles that make up an execution instruction cycle (TCY) can be generalized as:

Q1:
Instruction decode or forced `NOP` (`FNOP`)
Data space read start
Destination EA calculation start

Q2:
Instruction decode or `FNOP`
Data space read complete
Destination EA calculation

Q3:
Next instruction pre-decode start
Next op read EA calculation start
Process the Data or `FNOP`
Data space write start

Q4:
Next instruction pre-decode
Next op read EA calculation
Data space write complete or `FNOP`

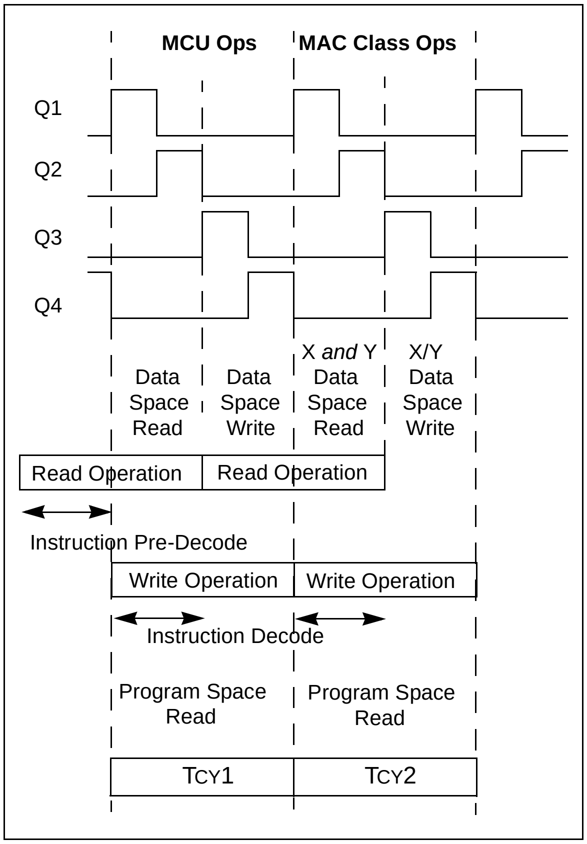
> **Note:** From a Q cycle perspective, the DSP instructions differ from the MCU instructions in their ability to perform two simultaneous source data reads during the Q1/Q2 access from X and Y data space.

The total time required for EA calculation and data space access is 4 Q clocks, for both read and write accesses. This device does not support any form of dual ported data space, so read and write operations cannot occur simultaneously. As a result, at least a portion of these operations occur concurrently across instruction boundaries. Data space accesses are partially pipelined.

There are separate AGUs for X reads and X writes so as to enable concurrent operation. The X RAGU (Read AGU) and X WAGU (Write AGU) serve this purpose and are functionally identical, with the exception that bit-reversed addressing is only supported on the X WAGU. There is only a single Y AGU.

A consequence of this degree of pipelining is that data dependencies can now occur between completing destination write EA operations and already started source read EA calculations. Data dependencies are discussed in detail in Section 7.

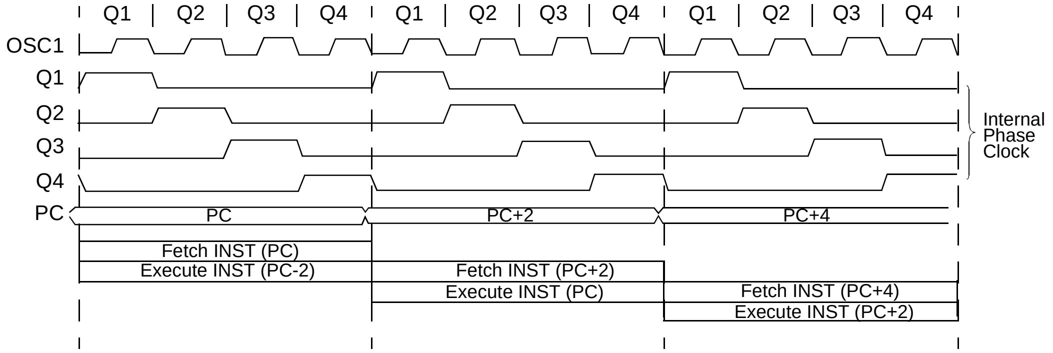
FIGURE 2-4: BASIC CORE TIMING



2.3.2 INSTRUCTION FETCH AND PRE-DECODE MECHANISM

The core does not support an instruction pipeline. A pre-fetching mechanism accesses instructions a cycle ahead to maximize available execution time and to allow for some pre-decode to occur. A conceptual timing diagram, demonstrating the pre-fetch mechanism, is shown in Figure 2-5.

FIGURE 2-5: CLOCK/INSTRUCTION CYCLE



Most instructions execute in a single cycle. Exceptions are:

1. Flow control instructions and interrupts where the IR (Instruction register) and pre-fetch buffer must be flushed and refilled.
2. Instructions where one operand is to be fetched from program space (using any method). These operations consume 2 cycles (with the notable exception of instructions executed within a REPEAT loop; in this case, it would execute in 1 cycle).
3. Instructions where an effective address calculation dependency has been detected and an instruction stall must be inserted.

Most instructions access data as required during instruction execution. Instructions which utilize the multiplier array must have data available at the beginning of the instruction cycle. Consequently, this data must be pre-fetched, usually by the preceding instruction, resulting in a simple out of order data processing model.

During the instruction pre-decode, the core determines if any address register data dependency is imminent across an instruction boundary. It compares the working register (if any) used for the destination EA (effective address) of the instruction currently being executed, with that about to be used by the source EA (if any) of the pre-fetched instruction. When it observes a match between the destination and source registers, a set of rules is applied to decide (by the falling edge of Q3) whether or not to stall the instruction by one cycle. See Section 7.0 for more details on data dependencies.

Due to the instruction pre-fetch mechanism, each instruction effectively executes in one cycle. This remains true until a flow change occurs, which will require the pre-fetch register (ROMLATCH) to be discarded and refilled again from the new instruction thread.

2.3.3 INSTRUCTION FLOW TYPES

There are 8 types of instruction flows:

1. Normal one-word, one-cycle pipelined instructions. These instructions take one effective cycle to execute, as shown in Figure 2-6.
2. One-word, two-cycle pipeline flush instructions. These instructions include the relative branches, relative call, skips and returns. When an instruction changes the PC (other than to increment it), the pipelined fetch is discarded. This causes the instruction to take two effective cycles to execute as shown in Figure 2-7.
3. One-word, two-cycle instructions that are not flow control instructions. The only instructions of this type are the `MOV.D` (load and store double word) instructions, as shown in Figure 2-8.
4. Table read/write instructions. These instructions will suspend the fetching to insert a read or write cycle to the program memory. The instruction fetched while executing the table operation is saved for 1 cycle and executed in the cycle immediately after the table operation, as shown in Figure 2-9.
5. Two-word instructions for `CALL` and `GOTO`. In these instructions, the fetch after the instruction provides the remainder of the jump or call destination address. These instructions require 2 cycles to execute, 1 for fetching the 2 instruction words (enabled by a high speed path on the second fetch), and 1 for the subsequent pipeline flush, as shown in Figure 2-10.
6. Two-word instructions for DO. In these instructions, the fetch after the instruction contains an address offset. This address offset is added to the first instruction address to generate the last loop instruction address. Therefore, these instructions require two cycles, as shown in Figure 2-11.
7. Instructions that are subjected to a stall due to a data dependency between the X RAGU and X WAGU. An additional cycle is inserted to resolve the resource conflict, as shown in Figure 2-11. Instruction stalls caused by data dependencies are further discussed in Section 7.0.
8. Interrupt recognition execution. Refer to Section 8.0 for details on interrupts.

FIGURE 2-6: INSTRUCTION PIPELINE FLOW - 1-WORD, 1-CYCLE

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
`1. MOV.b #0x55,W0`	Fetch 1	Execute 1				
`2. MOV.b #0x35,W1`		Fetch 2	Execute 2			
`3. ADD.b W0,W1,W2`			Fetch 3	Execute 3		

FIGURE 2-7: INSTRUCTION PIPELINE FLOW - 1-WORD, 2-CYCLE

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
`1. MOV  #0x55,W0`	Fetch 1	Execute 1				
`2. BTSC W1,#3`		Fetch 2	Execute 2 Skip Taken			
`3. ADD  W0,W1,W2`			Fetch 3	Flush		
`4. BRA  SUB_1`				Fetch 4	Execute 4	
`5. SUB  W0,W1,W3`					Fetch 5	Flush
`6. Instruction @ address SUB_1`						Fetch SUB_1

FIGURE 2-8: INSTRUCTION PIPELINE FLOW - 1-WORD, 2-CYCLE `MOV.D` OPERATIONS

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
`1. MOV  W0,0x1234`	Fetch 1	Execute 1				
`2. MOV.D [W0++],W1`		Fetch 2	Execute 2 R/W Cycle 1			
`3. MOV  W1,0x00AA`			Fetch 3	Execute 2 R/W Cycle2		
`3a. Stall`				Stall	Execute 3	
`4. MOV  0x0CC, W0`					Fetch 4	Execute 4

FIGURE 2-9: INSTRUCTION PIPELINE FLOW - 1-WORD, 2-CYCLE TABLE OPERATIONS

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
`1. MOV #0x1234,W0`	Fetch 1	Execute 1				
`2. TBLRDL [W0++],W1`		Fetch 2	Execute 2			
`3. MOV #0x00AA,W1`			Fetch 3	Execute 2 Read Cycle		
`3a. Table Operation`				Bus Read	Execute 3	
`4. MOV  #0x0CC,W0`					Fetch 4	Execute 4

FIGURE 2-10: INSTRUCTION PIPELINE FLOW - 2-WORD, 2-CYCLE `GOTO, CALL`

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
`1. MOV #0x1234,W0`	Fetch 1	Execute 1				
`2. GOTO LABEL`		Fetch 2L	Update PC			
`2a. Second Word`			Fetch 2H	`NOP`		
`3. Instruction @ address LABEL`				Fetch LABEL	Execute LABEL	
`4. BSET  W1, #BIT3`					Fetch 4	Execute 4

FIGURE 2-11: INSTRUCTION PIPELINE FLOW - 2-WORD, 2-CYCLE `DO, DOW`

	TCY0	TCY1	TCY2	TCY3	TCY4
`1. PUSH DOEND`	Fetch 1	Execute 1			
`2. DO LABEL,#COUNT`		Fetch 2L	`NOP`		
`2a. Second Word`			Fetch 2H	Execute 2	
`3. 1st Instruction of Loop`				Fetch 3	Execute 3

FIGURE 2-12: INSTRUCTION PIPELINE FLOW - 1-WORD, 2-CYCLE WITH INSTRUCTION STALL

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
`1. MOV.b W0,[W1]`	Fetch 1	Execute 1				
`2. MOV.b [W1],PORTB`		Fetch 2	`NOP`			
`2a. Stall (NOP)`			Stall	Execute 2		
`3. MOV.b W0,PORTB`				Fetch 3	Execute 3	

2.4 Program Loop Control

The dsPIC core supports both REPEAT and DO instruction constructs to provide unconditional automatic program loop control.

2.4.1 REPEAT LOOP CONSTRUCT

The REPEAT instruction will cause the RA (Repeat Active) flag bit to be set if the REPEAT count is greater than 0. If RA = 1, PC increments and instruction fetches are inhibited until the REPEAT loop counter, RCOUNT, reaches 0 (at which point RA will also be cleared by the loop count hardware).

The REPEAT instruction causes the instruction immediately following it to be repeated a fixed number of times as defined by either:

- a fixed, 14-bit literal encoded in the instruction, or
- the variable contents of bits <13:0> of a W register declared within the instruction

The loop count is held in the 16-bit RCOUNT register (which is memory mapped) and is thus, user accessible. It is initialized by the REPEAT instruction during Q2.

The instruction to be repeated is pre-fetched during the REPEAT instruction and held in the ROMLATCH. It is not fetched again for all subsequent iterations, and the Instruction register is loaded from the locked ROMLATCH.

Note:	For a loop count value of 0, REPEAT will be executed like a NOP. The RA status bit is not set, but RCOUNT is loaded with the value 0. However, the instruction within the REPEAT loop is executed once.

2.4.2 REPEAT LOOP INTERRUPT AND NESTING

A REPEAT instruction loop may be interrupted at any time. As is the case for all instructions, the PC will not be incremented during the instruction when an exception is acknowledged. For a repeated instruction, the PC update is already inhibited (by the RA bit) which ensures that, upon return, the RETFIE instruction will correctly pre-fetch the instruction (i.e. the stacked PC will point to the instruction to be repeated).

The RA state bit being present in SRL, is automatically saved on the stack during exception processing. This enables execution of further REPEAT loops from within nested interrupts. After SRL is stacked, RA is cleared during exception processing to restore normal execution within the ISR. Exception processing operates as normal (i.e., with instruction pre-fetch) irrespective of the state of RA.

Note:	The user must stack the RCOUNT (Repeat Count register), prior to executing a REPEAT within an ISR.

2.4.3 REPEAT EARLY TERMINATION

Clearing the RA bit in the stacked SR from within an ISR is a method to force an interrupted loop to terminate early (subject to one more iteration) after the interrupt or trap returns. RA is not software modifiable within the SR, but will be updated with whatever value is present in the stacked SRL during a return. Even if the RCOUNT ‘decrement then test’ does not indicate a loop end condition (RCOUNT = 0), RA will remain clear and will force the loop to exit. A subsequent REPEAT instruction will set RA, but will also update RCOUNT with the new loop count.

2.4.4 DO LOOP CONSTRUCT

The DO instruction executes instructions following the DO until an end address is reached, at which time instruction execution will start again at the instruction immediately following the DO. This will be repeated a finite number of times as defined by either:

- a fixed, 14-bit literal encoded in the first word of the instruction, or
- the variable contents of bits <13:0> of a W register declared within the instruction

The instructions within a loop (including the instruction immediately preceding the last instruction in the loop) need not be executed in the same order in which they appear in program memory, i.e., branches within a loop are allowed. Moreover, the loop end address may be smaller than the start address.

The instruction executed immediately before the last instruction in the loop does not have to be the one immediately preceding the last loop instruction in program memory.

The DO loop will always be executed at least once, since the loop count is examined only at the end of each iteration. For a DCOUNT loop count value of 0, DO will iterate the loop once.

Note:	The loop end comparison is an equality test only. The instruction at the loop end address must be pre-fetched, in order for the end of the loop condition to be recognized. That is, exiting the loop to a PC value greater than the end address (or less than the start address) will not cause the loop count to change. An exact address match must occur in order to change the loop count.

2.4.5 DO LOOP NESTING

The DOSTART, DOEND and DCOUNT loop registers have a shadow register associated with each of them. This permits a single level of nesting. In addition, as the DOSTART, DOEND and DCOUNT registers are user accessible, they may be manually saved to permit additional nesting.

When a DO is executed, the DOSTART, DOEND and DCOUNT registers are transferred into the shadow registers, prior to being updated with the new loop values. The DA bit (SR<7>) is also shadowed prior to being set during DO execution. Similarly, during all loop exits, the shadow contents of the DOSTART, DOEND and DCOUNT registers and the DA bit are transferred back into their respective host registers.

2.4.6 DO LOOPS AND INTERRUPTS

A DO loop may be interrupted at any time without penalty.

Note:	The LS Byte of the SR (SRL), which is stacked during exception processing, does not include the DA bit.

If a DO loop is required within an exception handler, the DOSTART, DOEND, DCOUNT and MS Byte of the SR must be stacked before another DO loop may be executed. These registers must be restored prior to returning from the ISR or trap handler.

2.4.7 DO EARLY TERMINATION

The DA bit in the MS Byte of the SR may be cleared (but not set) by the user. Clearing the DA bit is a method to force a loop to terminate early at any time. The loop will complete the current iteration and then terminate. If DA is cleared during one of the last two instructions of the loop, one more iteration of the loop will occur.

A branch with a target instruction outside the loop may be executed from within the loop. However, the last instruction in the DO loop cannot be a flow control instruction (see Section 2.4.8). It is recommended that the DA bit is cleared by software whenever a DO loop is terminated early.

2.4.8 DO AND REPEAT RESTRICTIONS

Any instruction can follow a `REPEAT` except for:

1. Flow control (any branch, compare and skip, `GOTO`, `CALL`, `RCALL`, `RETURN`, `RETLW` or `RETFIE`) instructions
2. `DISI`, `ULNK`, `LNK`, `RESET` and `PWRSAV` instructions
3. Another `REPEAT` or `DO`

All DO loops must contain at least 2 instructions, because the loop termination tests are performed in the second last instruction executed. REPEAT should be used for single instruction loops. All other restrictions with regard to the DO loop revolve around the last two instructions. The last instruction in a DO loop should not be:

1. Flow control (any branch, compare and skip, `GOTO`, `CALL`, `RCALL`, `RETURN`, `RETLW` or `RETFIE`) instructions, except the indirect `CALL` `(CALL Wn)`
2. Another `REPEAT` or `DO`
3. Instruction within a REPEAT loop
4. Any 2-word instruction

The (one-word) `CALL (CALL Wn)` instruction will function correctly at the end of a DO loop because the stacked PC will address the first instruction in the loop (to fetch upon return).

The last instruction in a DO loop should not be either a `REPEAT` instruction, or the instruction repeated within a REPEAT loop. In such cases, the DO loop counter will take priority, and the `REPEAT` instruction will not function correctly.

If the last instruction of a DO loop or the instruction within a REPEAT loop detects a data dependency that indicates an instruction stall is necessary, the extra cycle will be expended and the loop will continue normally.

2.5 Divide Support

Note: The Divide Support architecture is under development. Please refer to our Microchip website (http://www.microchip.com) for future updates.

The dsPIC devices 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.sw` — 16/16 signed divide
5. `DIV.uw` — 16/16 unsigned divide

The 16/16 divides are similar to the 32/16 (same number of iterations), but either zero or sign extend the dividend during the first iteration.

The quotient for all divide instructions ends up in W0, and the remainder in W1. `DIV` and `DIVF` can specify any W register for both the 16-bit dividend and divisor. All other divides can specify any W register for the16-bit divisor, but the 32-bit dividend must be in the W1:W0 register pair.

The non-restoring divide algorithm requires one cycle for an initial dividend shift for (integer divides only) one cycle per divisor bit, plus a remainder/quotient correction cycle. The correction cycle is the last cycle of the iteration loop, but must be performed (even if the remainder is not required) because it may also adjust the quotient. A consequence of this is that `DIVF` will also produce a valid remainder (though it is of little use in fractional arithmetic).

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 is conditional 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` will execute the target instruction {operand value+1} times).

Note: The Divide flow is interruptible.

TABLE 2-1: DIVIDE EXECUTION TIME

Instruction	Function	Iterations	REPEAT Operand Value	Total Execution Time (Incl. REPEAT)
`DIVF`	Signed fractional divide: Wm/Wn → W0; Rem → W1	18	17	19
`DIV.sd`	Signed divide: (Wm+1:Wm)/Wn → W0; Rem → W1	18	17	19
`DIV.sw`	Signed divide: Wm/Wn → W0; Rem → W1	18	17	19
`DIV.ud`	Unsigned divide: (Wm+1:Wm)/Wn → W0; Rem → W1	18	17	19
`DIV.uw`	Unsigned divide: Wm/Wn → W0; Rem → W1	18	17	19

3.0 MEMORY ORGANIZATION

3.1 Program Address Space

The program address space is 4M instruction words. It is addressable by a 24-bit value from either the PC, table instruction 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.

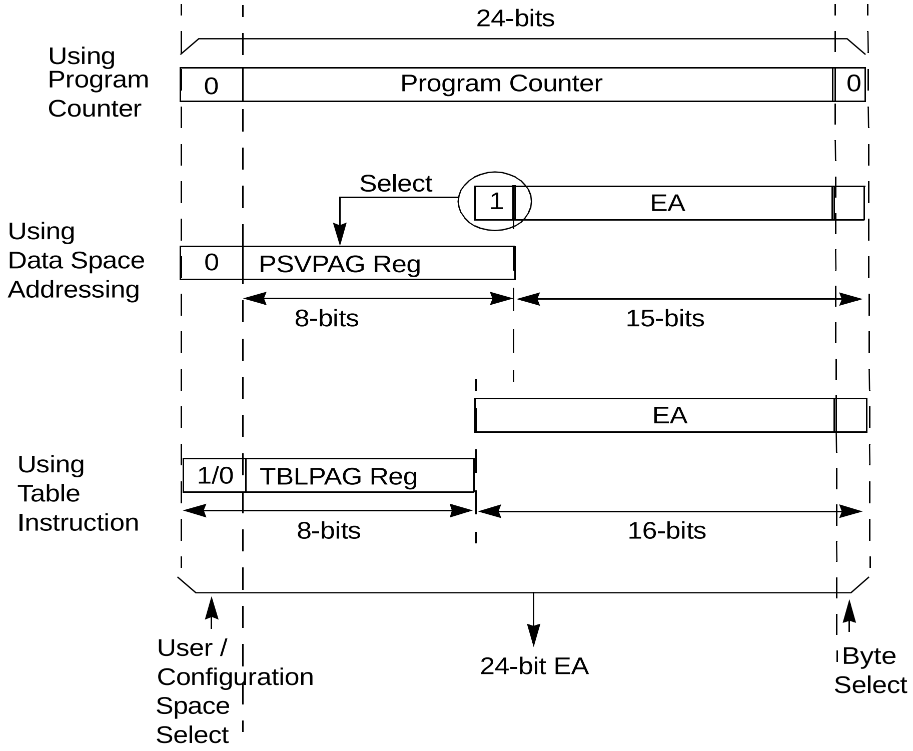
User program space access is restricted to the lower 4M instruction word address range (0x000000 to 0x7FFFFE), for all accesses other than TBLRD/TBLWT, which use TBLPAG<7> to determine user or configuration space access. In Table Read/Write instructions, bit 23 allows access to the Device ID, the User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

Note: The address map shown in Figure 3-4 is conceptual, and the actual memory configuration may vary across individual devices depending on available memory.

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>		
DS Window into PS	User	0	PSVPAG<7:0>		Data EA <14:0>	

FIGURE 3-1: DATA ACCESS FROM PROGRAM SPACE ADDRESS GENERATION



Note: PSVPAG cannot be used to access bits <23:16> of a word in program memory.

3.1.1 PROGRAM SPACE ALIGNMENT AND DATA ACCESS 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). The `TBLRDL` and `TBLWTL` instructions offer a direct method of reading or writing the LS Word 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 word can be accessed as data.

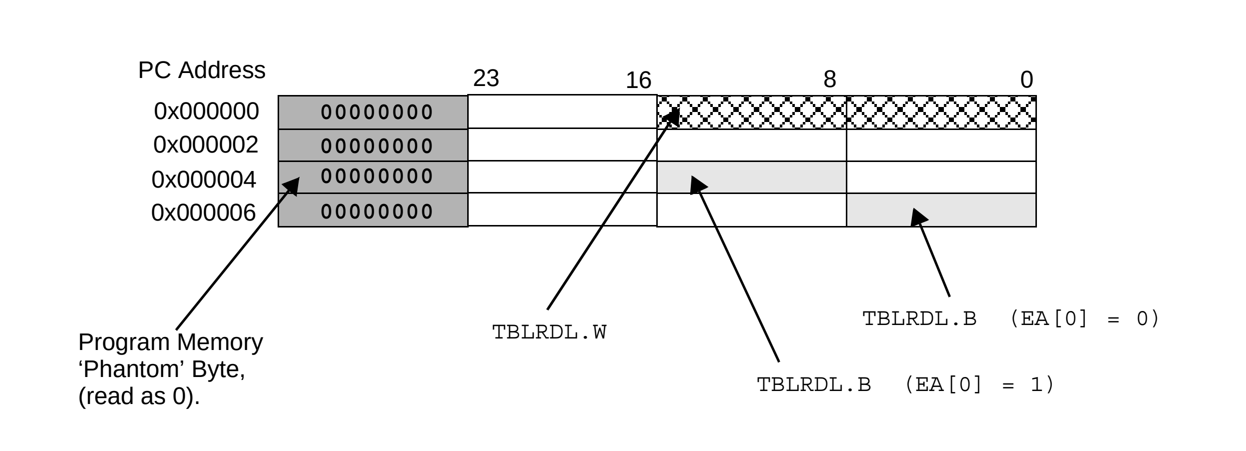
The PC is incremented by two for each successive 24-bit program word. This allows program memory addresses to directly map to data space addresses. Program memory can thus be regarded as two 16-bit word wide address spaces, residing side by side, each with the same address range. `TBLRDL` and `TBLWTL`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

Figure 3-1 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 are provided to move byte or word sized data to and from program space.

1. **`TBLRDL:`** Table Read Low
   Word: Read the LS Word of the program address;
   P<15:0> maps to D<15:0>.
   Byte: Read one of the LS Bytes of the program address;
   P<7:0> maps to D<7:0> when byte select = 0;
   P<15:8> maps to D<7:0> when byte select = 1.
2. **`TBLWTL:`** Table Write Low (refer to Section 4.0 for details on FLASH Programming).
3. **`TBLRDH:`** Table Read High
   Word: Read the MS Word of the program address;
   P<23:16> maps to D<7:0>; D<15:8> always = 0.
   Byte: Read one of the MSBs of the program address;
   P<23:16> maps to D<7:0> when byte select = 0;
   D<7:0> will always = 0 when byte select = 1.
4. **`TBLWTH:`** Table Write High (refer to Section 4.0 for details on FLASH Programming).

FIGURE 3-2: PROGRAM DATA TABLE ACCESS (LS WORD)



3.1.2 PROGRAM SPACE VISIBILITY FROM DATA SPACE

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 MS bit 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 6.0, DSP Engine.

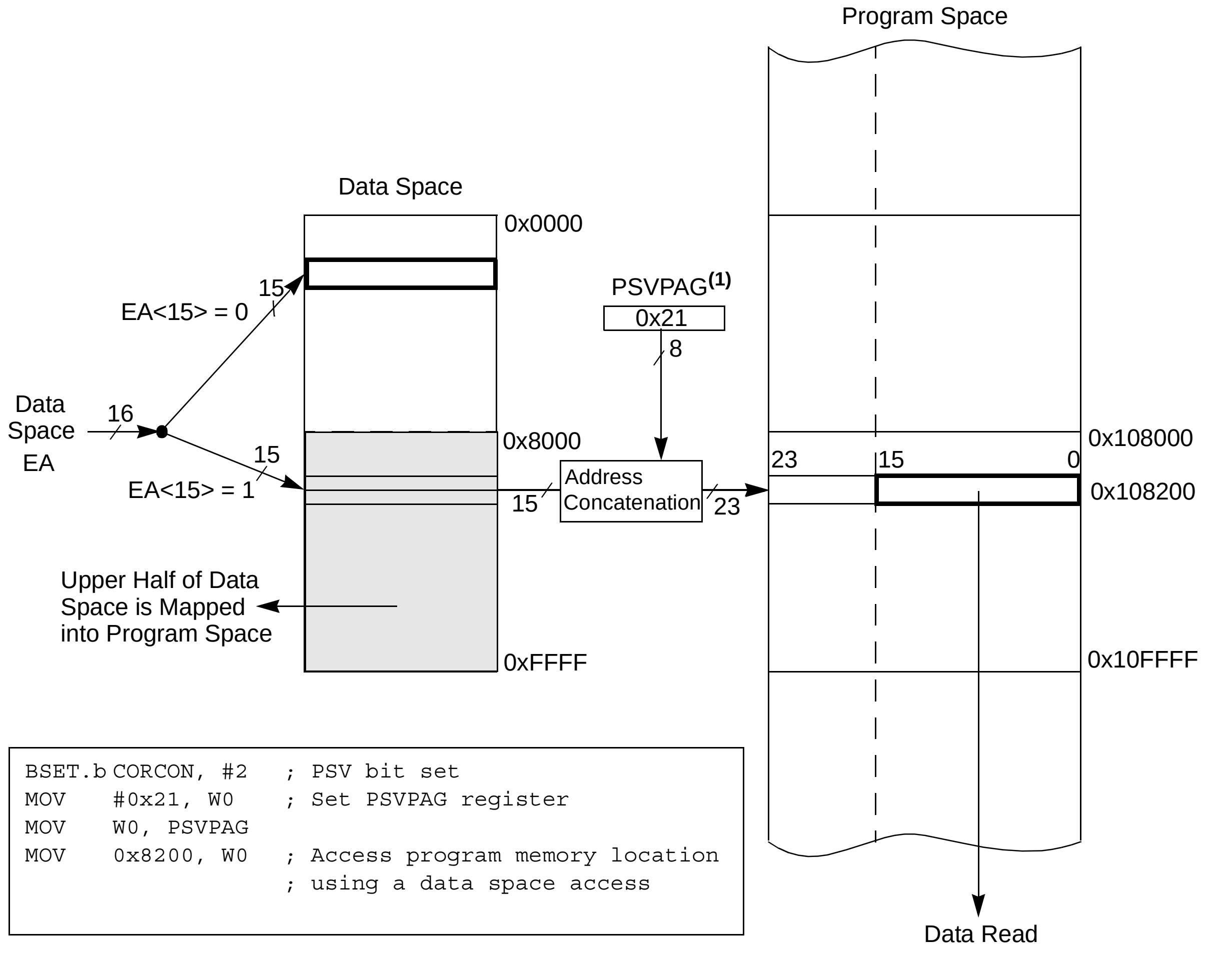
Data accesses to this area add an additional cycle to the instruction being executed, since two program memory fetches are required.

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 3-3), 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, or software trap, to maintain machine robustness. Refer to the Programmer's Reference Manual (DS70030) for details on instruction encoding.

Note that by incrementing the PC by 2 for each program memory word, the LS 14 bits (15 bits for the `TBLRDL/H, TBLWTL/H` instructions) of data space addresses directly map to the LS 14 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 3-3.

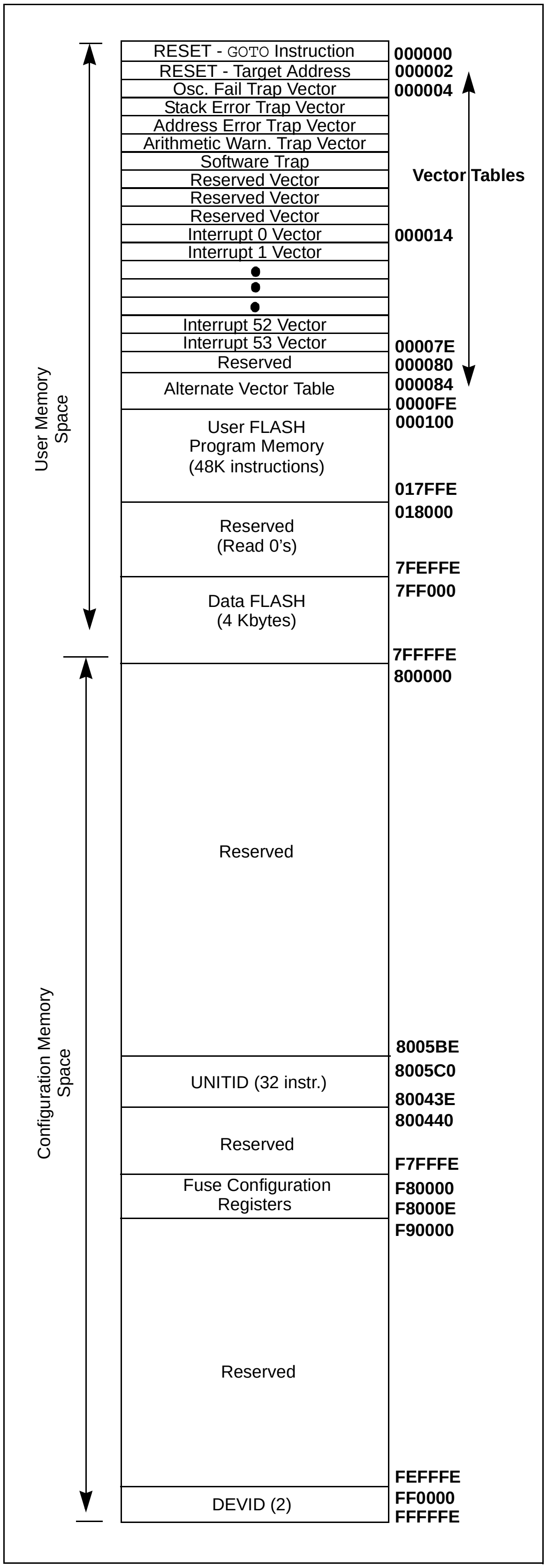
FIGURE 3-3: DATA SPACE WINDOW INTO PROGRAM SPACE OPERATION



```
BSET.b CORCON, #2   ; PSV bit set
MOV    #0x21, W0    ; Set PSVPAG register
MOV    W0, PSVPAG
MOV    0x8200, W0   ; Access program memory location
                    ; using a data space access
```

Note 1: PSVPAG is an 8-bit register, containing bits <22:15> of the program space address (i.e., it defines the page in program space to which the upper half of data space is being mapped).

FIGURE 3-4: SAMPLE PROGRAM SPACE MEMORY MAP



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 SPACES

The X AGU is used by all instructions and supports all addressing modes. It consists of a read AGU (X RAGU) and a write AGU (X WAGU), which operate independently during different phases of the instruction cycle. 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 RAGU and X WAGU also support Modulo Addressing for any instructions subject to addressing mode restrictions. Bit-Reversed Addressing is only supported by X WAGU.

The Y AGU and data path are used in concert with the X AGU 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 operate through the Y AGU and address Y data space, independent of X data space, whereas W8 and W9 operate through the X RAGU and address X data space. Note that during accumulator write back, the data address space is considered a combination of X and Y, so the write occurs across the X bus. Consequently, it can be to any address in the entire data space.

The Y AGU only supports the Post-Modification Addressing modes 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 AGU, when it is regarded 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 will be 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), will return 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 are 16-bits wide and point to bytes within the data space. Therefore, the data space address range is 64 Kbytes or 32K words.

3.2.2 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.

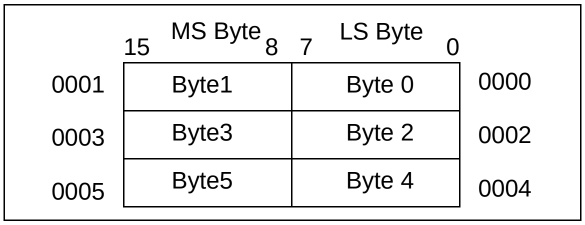
3.2.3 DATA ALIGNMENT

To help maintain backward compatibility with PICmicro® 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 will read the complete word, which contains the byte, using the LS bit of any EA to determine which byte to select. The selected byte is placed onto the LS Byte of the X data path (no byte accesses are possible from the Y data path as the `MAC` class of instruction 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. Mis-aligned 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 will be forced. If the error occurred on a read, the instruction underway is completed, whereas if it occurred on a write, the instruction will be inhibited and the PC will not be incremented. In either case, a trap will then be executed, allowing the system and/or user to examine the machine state prior to execution of the address fault.

FIGURE 3-5: DATA ALIGNMENT



	15 MS Byte 8	7 LS Byte 0	
0001	Byte1	Byte 0	0000
0003	Byte3	Byte 2	0002
0005	Byte5	Byte 4	0004

All byte loads into any W register are loaded into the LS Byte. 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.

EXAMPLE 3-1: `SE` AND `ZE` OPERATION

```
MOV.b   #0x7f, W1        ; W1 = 0x007f (0000 0000 0111 1111)
SE      W1,W1            ; W1 = 0x007f (0000 0000 0111 1111)
                         ; Sign bit = 0 is extended to MS byte

MOV.b   #0xf6, W2        ; W2 = 0x00f6 (0000 0000 1111 0110)
SE      W2,W2            ; W2 = 0xfff6 (1111 1111 1111 0110)
                         ; Sign bit = 1 is extended to MS byte

MOV     #WREG2, W3       ; WREG2 is the memory-mapped address of W2
ZE      [W3],[W3]        ; W3 = 0x00f6 (0000 0000 1111 0110)
```

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.

3.2.4 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.

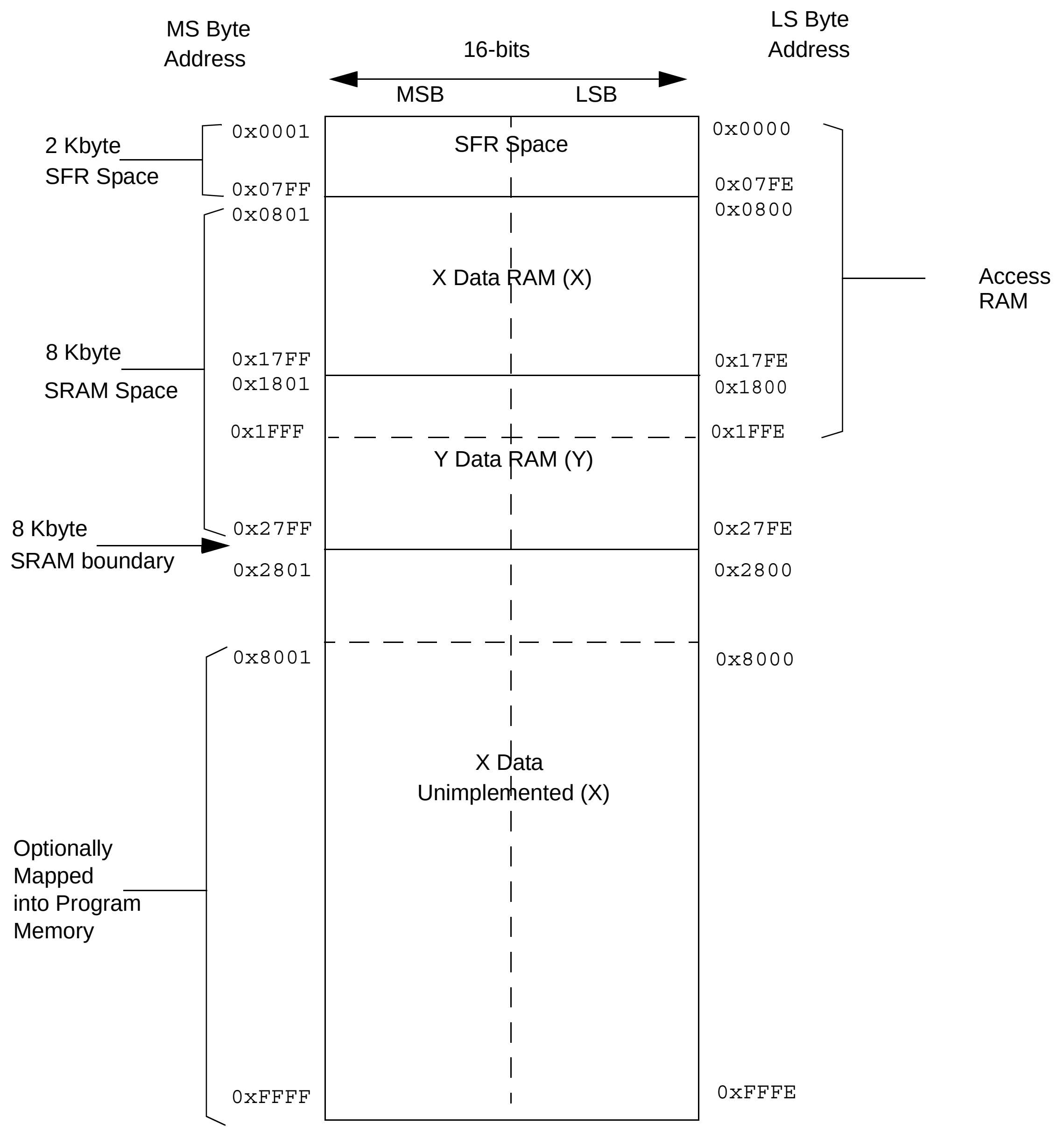
When executing any instruction other than one of the `MAC` class of instructions, the X block consists of the entire 64 Kbyte data address space (including all Y addresses). When executing a `MAC` class of instruction, the X block consists of the entire 64 Kbyte data address space, less the Y address block for data reads (only). In other words, all other instructions regard the entire data memory as one composite address space. The `MAC` class of instructions extracts the Y address space from data space and addresses 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 by the `MAC` class of instruction.

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

3.2.5 ACCESS RAM SPACE

An 8 Kbyte access 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.

FIGURE 3-6: SAMPLE DATA SPACE MEMORY MAP



Note: The address map shown in Figure 3-6 is conceptual, and may vary across individual devices depending on available memory.

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

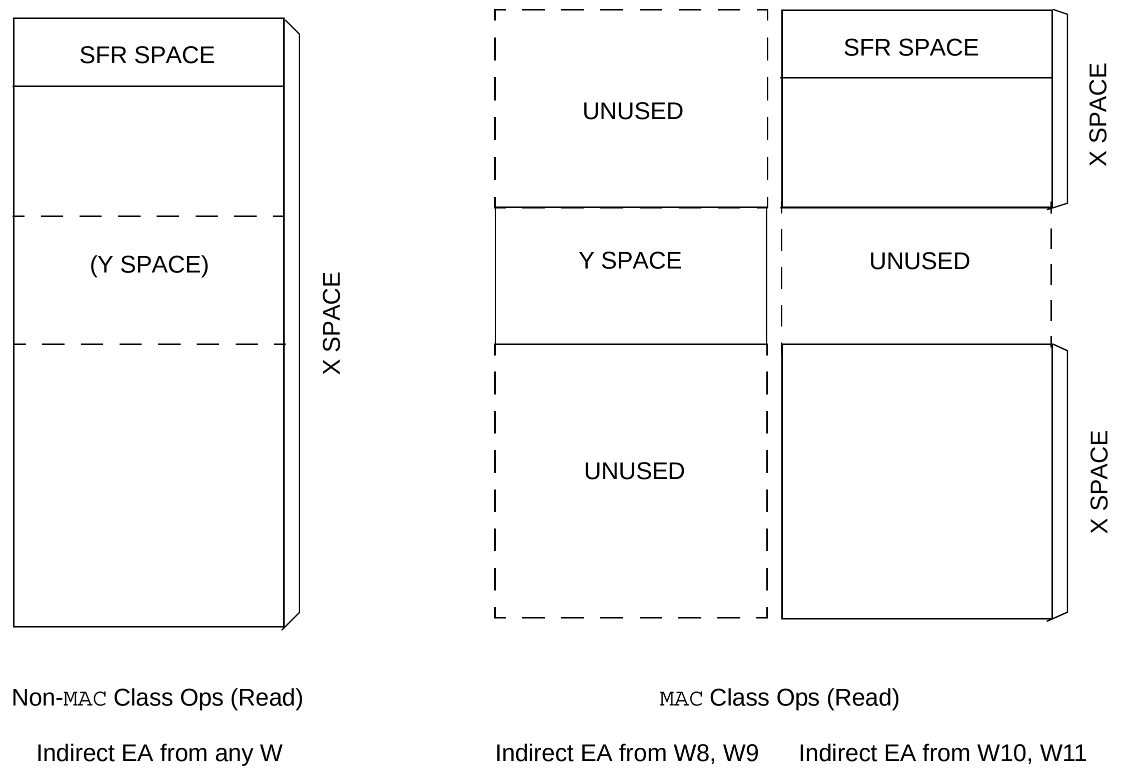


TABLE 3-3: CORE REGISTER MAP

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	W0 / WREG																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	—	—	—	—	—	—	—	—	ACCAU								uuuu uuuu 0000 0000
ACCBL	0028	ACCBL																0000 0000 0000 0000
ACCBH	002A	ACCBH																0000 0000 0000 0000
ACCBU	002C	—	—	—	—	—	—	—	—	ACCBU								uuuu uuuu 0000 0000
PCL	002E	PCL																0000 0000 0000 0000
PCH	0030	—	—	—	—	—	—	—	—	PCH								uuuu uuuu 0000 0000
TBLPAG	0032	—	—	—	—	—	—	—	—	TBLPAG								uuuu uuuu 0000 0000
PSVPAG	0034	—	—	—	—	—	—	—	—	PSVPAG								uuuu uuuu 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								uuuu uuuu 0000 0000
DOENDL	003E	DOENDL																0000 0000 0000 0000
DOENDH	0040	—	—	—	—	—	—	—	—	DOENDH								uuuu uuuu 0000 0000
SR	0042	OA	OB	SA	SB	OAB	SAB	DA	DC	IPL2	IPL1	IPL0	RA	N	OV	SZ	C	0000 0000 0000 0000
CORCON	0044	—	—	—	—	—	—	—	—	SATA	SATB	SATDW	ACCSAT	—	PSV	RND	IF	uuuu uuuu 0000 u000

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
MODCON	0046	XMODEN	YMODEN	—	—	BWM3	BWM2	BWM1	BWM0	YWM3	YWM2	YWM1	YWM0	XWM3	XWM2	XWM1	XWM0	00uu 0000 0000 0000
XMODSRT	0048	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	0000 0000 0000 0000
XMODEND	004A	XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8	XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0	0000 0000 0000 0000
YMODSRT	004C	YS15	YS14	YS13	YS12	YS11	YS10	YS9	YS8	YS7	YS6	YS5	YS4	YS3	YS2	YS1	YS0	0000 0000 0000 0000
YMODEND	004E	YE15	YE14	YE13	YE12	YE11	YE10	YE9	YE8	YE7	YE6	YE5	YE4	YE3	YE2	YE1	YE0	0000 0000 0000 0000
XBREV	0050	BREN	XB14	XB13	XB12	XB11	XB10	XB9	XB8	XB7	XB6	XB5	XB4	XB3	XB2	XB1	XB0	0000 0000 0000 0000
DISICNT	0052	—	—	DISICNT<13:0>														uu00 0000 0000 0000

3.2.6 DATA PRE-FETCH FROM PROGRAM SPACE WITHIN A REPEAT LOOP

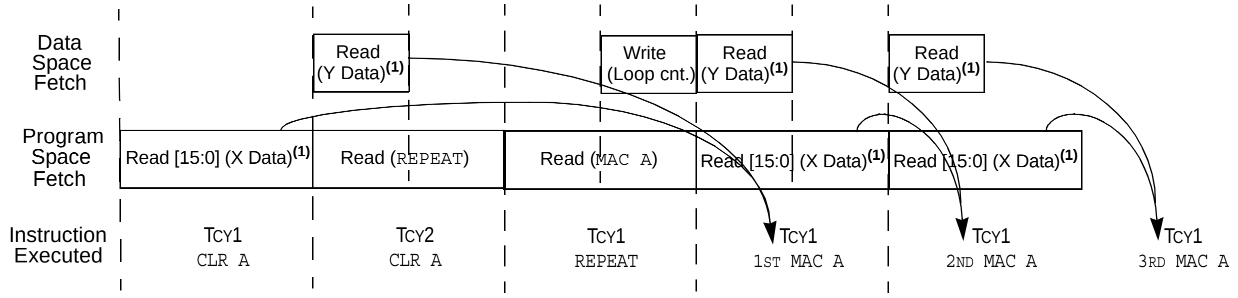
When pre-fetching data resident in program space, via the data space window from within a REPEAT loop, all iterations of the repeated instruction will reload the instruction from the Instruction Latch without re-fetching it, thereby releasing the program bus for a data pre-fetch, as shown in Figure 3-8. In this example, the initial 2 data words for the first iteration of the instruction to be repeated (`MAC A`) are fetched by a prior instruction (e.g., `CLR A` instruction). The subsequent `MAC` instructions, therefore, only need to fetch two more data pairs to complete the loop. The initial fetch of the `MAC A` instruction is performed by the `REPEAT` instruction.

EXAMPLE 3-2: PROGRAM SPACE DATA READ THROUGH DATA SPACE WITHIN A REPEAT LOOP

```
; In this example, data for MAC operations is stored in Program Space.

CLR     A, [W8]+=2, W4, [W10]+=2, W5        ; Acc. A cleared, data prefetched (2 cycles)
REPEAT  #99                                 ; Repeat the MAC operation 100 times (1 cycle)
MAC     W4*W5, A, [W8]+=2, W4, [W10]+=2, W5 ; MAC operation within REPEAT loop (1 cycle)
```

FIGURE 3-8: PROGRAM SPACE DATA READ THROUGH DATA SPACE WITHIN A REPEAT LOOP



Note 1: Program space window will always reside in X space.

NOTES:

4.0 FLASH PROGRAM MEMORY

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

4.1 In-Circuit Serial Programming (ICSP)

The details of ICSP will be provided at a later date.

4.2 Run Time Self-Programming (RTSP)

RTSP is accomplished using `TBLRD` (table read) and `TBLWT` (table write) instructions, and the following control registers:

- **NVMCON**: Non-volatile Memory Control Register
- **NVMKEY**: Non-volatile Memory Key Register
- **NVMADR**: Non-volatile Memory Address Register

With RTSP, the user may erase program memory, 32 instructions (96 bytes) at a time and can write program memory data, 4 instructions (12 bytes) at a time.

4.3 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. Since the program memory is only 24-bits wide, the `TBLRDH` and `TBLWTH` instructions have the ability to address an upper byte of program memory that does not exist. This byte is called the ‘phantom byte’. Any read of the phantom byte will return 0x00 and a write to the phantom byte has no effect (see Figure 4-2).

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. The upper 23 bits of the EA are used to select the program memory location. For the Byte mode table instructions, the LS bit of the W register EA is used to pick which byte of the 16 bit program memory word is addressed. A “1” selects bits <15:8>, a “0” selects bits <7:0>. The LSb of the W register EA is ignored for a table instruction in Word mode.

The table instruction also specifies a W register (or a W register pointer to a memory location) that is the source of the program memory data to be written, or the destination for a program memory read. For a table write operation in Byte mode, bits<15:8> of the source working register are ignored

FIGURE 4-1: ADDRESSING FOR TABLE AND NVM REGISTERS

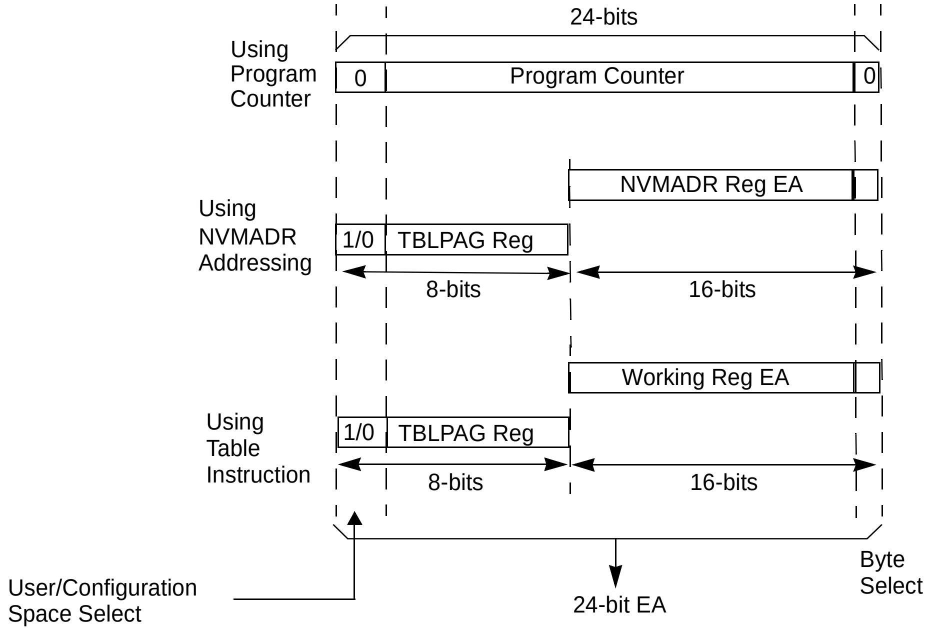
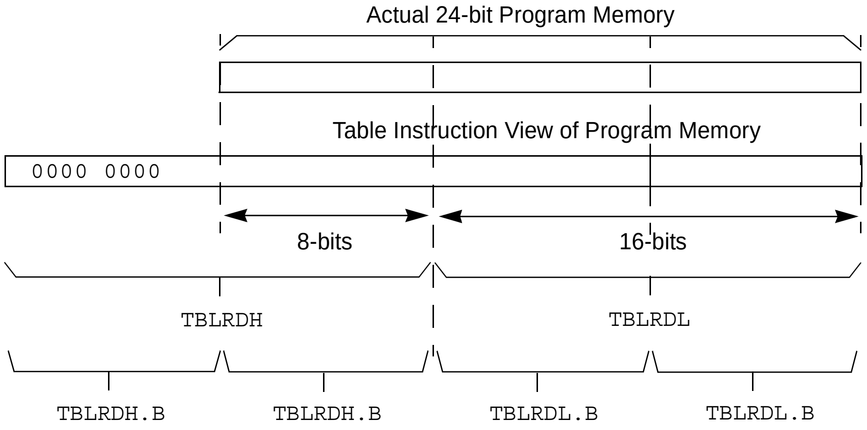


FIGURE 4-2: TABLE READ OPERATIONS ON PROGRAM MEMORY


Actual 24-bit Program Memory
Table Instruction View of Program Memory
0000 0000
8-bits
16-bits
TBLRDH
TBLRDL
TBLRDH.B
TBLRDH.B
TBLRDL.B
TBLRDL.B

4.4 Using Table Read Instructions

4.4.1 READ WORD MODE

The following instructions can be used to read a word of program memory, as shown in Example 4-1.

EXAMPLE 4-1: READING A FLASH PROGRAM MEMORY WORD

```
; setup pointer
          MOV     #tblpage(PROG_ADDR),W0   ;
          MOV     W0,TBLPAG                ; Initialize PM Page Boundary SFR
          MOV     #tbloffset(PROG_ADDR),W0 ; Initialize in-page EA[15:0] pointer
          TBLRDH  [W0],W3                  ; Read PM high word into W3
          TBLRDL  [W0],W4                  ; Read PM low word into W4
```

4.4.2 READ BYTE MODE

The following instructions can be used to read a byte of program memory, as shown in Example 4-2.

The post-increment operator on the read of the low byte causes the address in working register to increment by one. This sets EA<0> to a “1”, for access to the middle byte in the third write instruction. The last post-increment sets W0 back to an even address, pointing to the next program memory location.

EXAMPLE 4-2: READING A FLASH PROGRAM MEMORY BYTE

```
; setup pointer
          MOV      #tblpage(PROG_ADDR),W0  ;
          MOV      W0,TBLPAG               ; Initialize PM Page Boundary SFR
          MOV      #tbloffset(PROG_ADDR),W0; Intialize in-page EA[15:0] pointer
          TBLRDH.b [W0],W3                 ; Read PM high byte
          TBLRDL.b [W0++],W4               ; Read PM low byte
          TBLRDL.b [W0++],W5               ; Read PM middle byte
```

4.5 Using Table Write Instructions

4.5.1 WRITE WORD MODE

To write a single program latch location in Word mode, the following sequence can be implemented.

In Example 4-3, the contents of the upper byte of W3 has no effect. W0 is post-incremented by 2 after the TBLWTH instruction, to prepare for the write to the next program memory location.

EXAMPLE 4-3: WRITING A SINGLE PROGRAM LATCH LOCATION IN WORD MODE

```
; setup pointer
          MOV      #tblpage(PROG_ADDR),W0   ;
          MOV      W0,TBLPAG                ; Initialize PM Page Boundary SFR
          MOV      #tbloffset(PROG_ADDR),W0; Intialize in-page EA[15:0] pointer
; get data into W registers
          MOV      #PROG_LOW_WORD,W2        ; Low PM word into W2
          MOV      #PROG_HI_BYTE,W3         ; High PM byte into W3
; do the table writes
          TBLWTL   W2,[W0]                  ; Write PM low word into program latch
          TBLWTH   W3,[W0++]                ; Write PM high byte into program latch
```

4.5.2 WRITE BYTE MODE

To write a single memory latch location in Byte mode, the following sequence can be implemented.

In Example 4-4, the post-increment operator on the write to the low byte causes the address in W0 to increment by one. This sets EA<0> to a “1”, for access to the middle byte in the third write instruction. The last post-increment sets W0 back to an even address, pointing to the next program memory location.

EXAMPLE 4-4: WRITING A SINGLE PROGRAM LATCH LOCATION IN BYTE MODE

```
; setup pointer
          MOV       #tblpage(PROG_ADDR),W0   ;
          MOV       W0,TBLPAG                ; Initialize PM Page Boundary SFR
          MOV       #tbloffset(PROG_ADDR),W0 ; Intialize in-page EA[15:0] pointer
; Load data into working registers
          MOV       #LOW_BYTE,W2             ; PM Low byte into W2
          MOV       #MID_BYTE,W3             ; PM Middle byte into W3
          MOV       #HIGH_BYTE,W4            ; PM High byte W4
; Write data to 24-bit memory
          TBLWTH.b  W4,[W0]                  ; Write PM high byte into program latch
          TBLWTL.b  W2,[W0++]                ; Write PM low byte into program latch
          TBLWTL.b  W3,[W0++]                ; Write PM middle byte into program latch
```

4.6 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 four instructions at one time. RTSP may be used to program multiple program memory panels, but the table pointer must be changed at each panel boundary.

Each panel of program memory contains write latches that hold four 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 instruction words loaded must always be from a group of four boundary (e.g., loading of instructions 3, 4, 5, 6 is not allowed).

The basic sequence for RTSP programming is to setup 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. Four `TBLWTL` and four `TBLWTH` instructions are required to load the four instructions. To fully program a row of program memory, eight cycles of four `TBLWTL` and four `TBLWTH` are required. If multiple panel programming is required, the table pointer needs to be changed and the next set of multiple write latches written.

All of the table write operations are single word writes (2 instruction cycles), because only the table latches are written. A total of 8 programming passes, each writing 4 instruction words, are required per row. A 128 row panel requires 1024 programming cycles.

The FLASH Program Memory is readable, writable, and erasable during normal operation, over the entire VDD range.

4.7 Control Registers

The three SFRs used to read and write the program FLASH memory are:

- NVMCON
- NVMADR
- NVMKEY

4.7.1 NVMCON REGISTER

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

4.7.2 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.

4.7.3 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 write 0x55, then 0xAA, to the NVMKEY register. Refer to Section 4.8 for further details.

4.8 Programming Operations

A long write operation is necessary for programming or erasing the internal FLASH in RTSP mode. A long write 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.

4.8.1 PROGRAMMING ALGORITHM FOR PROGRAM FLASH

The user can erase program FLASH memory by rows. The user can program FLASH in blocks of 4 instruction words. 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. Setup NVMCON register for multi-word, program FLASH, erase, and set WREN bit.
    b. Write address of row to be erased into NVMADR.
    c. Disable interrupts.
    d. Write ‘55’ to NVMKEY.
    e. Write ‘AA’ to NVMKEY.
    f. Set the WR bit. This will begin erase cycle.
    g. CPU will stall for the duration of the erase cycle.
    h. The WR bit is cleared when erase cycle ends.
    i. Re-enable interrupts.
4. Write four instruction words of data from data RAM into the program FLASH write latches.
5. Program 4 instruction words into program FLASH.
    a. Setup NVMCON register for multi-word, program FLASH, program, and set WREN bit.
    b. Disable interrupts.
    c. Write ‘55’ to NVMKEY.
    d. Write ‘AA’ to NVMKEY.
    e. Set the WR bit. This will begin program cycle.
    f. CPU will stall for duration of the program cycle.
    g. The WR bit is cleared by the hardware when program cycle ends.
    h. Re-enable interrupts.
6. Repeat steps (4 - 5) seven more times to finish programming FLASH row.
7. Repeat steps 1 through 6 as needed to program desired amount of program FLASH memory.

4.8.2 ERASING A ROW OF PROGRAM MEMORY

The following is a code sequence that can be used to erase a row (32 instructions) of program memory.

EXAMPLE 4-5: ERASING A ROW OF PROGRAM MEMORY

```
; Setup NVMCON for erase operation, multi word write
; program memory selected, and writes enabled
          MOV    #0x4003,W0                 ;
          MOV    W0,NVMCON                  ; Init NVMCON SFR
; Init pointer to row to be ERASED
          MOV    #tblpag(PROG_ADDR),W0      ;
          MOV    W0,TBLPAG                  ; Initialize PM Page Boundary SFR
          MOV    #tbloffset(PROG_ADDR),W0   ; Intialize in-page EA[15:0] pointer
          MOV    W0, NVMADR                 ; Intialize NVMADR SFR
          BCLR.b INTCON1+1, #GIE            ; Disable interrupts (if required)
          MOV    #0x55,W1                   ;
          MOV    W1,NVMKEY                  ; Write the 0x55 key
          MOV    #0xAA,W1                   ;
          MOV    W1,NVMKEY                  ; Write the 0xAA key
          BSET.b NVMCON+1,#WR               ; Start the erase sequence
          NOP                               ; Insert two NOPs after the erase
          NOP                               ; command is asserted
          BSET.b INTCON1+1, #GIE            ; Re-enable interrupts (if required)
```

4.8.3 LOADING WRITE LATCHES

The following is a sequence of instructions that can be used to load the 96-bits of write latches. Four TBLWTL and four TBLWTH instructions are needed to load the write latches selected by the table pointer.

EXAMPLE 4-6: 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
; 3rd_program_word
          MOV    #LOW_WORD_3,W2             ;
          MOV    #HIGH_BYTE_3,W3            ;
          TBLWTL W2, [W0]                   ; Write PM low word into program latch
          TBLWTH W3, [W0++]                 ; Write PM high byte into program latch
```

Note: In example 4-6, the contents of the upper byte of W3 has no effect.

4.8.4 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. If interrupts are implemented, they must be disabled prior to the programming sequence.

EXAMPLE 4-7: INITIATING A PROGRAMMING SEQUENCE

```
BCLR.b INTCON1+1, #GIE        ; Disable interrupts (if required)
MOV    #0x55,W1               ;
MOV    W1,NVMKEY              ; Write the 0x55 Key
MOV    #0xAA,W1               ;
MOV    W1,NVMKEY              ; Write the 0xAA Key
BSET.b NVMCON+1,#WR           ; Start the erase sequence
NOP                           ; Insert two NOPs after the erase
NOP                           ; command is asserted
BSET.b INTCON1+1, #GIE        ; Re-enable interrupts (if required)
```

REGISTER 4-1: NVMCON REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
WR	WREN	WRERR	—	—	—	—	—
bit 15							bit 8

Lower Half:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	MEMSEL1	MEMSEL0	SIZSEL1	SIZSEL0	ERASE
bit 7							bit 0

bit 15 **WR**: Write (Program or Erase) Control bit
1 = Initiates a data FLASH or program FLASH erase or write cycle
(the WR bit can be set (not cleared) in software)
0 = Write cycle to the FLASH is complete

bit 14 **WREN**: FLASH Write (Erase or Program) Enable bit
1 = Allow an erase or program operation
0 = No operation allowed

bit 13 **WRERR**: FLASH Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{\text{MCLR}}$ or WDT Reset during programming operation)
0 = The write operation completed successfully

bit 12-5 **Unimplemented:** User code should write 0's to these locations

bit 4-3 **MEMSEL<1:0>:** Memory Array Select for Program or Erase Operation bits
11 = Program, Data and Configuration selected (select for Chip Erase)
10 = Configuration bits array selected
01 = Data FLASH memory array selected
00 = Program FLASH memory array selected (also data FLASH if erasing segments)

bit 2-1 **SIZSEL<1:0>:** Select Memory Unit Size for Program or Erase Operation bits
11 = All memory panels of array (select for Chip Erase)
10 = Reserved, do not use
01 = Multi-words of memory (Row or 4 I-Words)
00 = Single word of memory (Data FLASH only)

bit 0 **ERASE**: Operation Select bit
1 = Perform erase operation
0 = Perform program operation

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

REGISTER 4-2: NVMKEY REGISTER

Upper Half:

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Half:

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
NVMKEY7	NVMKEY6	NVMKEY5	NVMKEY4	NVMKEY3	NVMKEY2	NVMKEY1	NVMKEY0
bit 7							bit 0

bit 15-8 **Unimplemented:** Read as '0'

bit 7-0 **NVMKEY<7:0>:** Key Register (Write Only) bits

Legend:

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

REGISTER 4-3: NVMADR REGISTER

Upper Half:

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADR15	NVMADR14	NVMADR13	NVMADR12	NVMADR11	NVMADR10	NVMADR9	NVMADR8
bit 15							bit 8

Lower Half:

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADR7	NVMADR6	NVMADR5	NVMADR4	NVMADR3	NVMADR2	NVMADR1	NVMADR0
bit 7							bit 0

bit 15-0 **NVMADR<15:0>:** NV Memory Write Address bits
Selects the location to program in program or data FLASH memory.
This register may be read or written to by user. This register will contain the address of EA<15:0> of the last table write instruction executed, until written to by the user.

Legend:

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

NOTES:

5.0 DATA EEPROM MEMORY

The Data EEPROM Memory is readable and writable during normal operation over the entire V_{DD} range. The data memory is directly mapped in the program memory data space.

The three SFRs used to read and write the program FLASH memory are used to access data EEPROM memory, as well. As described in Section 4.0, these registers are:

- NVMCON
- NVMADR
- 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 TBLPAG register, are used to address the EEPROM location being accessed. `TBLRDL` and `TBLWRL` instructions are used to read and write data EEPROM. The dsPIC30F devices have up to 8 Kbytes (4K words) of data EEPROM, with an address range from 0x7FF0000 to 0x7FFFFFE.

A word write operation should be preceded by an erase of the location. The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature, and is typically 2 ms.

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. They are 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 $\overline{\text{MCLR}}$ 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.

5.1 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 5-1.

EXAMPLE 5-1: DATA EEPROM READ

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

5.2 Erasing Data EEPROM

5.2.1 ERASING ENTIRE DATA EEPROM

The following example shows a bulk erase of data EEPROM.

EXAMPLE 5-2: DATA EEPROM BULK ERASE

```
; Select all data EEPROM, set ERASE, WREN bits
  MOV    #0x400F,W0                   ;
  MOV    W0,NVMCON                    ; Initialize NVMCON SFR
  BCLR.b INTCON1+1, #GIE              ; Disable Interrupts (if-required)
; Start erase cycle by setting WR after writing key sequence
  MOV    #0x55,W1
  MOV    W1,NVMKEY                    ; Write the 0x55 key
  MOV    #0xAA,W1
  MOV    W1, NVMKEY                   ; Write the 0xAA key
  BSET.b NVMCON+1,#WR                 ; Initiate erase sequence
  BSET.b INTCON1+1, #GIE              ; Re-enable Interrupts (if-required)
; 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
```

5.2.2 ERASING A BLOCK OF DATA EEPROM

The TBLPAG and NVMADR registers must point to the block. Select erase a block of data FLASH, and set the ERASE and WREN bits in NVMCON register. Setting the WR bit initiates the erase, as shown in Example 5-3.

EXAMPLE 5-3: DATA EEPROM BLOCK ERASE

```
; Select data EEPROM block, ERASE, WREN bits
    MOV        #400B,W0
    MOV        W0,NVMCON             ; Initialize NVMCON SFR
    BCLR.b     INTCON1+1, #GIE       ; Disable Interrupts (if-required)
; Start erase cycle by setting WR after writing key sequence
    MOV        #0x55,W1
    MOV        W1,NVMKEY             ; Write the 0x55 key
    MOV        #0xAA,W1
    MOV        W1,NVMKEY             ; Write the 0xAA key
    BSET.b     NVMCON+1,#WR          ; Initiate erase sequence
    BSET.b     INTCON1+1, #GIE       ; Re-enable Interrupts (if-required)
; 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
```

5.2.3 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 ERASE and WREN bits in NVMCON register. Setting the WR bit initiates the erase, as shown in Example 5-4.

EXAMPLE 5-4: DATA EEPROM WORD ERASE

```
; Select data EEPROM word, ERASE,  WREN bits
    MOV     #4009,W0
    MOV     W0,NVMCON
    BCLR.b  INTCON1+1, #GIE                 ; Disable Interrupts (if-required)
; Start erase cycle by setting WR after writing key sequence
    MOV     #0x55,W1
    MOV     W1,NVMKEY                       ; Write the 0x55 key
    MOV     #0xAA,W1
    MOV     W1,NVMKEY                       ; Write the 0xAA key
    BSET.b  NVMCON+1,#WR                    ; Initiate erase sequence
    BSET.b  INTCON1+1, #GIE                 ; Re-enable Interrupts (if-required)
; 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
```

5.3 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 NVMADR.
   c. Optionally, enable NVM interrupt.
   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. Optionally, enable NVM write done interrupt.
   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 Non-Volatile 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.

5.3.1 WRITING A WORD OF DATA EEPROM

Assuming the user has erased the word to be programmed, then, use a table write instruction to write one write latch, as shown in Example 5-5.

EXAMPLE 5-5: 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
    BCLR.b    INTCON1+1, #GIE                ; Disable Interrupts (if-required)
; Operate key to allow write operation
    MOV       #0x55,W1
    MOV       W1,NVMKEY                      ; Write the 0x55 key
    MOV       #0xAA,W1
    MOV       W1,NVMKEY                      ; Write the 0xAA key
    BSET.b    NVMCON+1,#WR                   ; Initiate program sequence
    BSET.b    INTCON1+1, #GIE                ; Re-enable Interrupts (if-required)
; 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
```

5.3.2 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 5-6: 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
    BCLR.b   INTCON1+1, #GIE    ; Disable Interrupts (If required)
    MOV      #0x55,W1
    MOV      W1,NVMKEY          ; Write the 0x55 key
    MOV      #0xAA,W1
    MOV      W1,NVMKEY          ; Write the 0xAA key
    BSET.b   NVMCON+1, #WR      ; Start write cycle
    BSET.b   INTCON1+1, #GIE    ; Re-enable Interrupts (If required)
```

5.4 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.

Generally, a write failure will be a bit which was written as a ’1’, but reads back as a ’0’ (due to leakage off the cell).

5.5 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.

6.0 DSP ENGINE

Concurrent operation of the DSP engine with MCU instruction flow is not possible, though both the MCU ALU and DSP engine resources may be used concurrently by the same instruction (e.g., `ED` and `EDAC` instructions).

The DSP engine consists of a high speed 16-bit x 16-bit multiplier, a barrel shifter and a 40-bit adder/ subtractor with two target accumulators, round and saturation logic.

Data input to the DSP engine is derived from one of the following:

1. Directly from the W array (registers W4, W5, W6 or W7) via the X and Y data buses for the `MAC` class of instructions (`MAC, MSC, MPY, MPY.N, ED, EDAC, CLR` and `MOVSAC`).
2. From the X bus for all other DSP instructions.
3. From the X bus for all MCU instructions which use the barrel shifter.

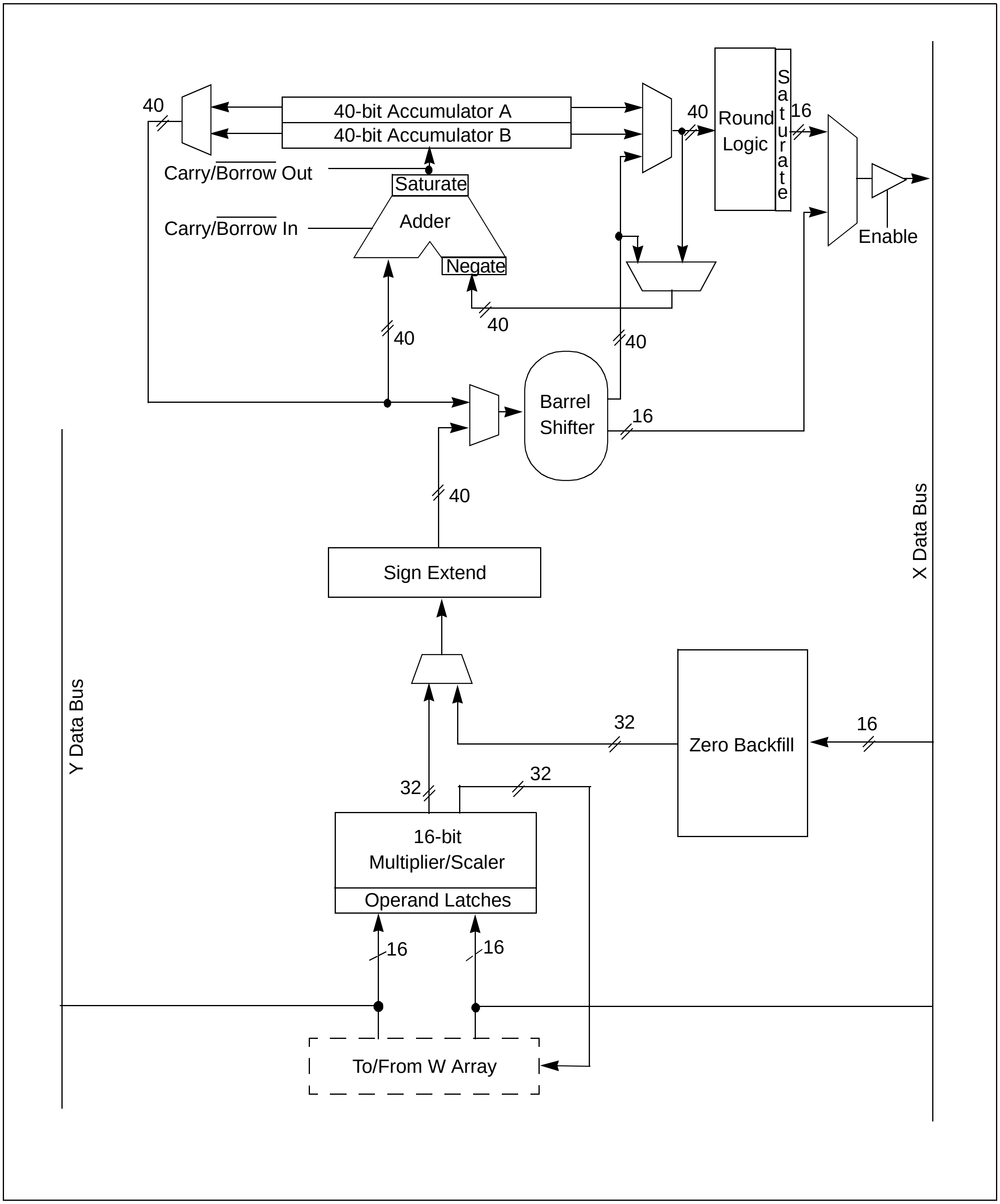
Data output from the DSP engine is written to one of the following:

1. The target accumulator, as defined by the DSP instruction being executed.
2. The X bus for `MAC, MSC, CLR` and `MOVSAC` accumulator writes, where the EA is derived from W13 only. (`MPY, MPY.N, ED` and `EDAC` do not offer an accumulator write option.)
3. The X bus for all MCU instructions which use the barrel shifter.

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`.

A block diagram of the DSP engine is shown in Figure 6-1.

FIGURE 6-1: DSP ENGINE BLOCK DIAGRAM



6.1 Multiplier

The 16 x 16-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. The respective number representation formats are shown in Figure 6-2. 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 (see Figure 6-2). For a 32-bit integer, the data range is –2,147,483,648 (0x8000 0000) to 2,147,483,645 (0x7FFF FFFF).

When the dsPIC30F controller is in Fractional mode, 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 ($1\text{-}2^{1-N}$). For a 16-bit fraction, the Q15 data range is –1.0 (0x8000) to 0.999969482 (0x7FFF), including 0 and has a precision of 3.01518x10^{-5}. In Fractional mode, the 16x16 dsPIC30F multiplier generates a 1.31 product, which has a precision of 4.65661x10^{-10}.

FIGURE 6-2: 16-BIT INTEGER AND FRACTIONAL MODES

Different representations of 0x4001

Integer:

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2^0	2^{14}	2^{13}	2^{12}	2^{11}	. . . .										2^0

0x4001 = $2^{14} + 2^0$ = 16385

1.15 Fractional:

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
-2^0 .	2^{-1}	2^{-2}	2^{-3}	. . .											2^{-15}

0x4001 = $2^{-1} + 2^{-15}$ = 0.500030518

`MAC/MSC`, `MPY[.N]` and `ED[AC]` operations are always signed. The 40-bit adder/subtractor may also optionally negate one of its operand inputs to change the result sign (without changing the operands). This is used to create a multiply and subtract (MSC) or multiply and negate (MPY.N) operation.

In the special case when both input operands are 1.15 fractions and equal to 0x8000 (-1_{10}), the result of the multiplication is corrected to 0x7FFFFFFF (as the closest approximation to +1) by hardware, before it is used.

It should be noted that with the exception of DSP multiplies, the dsPIC30F ALU operates identically on integer and fractional data. Namely, an addition of two integers will yield the same result (binary number) as the addition of two fractional numbers. The only difference is how the result is interpreted by the user. However, multiplies performed by DSP operations are different. In these instructions, data format selection is made by the IF bit (CORCON<0>), and it must be set accordingly ('0' for Fractional mode, '1' for Integer mode). This is required because of the implied radix point used by dsPIC30F fractions. In Integer mode, multiplying two 16-bit integers produces a 32-bit integer result. However, multiplying two 1.15 values generates a 2.30 result. Since the dsPIC30F uses 1.31 format for the accumulators, a DSP multiply in Fractional mode also includes a left shift by one bit to keep the radix point properly aligned. This feature reduces the resolution of the DSP multiplier to 2^{-30}, but has no other effect on the computation.

The same multiplier is used to support the MCU multiply instructions, which include integer 16-bit signed, unsigned and mixed sign multiplies. Additional data paths are provided to allow these instructions to write the result back into the W array and X data bus (via the W array). These paths are placed prior to the data scaler. The IF bit in the CORCON register, therefore, only affects the result of the `MAC` and `MPY` instructions. All other multiply operations are assumed to be integer operations. If the user executes a `MAC` or `MPY` instruction on fractional data, without clearing the IF bit, the result must be explicitly shifted left by the user program after multiplication, in order to obtain the correct result.

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

6.2 Data Accumulators and Adder/ Subtractor

The data accumulator consists of a 40-bit adder/ subractor 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.

6.2.1 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/$\overline{\text{borrow}}$ input is active high and the other input is true data (not complemented), whereas in the case of subtraction, the carry/$\overline{\text{borrow}}$ 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 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 to saturate 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). The OA and OB bits can also optionally generate an arithmetic warning trap when set and the corresponding overflow trap flag enable bit (OVATEN, OVBTEN) in the INTCON1 register (see Section 8.0) is set. This allows the user to take immediate action, for example, to correct system gain.

The SA and SB bits can be set 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 will be saturated (if saturation is enabled). When saturation is not enabled, the 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 will generate an arithmetic warning trap when saturation is disabled.

The overflow and saturation status bits can optionally be viewed in the STATUS register 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:
   Using the bit 39 overflow status bit from the adder, and the bit 39 value after the addition, the correct sign of the 9.31 result can be determined. The saturate logic then loads the maximally positive 9.31 (0x7FFFFFFFFF) or maximally negative 9.31 value (0x8000000000) 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:
   Using the bit 31 to 39 overflow status bit from the adder, and the bit 39 value after the addition, the correct sign of the required 1.31 result can be determined. The saturate logic then loads the maximally positive 1.31 value (0x007FFFFFFF) or maximally negative 1.31 value (0x0080000000) 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.

6.2.2 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).

6.2.3 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 (see Figure 6-3). If rounding is not indicated by the instruction, a truncated 1.15 data value is stored and the LS Word is simply discarded.

The two Rounding modes are shown in Figure 7-7. Conventional rounding takes bit 15 of the accumulator, zero extends it and adds it to the ACCH word (bits 16 through 31 of the accumulator). If the ACCL word (bits 0 through 15 of the accumulator) is between 0x8000 and 0xFFFF (0x8000 included), ACCH is incremented. If ACCL is between 0x0000 and 0x7FFF, ACCH is left unchanged. A consequence of this algorithm is that over a succession of random rounding operations, the value will tend to be biased slightly positive.

Convergent (or unbiased) rounding operates in the same manner as conventional rounding, except when ACCL equals 0x8000. If this is the case, the LS bit (bit 16 of the accumulator) of ACCH is examined. If it is 1, ACCH is incremented. If it is 0, ACCH is not modified. Assuming that bit 16 is effectively random in nature, this scheme will remove 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 6.2.4). Note that for the `MAC` class of instructions, the accumulator write back operation will function in the same manner, addressing combined MCU (X and Y) data space though the X bus. For this class of instructions, the data is always subject to rounding.

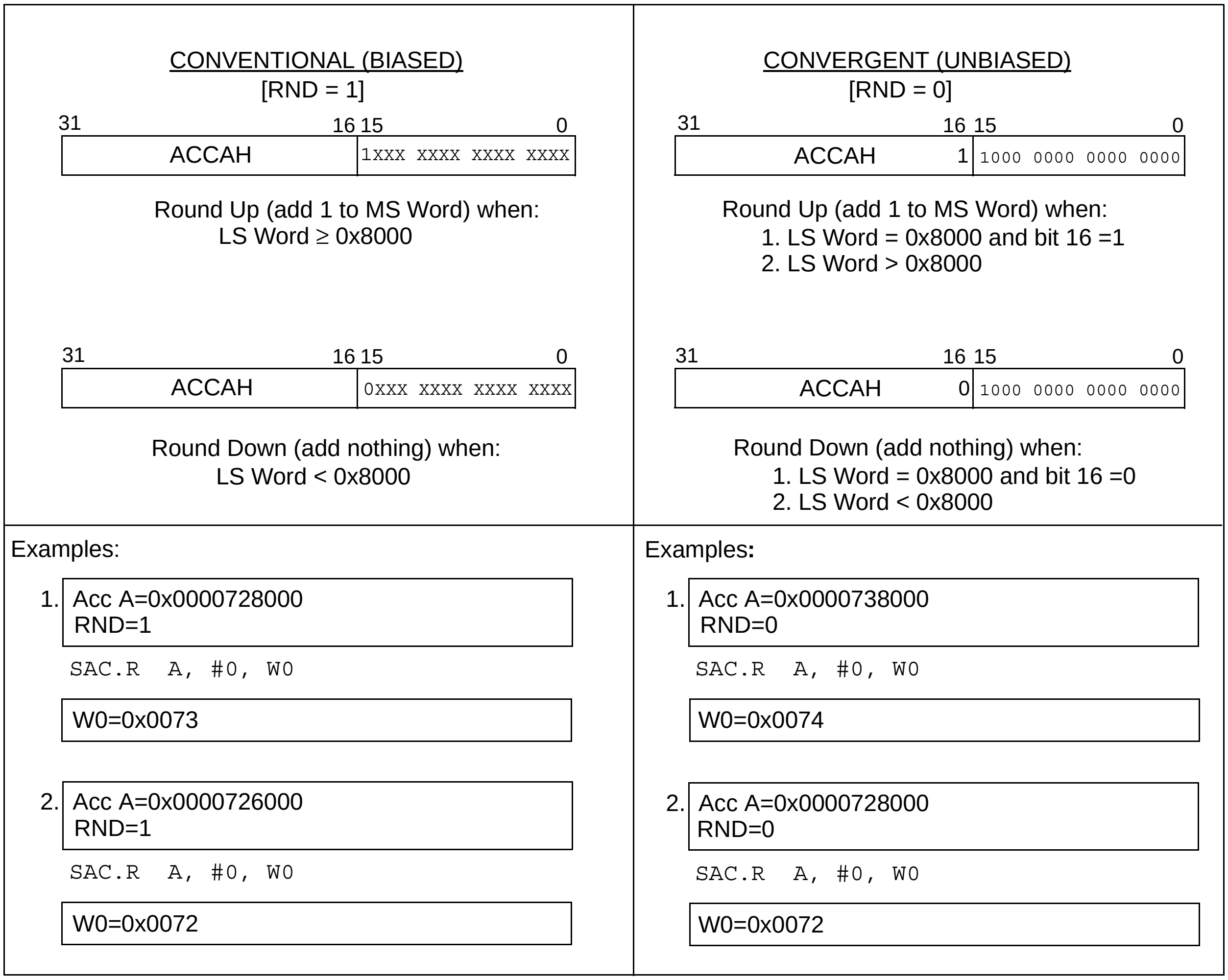
6.2.4 DATA SPACE WRITE SATURATION

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 MS bit 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. All data writes from the DSP Engine into data space, may be optionally saturated by setting the SATDW bit.

FIGURE 6-3: CONVENTIONAL AND CONVERGENT ROUNDING MODES



CONVENTIONAL (BIASED)
[RND = 1]

Round Up (add 1 to MS Word) when:
LS Word ≥ 0x8000

Round Down (add nothing) when:
LS Word < 0x8000

CONVERGENT (UNBIASED)
[RND = 0]

Round Up (add 1 to MS Word) when:
1. LS Word = 0x8000 and bit 16 =1
2. LS Word > 0x8000

Round Down (add nothing) when:
1. LS Word = 0x8000 and bit 16 =0
2. LS Word < 0x8000

Examples:

1. Acc A=0x0000728000
RND=1

```
SAC.R  A, #0, W0
```

W0=0x0073

2. Acc A=0x0000726000
RND=1

```
SAC.R  A, #0, W0
```

W0=0x0072

Examples:

1. Acc A=0x0000738000
RND=0

```
SAC.R  A, #0, W0
```

W0=0x0074

2. Acc A=0x0000728000
RND=0

```
SAC.R  A, #0, W0
```

W0=0x0072

6.3 Barrel Shifter

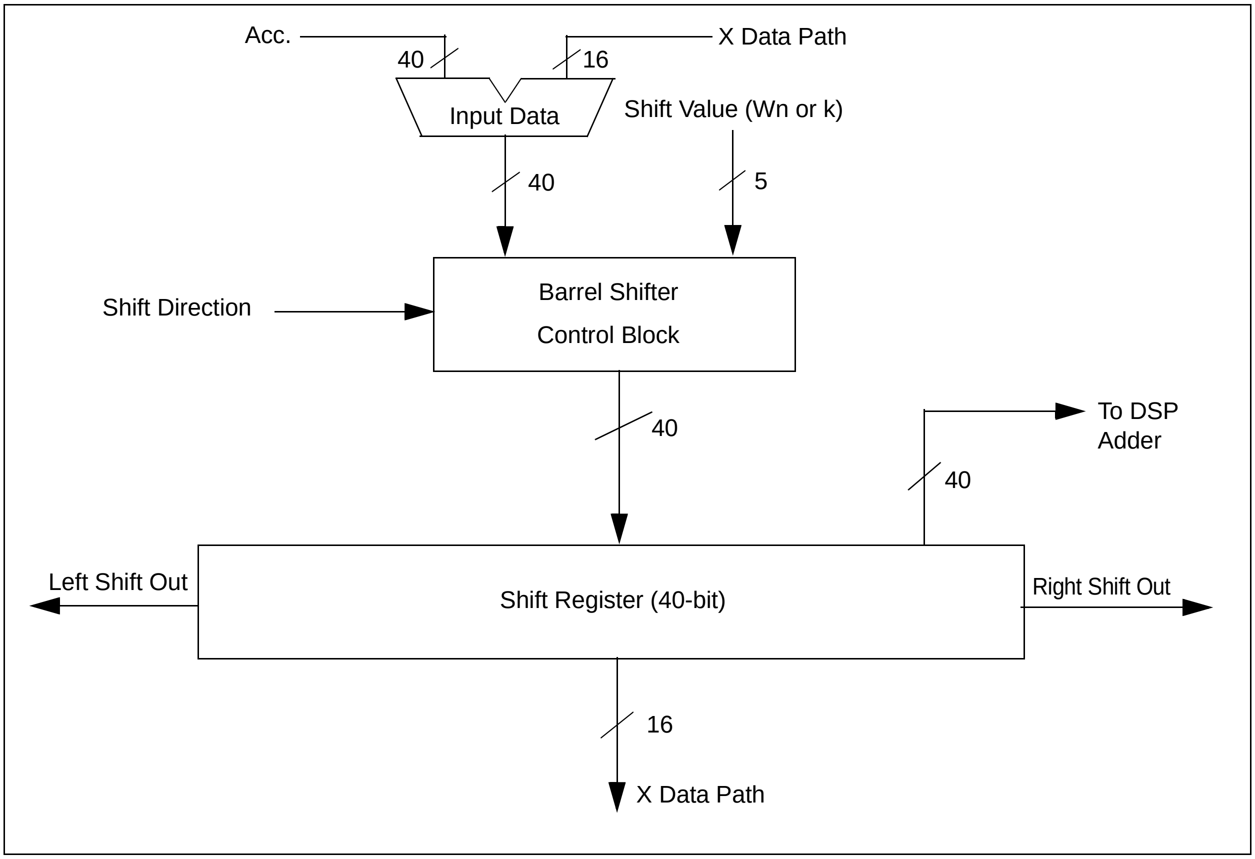
The barrel shifter is capable of performing up to 15-bit arithmetic or logic right shifts, or up to 16-bit left shifts in a single cycle. The source can be either of the two DSP accumulators, or the X bus (to support multi-bit shifts of register or memory data).

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

The barrel shifter is 40 bits wide to accommodate the width of the accumulators. The barrel shifter is used for both DSP and MCU shift operations. A 40-bit result is obtained for DSP shift operations, and a 16-bit result is obtained 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.

The block diagram of the barrel shifter sub-system is shown in Figure 6-4.

FIGURE 6-4: BARREL SHIFTER SIMPLIFIED BLOCK DIAGRAM



6.4 DSP Engine Mode Selection

The DSP engine has various options selected through the CPU Core Configuration Register (CORCON), as listed below:

1. Fractional or integer.
2. Conventional or convergent rounding.
3. Automatic saturation on/off for AccA.
4. Automatic saturation on/off for AccB.
5. Automatic saturation on/off for writes to data memory.
6. Accumulator Saturation mode selection.

REGISTER 6-1: CORCON CPU MODE CONTROL REGISTER

Upper Half:

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
SATA	SATB	SATDW	ACCSAT	—	PSV	RND	IF
bit 7							bit 0

bit 15-8 **Unimplemented:** Read as '0’

bit 7 **SATA:** AccA Saturation Enable bit
1 = Accumulator A saturation enabled
0 = Accumulator A saturation disabled

bit 6 **SATB:** AccB Saturation Enable bit
1 = Accumulator B saturation enabled
0 = Accumulator B saturation disabled

bit 5 **SATDW:** Data Space Write from DSP Engine Saturation Enable bit
1 = Data space write saturation enabled
0 = Data space write saturation disabled

bit 4 **ACCSAT:** Accumulator Saturation Mode Select bit
1 = 9.31 saturation (super saturation)
0 = 1.31 saturation (normal saturation)

bit 3 **Unimplemented:** Read as ’0’

bit 2 **PSV:** Program Space Visibility in Data Space Enable bit
1 = Program Space Visible in Data Space
0 = Program Space Not Visible in Data Space

bit 1 **RND:** Rounding Mode Select bit
1 = Biased (conventional) rounding enabled
0 = Unbiased (convergent) rounding enabled

bit 0 **IF:** Integer or Fractional Multiplier Mode Select bit
1 = Integer mode enabled for DSP ops
0 = Fractional mode enabled for DSP ops

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

7.0 ADDRESS GENERATOR UNITS

The dsPIC core contains three independent address generator units. The X RAGU (Read AGU) and X WAGU (Write AGU) support byte and word sized data space reads and writes, respectively, for MCU and DSP instructions. The Y AGU supports word sized data reads for the DSP MAC class of instructions only. They are each capable of supporting two types of data addressing:

- Linear Addressing
- Modulo (Circular) Addressing

In addition, the X WAGU can support:

- 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.

7.1 Data Space Organization

Although the data space memory is organized as 16-bit words, all effective addresses (EAs) are byte addresses. Instructions can thus, access individual bytes, as well as properly aligned words. Word addresses must be aligned at even boundaries. Misaligned word accesses are not supported, and if attempted, will initiate an address error trap.

When executing instructions which require just one source operand to be fetched from data space, the X RAGU and X WAGU are used to calculate the effective address. The X RAGU and X WAGU can generate any address in the 64 Kbyte data space. They support all MCU Addressing modes and Modulo Addressing for low overhead circular buffers. The X WAGU also supports Bit-Reversed Addressing to facilitate FFT data reorganization.

When executing instructions which require two source operands to be concurrently fetched (i.e., the MAC class of DSP instructions), both the X RAGU and Y AGU are used simultaneously and the data space is split into 2 independent address spaces, X and Y. The Y AGU supports Register Indirect Post-Modified and Modulo Addressing only. Note that the data write phase of the MAC class of instruction does not split X and Y address space. The write EA is calculated using the X WAGU and the data space is configured for full 64 Kbyte access.

In the Split Data Space mode, some W register address pointers are dedicated to X RAGU, and others to Y AGU. The EAs of each operand must, therefore, be restricted to be within different address spaces. If they are not, one of the EAs will be outside the address space of the corresponding data space (and will fetch the bus default value, 0x0000).

7.2 Instruction Addressing Modes

The Addressing modes in Table 7-1 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.

Some Addressing mode combinations may lead to a one-cycle stall during instruction execution, or are not allowed, as discussed in Section 7.3.

TABLE 7-1: 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.

7.2.1 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. These memory locations are known as File Registers. 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.

7.2.2 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 W register, fetched from data memory, or 5-bit literal. In two-operand instructions, the result location is the same as that of one of the operands. Certain MCU instructions are one-operand operations. The following addressing modes are supported by MCU instructions:

- Register Direct
- Register Indirect
- Register Indirect Post-decremented
- Register Indirect Post-incremented
- Register Indirect Pre-decremented
- Register Indirect Pre-incremented
- 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.

7.2.3 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 supported by Move and Accumulator instructions:

- Register Direct
- Register Indirect
- Register Indirect Post-decremented
- Register Indirect Post-incremented
- Register Indirect Pre-decremented
- Register Indirect Pre-incremented
- Register Indirect with Register Offset (Indexed)
- 10-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.

7.2.4 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 2 source operand pre-fetch registers must be a member of the set {W8, W9, W10, W11}. For data reads, W8 and W9 will always be directed to the X RAGU and W10 and W11 will always be directed to the Y AGU. The effective addresses 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 space) and W11 (in Y space).

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

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

7.2.5 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 `NOP`, do not have any operands.

7.3 Instruction Stalls

7.3.1 INTRODUCTION

In order to maximize data space EA calculation and operand fetch time, the X data space read and write accesses are partially pipelined. The latter half of the read phase overlaps the first half of the write phase of an instruction, as shown in Section 2.

Address register data dependencies may arise between successive read and write operations, using common registers.

There are two types of dependencies possible in the dsPIC30F architecture. ‘Read After Write’ (RAW) dependencies occur across instruction boundaries and are detected by the hardware. ‘Write After Read’ (WAR) dependencies occur within instructions and are detected by the assembler, enabling users to modify their program to avoid it.

An example of a RAW dependency is a write operation that modifies W5 (on falling Q4), followed by a read operation that uses W5 as an address pointer (starting in Q3). W5 will not be valid for the read operation until the earlier write completes. This problem is resolved by stalling the instruction execution for one instruction cycle, thereby, allowing the write to complete before the next read is started.

7.3.2 RAW DEPENDENCY DETECTION

During the instruction pre-decode, the core determines if any address register dependency is imminent across an instruction boundary. The stall detection logic compares the W register (if any) used for the destination EA of the instruction currently being executed, with the W register to be used by the source EA (if any) of the pre-fetched instruction. As the W registers are also memory mapped, the stall detection logic also derives an SFR address from the W register being used by the destination EA (see Figure 7-1), and determines whether this address is being issued during the write phase of the instruction currently being executed.

When it observes a match between the destination and source registers, a set of rules are applied to decide whether or not to stall the instruction by one cycle. Table 7-2 lists out the various RAW conditions which cause an instruction execution stall.

FIGURE 7-1: W REGISTER SFR ADDRESS DERIVATION

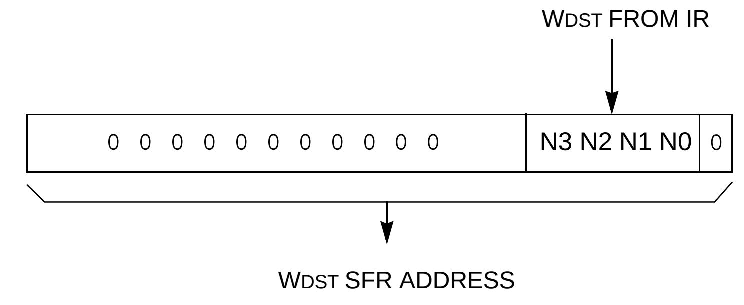


TABLE 7-2: RAW DEPENDENCY RULES (DETECTION BY HARDWARE)

Destination Addressing Mode Using Wn	Source Addressing Mode Using Wn	Status	Examples (Wn = W2)

Direct	Direct	Allowed	`ADD.w  W0, W1, W2`
			`MOV.w  W2, W3`

Direct	Indirect	Stall	`ADD.w  W0, W1, W2`
			`MOV.w  [W2], W3`

Direct	Indirect with Pre- or Post-Modification	Stall	`ADD.w  W0, W1, W2`
			`MOV.w  [W2++], W3`

Indirect	Direct	Allowed	`ADD.w  W0, W1, [W2]`
			`MOV.w  W2, W3`

Indirect	Indirect	Allowed	`ADD.w  W0, W1, [W2]`
			`MOV.w  [W2], W3`

Indirect	Indirect	Stall	`ADD.w  W0, W1, [W2] ; W2=0x0004 (mapped W2)`
			`MOV.w  [W2], W3     ; (i.e. if W2 = addr. of W2)`

Indirect	Indirect with Pre- or Post-Modification	Allowed	`ADD.w  W0, W1, [W2]`
			`MOV.w  [W2++], W3`

Indirect	Indirect with Pre- or Post-Modification	Stall	`ADD.w  W0, W1, [W2] ; W2=0x0004 (mapped W2)`
			`MOV.w  [W2++], W3   ; (i.e. if W2 = addr. of W2)`

Indirect with Pre- or Post-Modification	Direct	Allowed	`ADD.w  W0, W1, [W2++]`
			`MOV.w  W2, W3`

Indirect with Pre- or Post-Modification	Indirect	Stall	`ADD.w  W0, W1, [W2++]`
			`MOV.w  [W2], W3`

Indirect with Pre- or Post-Modification	Indirect with Pre- or Post-Modification	Stall	`ADD.w  W0, W1, [W2++]`
			`MOV.w  [W2++], W3`

7.3.3 INSTRUCTION STALLS AND EXCEPTIONS

In order to maintain deterministic operation, instruction stalls are allowed even if they occur immediately prior to exception processing.

7.3.4 INSTRUCTION STALLS AND FLOW CHANGE INSTRUCTIONS

`CALL[W]` and `RCALL` write to the stack using W15 and may, therefore, be subjected to an instruction stall if the source read of the subsequent instruction uses W15.

`GOTO`, `RETFIE` and `RETURN` instructions are never subjected to an instruction stall, because they only perform read operations.

7.3.5 INSTRUCTION STALLS AND PSV

Instructions operating in PSV address space are subject to instruction stalls just like any other instructions.

Should a data dependency be detected in the instruction immediately following the PSV data access, the second cycle of the instruction will initiate a stall.

Should a data dependency be detected in the instruction immediately before the PSV data access, the last cycle of the previous instruction will initiate a stall.

7.3.6 WAR DEPENDENCY DETECTION

In this architecture, WAR dependencies can only occur within an instruction (i.e., not across instruction boundaries). Therefore, all WAR dependencies are detected by the assembler by disallowing the use of a common W register for both source and destination under certain conditions, as listed in Table 7-3.

TABLE 7-3: WAR DEPENDENCY RULES (DETECTION BY ASSEMBLER)

Source Addressing Mode Using Wn	Destination Addressing Mode Using Wn	Allowed?	Examples (Wn = W1)
Direct	Direct	Yes	`ADD.w W0, W1, W1`
Direct	Indirect	Yes	`ADD.w W0, W1, [W1]`
Direct	Indirect with Pre- or Post-Modification	Yes	`ADD.w W0, W1, [W1++]`
Indirect	Direct	Yes	`ADD.w W0, [W1], W1`
Indirect	Indirect	Yes	`ADD.w W0, [W1], [W1]`
Indirect with Pre- or Post-Modification	Indirect with Pre- or Post-Modification	Yes	`ADD.w W0, [W1], [W1++]`
Indirect with Pre- or Post-Modification	Direct	Yes	`ADD.w W0, [W1++], W1`
Indirect with Pre- or Post-Modification	Indirect	No	`ADD.w W0, [W1++], [W1]`
Indirect with Pre- or Post-Modification	Indirect with Pre- or Post-Modification	No	`ADD.w W0, [W1++], [W1++]`

7.4 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 order to minimize the hardware size for modulo addressing support, certain usage restrictions are imposed which are discussed in detail in Section 7.4.1 and Section 7.4.4. In summary, any particular circular buffer can only be allowed 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 direction is determined from the address offset sign.

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 Bi-Directional mode, i.e., address boundary checks will be performed on both the lower and upper address boundaries.

7.4.1 START AND END ADDRESS

The Modulo Addressing scheme requires that a starting and an end address be specified and loaded into the 16-bit modulo buffer address registers: XMODSRT, XMODEND, YMODSRT, YMODEND.

Note:	The start and end addresses are the first and last byte addresses of the buffer (irrespective of whether it is a word or byte buffer, or an increasing or decreasing buffer).

If the length of an incrementing buffer is greater than M = 2^{N-1}, but not greater than M = 2^{N} bytes, then the last ’N’ bits of the data buffer start address must be zeros. There are no such restrictions on the end address of an incrementing buffer. For example, if the buffer size (modulus value) is chosen to be 100 bytes (0x64), then the buffer start address for an incrementing buffer must contain 7 Least Significant zeros. Valid start addresses may, therefore, be 0xXX00 and 0xXX80, where ‘X’ is any hexadecimal value. Adding the buffer length to this value and subtracting 1 will give the end address to be written into X/YMODEND. For example, if the start address was chosen to be 0x2000, then the X/YMODEND would be set to (0x2000 + 0x0064 - 1) = 0x2063.

Note:	‘Starting address’ refers to the smallest address boundary of the circular buffer. The first access of the buffer may be at any address within the modulus range (see Section 7.4.4).

In the case of a decrementing buffer, the last ‘N’ bits of the data buffer end address must be ones. There are no such restrictions on the start address of a decrementing buffer. For example, if the buffer size (modulus value) is chosen to be 100 bytes (0x64), then the buffer end address for a decrementing buffer must contain 7 Least Significant ones. Valid end addresses may, therefore, be 0xXXFF and 0xXX7F, where ‘X’ is any hexadecimal value. Subtracting the buffer length from this value and adding 1 will give the start address to be written into X/YMODSRT. For example, if the end address was chosen to be 0x207F, then the start address would be (0x207F - 0x0064+1) = 0x201C, which is the first physical address of the buffer.

Note: Y AGU Modulo Addressing EA calculations assume word sized data (LS bit of every EA is always clear), since the Y AGU only supports word accesses.

In incrementing, as well as decrementing modulo buffers, the buffers must start and end on an aligned word. Therefore, XMODSRT must be an even address (LS bit = 0), whereas XMODEND must be an odd address (LS bit = 1).

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).

7.4.2 W ADDRESS REGISTER SELECTION

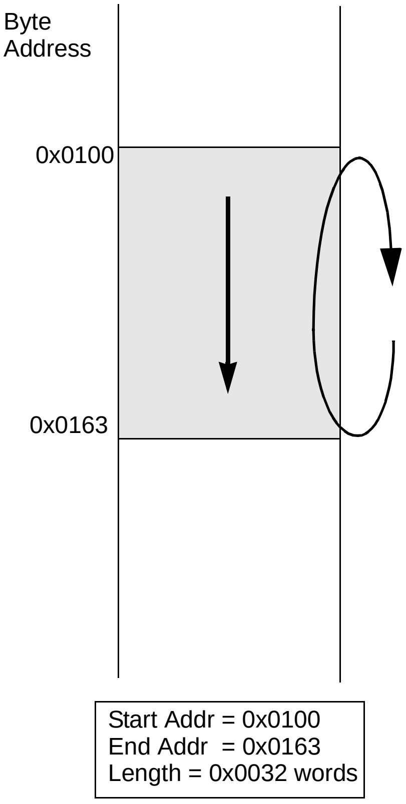
The Modulo and Bit-Reversed Addressing control register MODCON<15:0> contains enable flags plus W register field to specify the W address registers. The XWM and YWM fields select which registers will operate with Modulo Addressing. If XWM = 15, X RAGU and X WAGU Modulo Addressing is disabled. Similarly,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

Note: The XMODSRT and XMODEND registers, and the XWM register selection, are shared between X RAGU and X WAGU.

The X address space pointer W register (XWM) to which Modulo Addressing is to be applied, is stored in MODCON<3:0> (see Register 7-1). 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> (see Register 7-1). 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 7-2: INCREMENTING BUFFER MODULO ADDRESSING OPERATION EXAMPLE



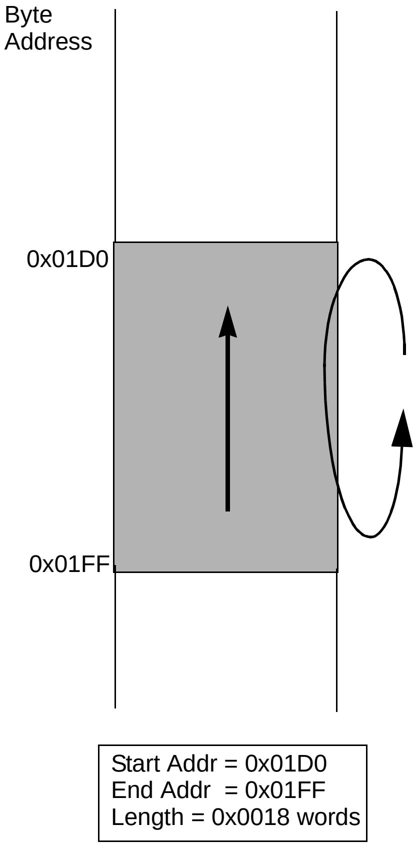
```
MOV     #0x0100,W0
MOV     W0,XMODSRT       ;set modulo start address
MOV     #0x0163,W0
MOV     W0,MODEND        ;set modulo end address
MOV     #0x8001,W0
MOV     W0,MODCON        ;enable W1, X AGU for modulo

MOV     #0x0000,W0       ;W0 holds buffer fill value

MOV     #0x0110,W1       ;point W1 to buffer

DO      AGAIN,#0x31      ;fill the 50 buffer locations
MOV     W0,[W1++]        ;fill the next location
AGAIN: INC    W0,W0      ;increment the fill value
```

FIGURE 7-3: DECREMENTING BUFFER MODULO ADDRESSING OPERATION EXAMPLE



```
MOV    #0x01D0,W0
MOV    #0,XMODSRT      ;set modulo start address
MOV    0x01FF,W0
MOV    W0,XMODEND      ;set modulo end address
MOV    #0x8001,W0
MOV    W0,MODCON       ;enable W1, X AGU for modulo

MOV    #0x000F,W0      ;W0 holds buffer fill value

MOV    #0x01E0,W1      ;point W1 to buffer

DO     AGAIN,#0x17     ;fill the 24 buffer locations
MOV    W0,[W1--]       ;fill the next location
AGAIN: DEC     W0,W0   ; decrement the fill value
```

7.4.3 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 over boundaries and still be adjusted correctly (see Section 7.4.4 for restrictions).

Note:	Modulo Addressing works 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, no address correction is performed and Modulo Addressing fails to produce the desired result.

7.4.4 MODULO ADDRESSING RESTRICTIONS

As stated in Section 7.4.1, for an incrementing buffer the circular buffer start address (lower boundary) is arbitrary, but must be at a ‘zero’ power-of-two boundary. For a decrementing buffer, the circular buffer end address is arbitrary, but must be at a ‘ones’ boundary.

There are no restrictions regarding how much an EA calculation can exceed the address boundary being checked, and still be successfully corrected.

Once configured, the direction of successive addresses into a buffer should not be changed. Although all EAs will continue to be generated correctly irrespective of offset sign, only one address boundary is checked for each type of buffer. Thus, if a buffer is set up to be an incrementing buffer by choosing an appropriate starting address, then correction of the effective address will be performed by the AGU at the upper address boundary, but no address correction will occur if the EA crosses the lower address boundary. Similarly, for a decrementing boundary, address correction will be performed by the AGU at the lower address boundary, but no address correction will take place if the EA crosses the upper address boundary. The circular buffer pointer may be freely modified in both directions without a possibility of out-of-range address access only when the start address satisfies the condition for an incrementing buffer (last ‘N’ bits are zeroes) and the end address satisfies the condition for a decrementing buffer (last ‘N’ bits are ones). Thus, the Modulo Addressing capability is truly bi-directional only for modulo-2 length buffers.

7.5 Bit-Reversed Addressing

Bit-Reversed Addressing is intended to simplify data re-ordering for radix-2 FFT algorithms. It is supported by the X WAGU only, i.e., 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.

7.5.1 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 is Register Indirect with Post-Increment.

XB<14:0> is the Bit-Reversed Address 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 (LS bit of every EA is always clear). The XB value is scaled accordingly to generate compatible (byte) addresses.

When enabled, Bit-Reversed Addressing will only be executed with register indirect with Post-Increment Addressing 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 LS bit 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.

FIGURE 7-4: BIT-REVERSED ADDRESS EXAMPLE

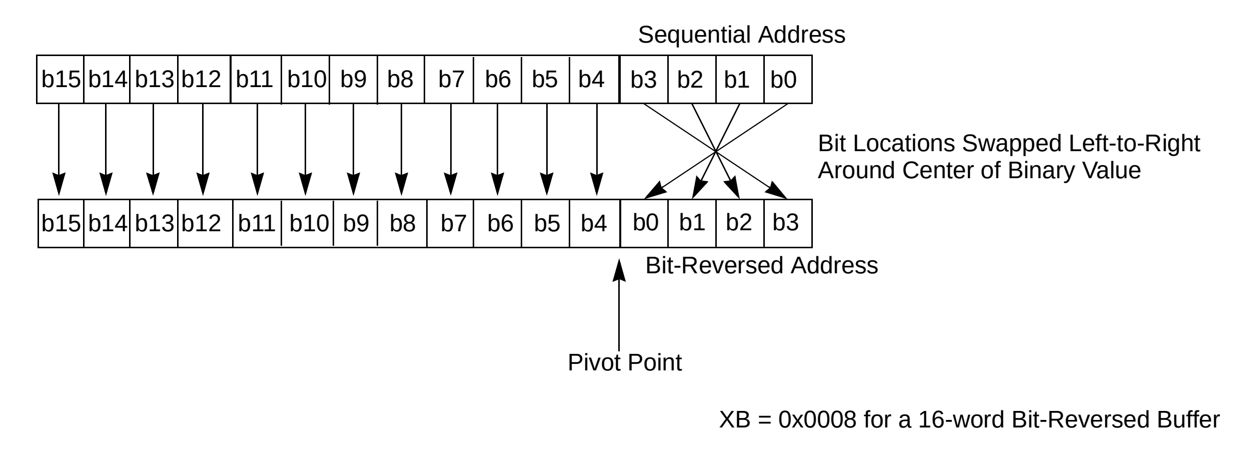


TABLE 7-4: BIT-REVERSED ADDRESS SEQUENCE (16-ENTRY)

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 7-5: BIT-REVERSED ADDRESS MODIFIER VALUES

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

REGISTER 7-1: MODCON, MODULO AND BIT-REVERSED ADDRESSING CONTROL REGISTER

Upper Half:							
R/W-0	R/W-0	U	U	R/W-0	R/W-0	R/W-0	R/W-0
XMODEN	YMODEN	—	—	BWM3	BWM2	BWM1	BWM0
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YWM3	YWM2	YWM1	YWM0	XWM3	XWM2	XWM1	XWM0
bit 7							bit 0

bit 15 **XMODEN:** X RAGU and X WAGU Modulus Addressing Enable bit
`1` = X RAGU and X WAGU Modulus Addressing enabled
`0` = X RAGU and X WAGU Modulus Addressing disabled

bit 14 **YMODEN:** Y AGU Modulus Addressing Enable bit
`1` = Y AGU Modulus Addressing enabled
`0` = Y AGU Modulus Addressing disabled

bit 13-12 **Unimplemented:** Read as '0’

bit 11-8 **BWM<3:0>:** X WAGU W Register Select for Bit-Reversed Addressing bits
`1111` = Bit-Reversed Addressing disabled
`1110` = W14 selected for Bit-Reversed Addressing
| |
`0000` = W0 selected for Bit-Reversed Addressing

bit 7-4 **YWM<3:0>:** Y AGU W Register Select for Modulo Addressing bits
`1111` = Modulo Addressing disabled
`1110` = W14 selected for Modulo Addressing
| |
`0000` = W0 selected for Modulo Addressing

bit 3-0 **XWM<3:0>:** X RAGU and X WAGU W Register Select for Modulo Addressing bits
`1111` = Modulo Addressing disabled
`1110` = W14 selected for Modulo Addressing
| |
`0000` = W0 selected for Modulo Addressing

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

REGISTER 7-2: XMODSRT, X AGU MODULO ADDRESSING START REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0
bit 7							bit 0

bit 15-0 **XS<15:0>:** X RAGU and X WAGU Modulo Addressing Start Address bits

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

REGISTER 7-3: XMODEND, X AGU MODULO ADDRESSING END REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0
bit 7							bit 0

bit 15-0 **XE<15:0>:** X RAGU and X WAGU Modulo Addressing End Address bits

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

REGISTER 7-4: YMODSRT, Y AGU MODULO ADDRESSING START REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YS15	YS14	YS13	YS12	YS11	YS10	YS9	YS8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YS7	YS6	YS5	YS4	YS3	YS2	YS1	YS0
bit 7							bit 0

bit 15-0 **YS<15:0>:** Y AGU Modulo Addressing Start Address bits

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

REGISTER 7-5: YMODEND, Y AGU MODULO ADDRESSING END REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YE15	YE14	YE13	YE12	YE11	YE10	YE9	YE8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
YE7	YE6	YE5	YE4	YE3	YE2	YE1	YE0
bit 7							bit 0

bit 15-0 **YE<15:0>:** Y AGU Modulo Addressing End Address bits

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

REGISTER 7-6: XBREV, X WAGU BIT REVERSAL ADDRESSING CONTROL REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BREN	XB14	XB13	XB12	XB11	XB10	XB9	XB8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
XB7	XB6	XB5	XB4	XB3	XB2	XB1	XB0
bit 7							bit 0

bit 15 **BREN:** Bit-Reversed Addressing (X WAGU) Enable bit
1 = Bit-Reversed Addressing enabled
0 = Bit-Reversed Addressing disabled

bit 14-0 **XB<14:0>:** X WAGU Bit-Reversed Modifier bits
e.g., XB<14:0> = 0x0080; modifier for a 128-point radix-2 FFT

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

NOTES:

8.0 EXCEPTION PROCESSING

The dsPIC30F has 45 interrupt sources and 8 processor exceptions (traps), which must be arbitrated based on a priority scheme.

The processor core 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 Table 8-2.

The interrupt controller is responsible for pre-processing the interrupts, prior to them being presented to the processor core. The peripheral interrupts 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 45 interrupts is held centrally in these twelve registers.
- IPL<2:0> The current CPU priority level is explicitly stored in the 16-bit STATUS register that resides in the processor core.
- INTCON1<15:0>, INTCON2<15:0>
  Global interrupt control functions are derived from these two registers. INTCON1 contains the Global Interrupt Enable (GIE) bit, as well as the control and status flags for the processor exceptions. The INTCON2 register controls the external interrupt request signal behavior and the use of the alternate vector table.

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

All interrupt sources can be user assigned to one of 8 priority levels, 0 through 7, via the IPCx registers. Each interrupt source is associated with an interrupt vector, as shown in Table 8-2. Levels 6 and 0 represent the highest and lowest maskable priorities, respectively. Level 7 interrupts are non-maskable.

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 7 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 Table 8-2). These vectors are contained in locations 0x000004 through 0x0000FE of program memory (refer to Table 8-2). 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 through 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.

8.1 Interrupt Priority

The user assignable Interrupt Priority (IP<2:0>) bits for each individual interrupt source are located in the LS 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”

Table 8-1 lists the interrupt numbers and interrupt sources for the dsPIC device, and their associated vector numbers. The Natural Order Priority of an interrupt is numerically identical to its Vector Number.

> **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 vector number.

The ability for the user to assign every interrupt to one of eight 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 0, thus giving it a very low effective priority.

TABLE 8-1: NATURAL ORDER PRIORITY

INT Number	Vector Number	Interrupt Source
Highest Natural Order Priority		
0	8	INT0 - External Interrupt 0
1	9	IC1 - Input Compare 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	TMR3 - 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	I2C - I^{2}C Transfer Complete
14	22	BCL - I^{2}C Bus Collision
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 - Timer 4
22	30	T5 - Timer 5
23	31	INT2 - External Interrupt 2
24	32	U2RX - UART2 Receiver
25	33	U2TX - UART2 Transmitter
26	34	SPI2
27	35	C1 - Combined IRQ for CAN1
28	36	IC3 - Input Capture 3
29	37	IC4 - Input Capture 4
30	38	IC5 - Input Capture 5
31	39	IC6 - Input Capture 6
32	40	OC5 - Output Compare 5
33	41	OC6 - Output Compare 6
34	42	OC7 - Output Compare 7
35	43	OC8 - Output Compare 8
36	44	INT3 - External Interrupt 3
37	45	INT4 - External Interrupt 4
38	46	C2 - Combined IRQ for CAN2
39	47	PWM - PWM Period Match
40	48	QEI - Position Counter Compare
41	49	DCI - CODEC Transfer Done
42	50	LVD - Low Voltage Detect
43	51	FLTA - Motor Control PWM Fault A
44	52	FLTB - Motor Control PWM Fault B
45- 53	53 - 61	Reserved
Lowest Natural Order Priority		

8.2 RESET Sequence

A RESET is not a true exception, because the interrupt controller is not involved in the RESET process. The processor clears 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.

8.2.1 RESET SOURCES

In addition to external, Power-on Resets (POR) and software RESETS, there are three sources of error conditions which ‘trap’ to the RESET vector.

- Watchdog Time-out:
  The watchdog has timed out, indicating that the processor is no longer executing the correct flow of code.
- 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.

8.3 Traps

Traps can be considered as non-maskable, non-stable interrupts, which adhere to a predefined priority as shown in Table 8-2. 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 RESET vector address. 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 which caused the trap may, therefore, have to be corrected.

8.3.1 TRAP SOURCES

The following traps are provided with increasing priority. However, as all traps are nestable, priority has little effect.

- Software Trap:
  Execution of the TRAP instruction causes an exception.
- Arithmetic Error Trap:
  The Arithmetic Error trap executes under the following three 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, an Arithmetic 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 unutilized.
  3. If enabled, an Arithmetic 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.
- Address Error Trap:
  This trap is initiated when any of the following circumstances occurs:
  1. A misaligned data word fetch is attempted.
  2. A data fetch from unused data address space is attempted.
  3. A program fetch from unimplemented user program address space is attempted.
  4. A program fetch from vector address space is attempted.
  5. A read (for address) of an uninitialized W register is attempted.
- 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).
- Oscillator Fail Trap:
  This trap is initiated if the external oscillator fails and operation becomes reliant on an internal RC backup.

It is possible that multiple traps can become active within the same cycle (e.g., a mis-aligned word stack write to an overflowed address). In such a case, the fixed priority shown in Figure 8-2 is implemented, which may require the user to check if other traps are pending, in order to completely correct the fault.

8.4 Interrupt Sequence

The GIE (Global Interrupt Enable) bit in the INTCON1 register must be set to enable interrupts, but exceptions can be processed, regardless of the state of the GIE bit.

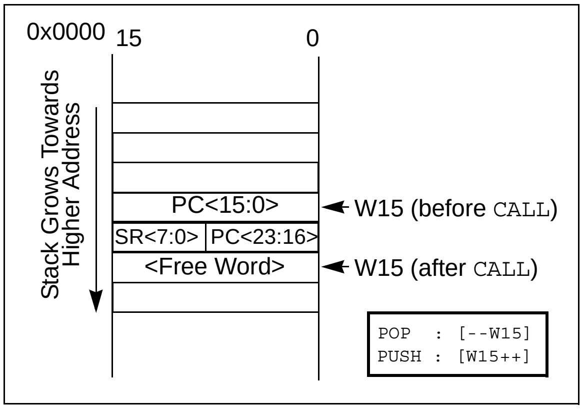
All interrupt event flags are sampled in the beginning of each instruction cycle by the IFSx registers. A pending interrupt request (IRQ) is indicated by the flag bit being equal to a ‘1’ in an IFSx register. The IRQ will cause the interrupt to occur if the corresponding bit in the interrupt enable (IECx) registers 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 equal to or greater than the current processor priority level in the status register, an interrupt will be presented to the processor.

The processor reacts to the interrupt request by asserting the IACK (Interrupt Acknowledge) signal. The GIE bit in the INTCON1 register is cleared. This disables all interrupts until either a `RETFIE` instruction is executed, or the user sets the GIE bit.

The processor then stacks the current program counter and the low byte of the processor STATUS register, as shown in Figure 8-1. The low byte of the STATUS register contains the processor priority level at the time, prior to the beginning of the interrupt cycle.

FIGURE 8-1: INTERRUPT STACK FRAME



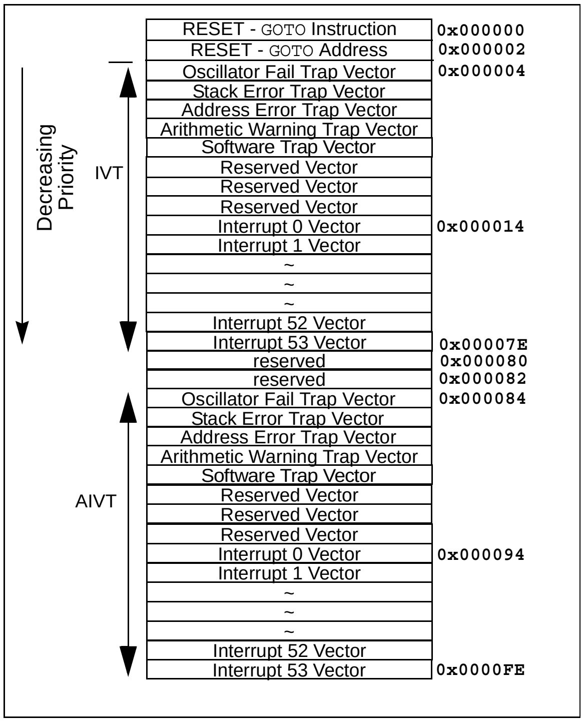
The processor then loads the priority level for this interrupt into the status register. This action will disable all lower priority interrupts until the completion of the Interrupt Service Routine.

Note:	The user can always lower the priority level by writing a new value into the STATUS register.

Unless the GIE bit is re-enabled, NO further interrupts will occur. The Interrupt Service Routine must clear the interrupt flag bits in the IFSx register before re-enabling interrupts, in order to avoid recursive interrupts.

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. The `RETFIE` instruction will also set the GIE bit to re-enable interrupts.

TABLE 8-2: EXCEPTION VECTORS


Decreasing Priority
IVT
AIVT

RESET - GOTO Instruction	0x000000
RESET - GOTO Address	0x000002
Oscillator Fail Trap Vector	0x000004
Stack Error Trap Vector	
Address Error Trap Vector	
Arithmetic Warning Trap Vector	
Software Trap Vector	
Reserved Vector	
Reserved Vector	
Reserved Vector	
Interrupt 0 Vector	0x000014
Interrupt 1 Vector	
~	
~	
~	
Interrupt 52 Vector	
Interrupt 53 Vector	0x00007E
reserved	0x000080
reserved	0x000082
Oscillator Fail Trap Vector	0x000084
Stack Error Trap Vector	
Address Error Trap Vector	
Arithmetic Warning Trap Vector	
Software Trap Vector	
Reserved Vector	
Reserved Vector	
Reserved Vector	
Interrupt 0 Vector	0x000094
Interrupt 1 Vector	
~	
~	
~	
Interrupt 52 Vector	
Interrupt 53 Vector	0x0000FE

8.5 Alternate Vector Table

In Program Memory, the Interrupt Vector Table (IVT) is followed by the Alternate Interrupt Vector Table (AIVT), as shown in Table 8-2. Access to the Alternate Vector Table is provided by the ALTIVT bit in the INTCON2 register. If the ALTIVT 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.

8.6 Fast Context Saving

A context saving option is available using shadow registers. Shadow registers are provided for the DA, DC, N, OV, SZ and C bits in SR, the TBLPAG and PSVPAG registers, and the registers W0 through W14. 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.

8.7 External Interrupt Requests

The interrupt controller supports up to five external interrupt request signals, INT0 - INT4. These inputs are edge sensitive, i.e., they require a low-to-high or a high-to-low transition to generate an interrupt request. The INTCON2 register has five bits, INT0EP - INT4EP, that select the polarity of the edge detection circuitry.

8.8 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 mode is active when the interrupt is generated.

If the GIE in the INTCON1 register is set, and 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.

If the GIE control bit is cleared, the interrupt controller will not generate an interrupt request to the processor. The interrupt enable bit must be set for a peripheral to wake the device from SLEEP. In this case, the processor will resume normal execution without processing an ISR.

FIGURE 8-2: INTERRUPT CONTROLLER BLOCK DIAGRAM

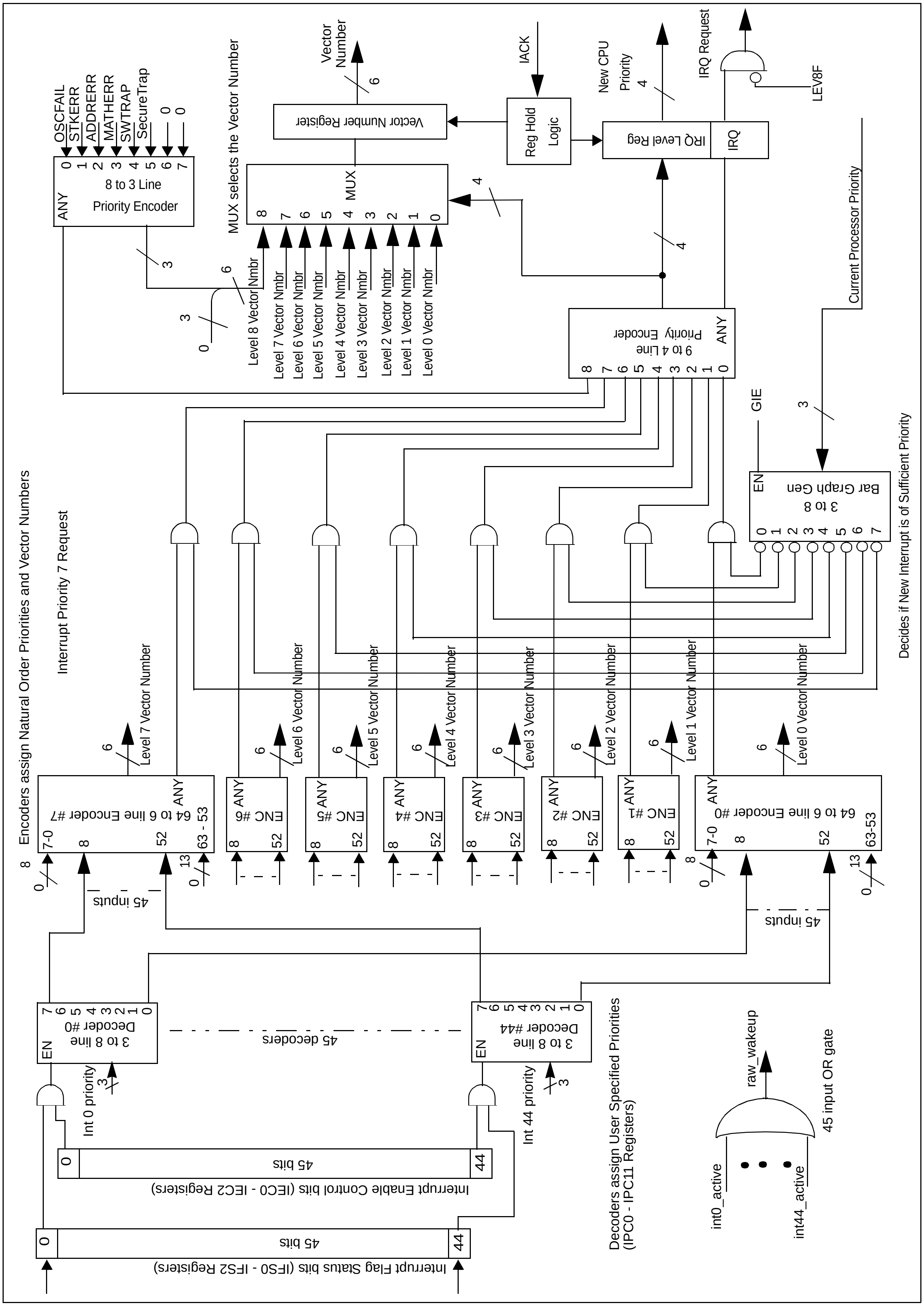


TABLE 8-3: INTERRUPT CONTROLLER REGISTER MAP

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	GIE	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	SWTRAP	OVRFLOW	ADDRERR	STKERR	—	0uuu u000 uuu0 000u
INTCON2	0082	ALTIVT	DISI	—	—	—	—	LEV8F	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	00uu uu0u uuu0 0000
IFS0	0084	CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF	T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0	0000 0000 0000 0000
IFS1	0086	IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF	0000 0000 0000 0000
IFS2	0088	—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	uuu0 0000 0000 0000
IEC0	008C	CNIE	BCLIE	I2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE	T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE	0000 0000 0000 0000
IEC1	008E	IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE	0000 0000 0000 0000
IEC2	0090	—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIE	PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	uuu0 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			u000 u000 u000 u000
IPC1	0096	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			u000 u000 u000 u000
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			u000 u000 u000 u000
IPC3	009A	—	CNIP<2:0>			—	BCLIP<2:0>			—	I2CIP<2:0>			—	NVMIP<2:0>			u000 u000 u000 u000
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			u000 u000 u000 u000
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			—	OC4IP<2:0>			u000 u000 u000 u000
IPC6	00A0	—	C1IP<2:0>			—	SPI2IP<2:0>			—	U2TXIP<2:0>			—	U2RXIP<2:0>			u000 u000 u000 u000
IPC7	00A2	—	IC6IP<2:0>			—	IC5IP<2:0>			—	IC4IP<2:0>			—	IC3IP<2:0>			u000 u000 u000 u000
IPC8	00A4	—	OC8IP<2:0>			—	OC7IP<2:0>			—	OC6IP<2:0>			—	OC5IP<2:0>			u000 u000 u000 u000
IPC9	00A6	—	PWMIP<2:0>			—	C2IP<2:0>			—	INT41IP<2:0>			—	INT3IP<2:0>			u000 u000 u000 u000
IPC10	00A8	—	FLTAIP<2:0>			—	LVDIP<2:0>			—	DCIIP<2:0>			—	QEIIP<2:0>			u000 u000 u000 u000
IPC11	00AA	—	—	—	—	—	—	—	—	—	—	—	—	—	FLTBIP<2:0>			uuuu uuuu uuuu u000

REGISTER 8-1: INTCON1: INTERRUPT CONTROL REGISTER1

Upper Half:							
R/W-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
GIE	—	—	—	—	OVATE	OVBTE	COVTE
bit 15							bit 8

Lower Half:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	—	—	SWTRAP	OVRFLOW	ADDRERR	STKERR	—
bit 7							bit 0

bit 15 **GIE:** Global Interrupt Enable bit
1 = Interrupts are enabled
0 = Interrupts are disabled

bit 14-11 **Unimplemented:** Read as ‘0’

bit 10 **OVATE:** Accumulator A Overflow Trap Enable bit
1 = Trap overflow of Accumulator A
0 = Trap disabled

bit 9 **OVBTE:** Accumulator B Overflow Trap Enable bit
1 = Trap overflow of Accumulator B
0 = Trap disabled

bit 8 **COVTE:** Catastrophic Overflow Trap Enable bit
1 = Trap on catastrophic overflow of Accumulator A or B enabled
0 = Trap disabled

bit 7-5 **Unimplemented:** Read as ‘0’

bit 4 **SWTRAP:** Software Trap Status bit
1 = Software trap has occurred
0 = Software trap has not occurred

bit 3 **OVRFLOW:** Arithmetic Error Status bit
1 = Overflow trap has occurred
0 = Overflow trap has not occurred

bit 2 **ADDRERR:** Address Error Trap Status bit
1 = Address error trap has occurred
0 = Address error trap has not occurred

bit 1 **STKERR:** Stack Error Trap Status bit
1 = Stack error trap has occurred
0 = Stack error trap has not occurred

bit 0 **Unimplemented:** Read as ‘0’

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

REGISTER 8-2: INTCON2: INTERRUPT CONTROL REGISTER2

Upper Half:							
R/W-0	R-0	U-0	U-0	U-0	U-0	HS/HC, SC-0	U-0
ALTIVT	DISI	—	—	—	—	LEV8F	—
bit 15							bit 8

Lower Half:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
bit 7							bit 0

bit 15 **ALTIVT:** Enable Alternate Interrupt Vector Table bit
1 = Use alternate vector table
0 = Use standard (default) vector table

bit 14 **DISI:** DISI Instruction Status bit
1 = DISI instruction is active
0 = DISI is not active

bit 13-10 **Unimplemented:** Read as ‘0’

bit 9 **LEV8F:** Level 8 Exception in Progress bit
1 = New exceptions (level 8 and below) are inhibited
0 = Exceptions are enabled

bit 8-5 **Unimplemented:** Read as ‘0’

bit 4 **INT4EP:** External Interrupt #4 Edge Detect Polarity Select bit
1 = Enable negative edge detect
0 = Enable positive edge detect

bit 3 **INT3EP:** External Interrupt #3 Edge Detect Polarity Select bit
1 = Enable negative edge detect
0 = Enable positive edge detect

bit 2 **INT2EP:** External Interrupt #2 Edge Detect Polarity Select bit
1 = Enable negative edge detect
0 = Enable positive edge detect

bit 1 **INT1EP:** External Interrupt #1 Edge Detect Polarity Select bit
1 = Enable negative edge detect
0 = Enable positive edge detect

bit 0 **INT0EP:** External Interrupt #0 Edge Detect Polarity Select bit
1 = Enable negative edge detect
0 = Enable positive edge detect

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

REGISTER 8-3: IFS0: INTERRUPT FLAG STATUS REGISTER0

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIF	BCLIF	I2CIF	NVMIF	ADIF	U1TXIF	U1RXIF	SPI1IF
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IF	T2IF	OC2IF	IC2IF	T1IF	OC1IF	IC1IF	INT0IF
bit 7							bit 0

bit 15 **CNIF:** Input Change Notification Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 14 **BCLIF:** I^{2}C Bus Collision Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 13 **I2CIF:** I^{2}C Transfer Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12 **NVMIF:** Non-Volatile Memory Write Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = No interrupt request has occurred

bit 11 **ADIF:** A/D Conversion Complete Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **U1TXIF:** UART1 Transmitter Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **U1RXIF:** UART1 Receiver Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **SPI1IF:** SPI1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **T3IF:** Timer3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **T2IF:** Timer2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **OC2IF:** Output Compare Channel 2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **IC2IF:** Input Capture Channel 2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **T1IF:** Timer1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **OC1IF:** Output Compare Channel 1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 1 **IC1IF:** Input Capture Channel 1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

REGISTER 8-3: IFS0: INTERRUPT FLAG STATUS REGISTER0 (Continued)

bit 0 **INT0IF:** External Interrupt 0 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

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

REGISTER 8-4: IFS1: INTERRUPT FLAG STATUS REGISTER1

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IC6IF	IC5IF	IC4IF	IC3IF	C1IF	SPI2IF	U2TXIF	U2RXIF
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IF	T5IF	T4IF	OC4IF	OC3IF	IC8IF	IC7IF	INT1IF
bit 7							bit 0

bit 15 **IC6IF:** Input Capture Channel 6 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 14 **IC5IF:** Input Capture Channel 5 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 13 **IC4IF:** Input Capture Channel 4 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 12 **IC3IF:** Input Capture Channel 3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 11 **C1IF:** CAN1 (Combined) Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **SPI2IF:** SPI2 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **U2TXIF:** UART2 Transmitter Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **U2RXIF:** UART2 Receiver Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **INT2IF:** External Interrupt 2 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **T5IF:** Timer5 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **T4IF:** Timer4 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **OC4IF:** Output Compare Channel 4 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **OC3IF:** Output Compare Channel 3 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **IC8IF:** Input Capture Channel 8 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

REGISTER 8-4: IFS1: INTERRUPT FLAG STATUS REGISTER1 (CONTINUED)

bit 1 **IC7IF:** Input Capture Channel 7 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **INT1IF:** External Interrupt 1 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

Legend:

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

REGISTER 8-5: IFS2: INTERRUPT FLAG STATUS REGISTER2

Upper Half:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLTBIF	FLTAIF	LVDIF	DCIIF	QEIIF
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as ‘0’

bit 12 **FLTBIF:** Fault B Input Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 11 **FLTAIF:** Fault A Input Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 10 **LVDIF:** Programmable Low Voltage Detect Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 9 **DCIIF:** Data Converter Interface Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 8 **QEIIF:** Quadrature Encoder Interface Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 7 **PWMIF:** Motor Control Pulse Width Modulation Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 6 **C2IF:** CAN2 (Combined) Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 5 **INT4IF:** External Interrupt 4 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 4 **INT3IF:** External Interrupt 3 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 3 **OC8IF:** Output Compare Channel 8 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 2 **OC7IF:** Output Compare Channel 7 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 1 **OC6IF:** Output Compare Channel 6 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **OC5IF:** Output Compare Channel 5 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

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

REGISTER 8-6: IEC0: INTERRUPT ENABLE CONTROL REGISTER0

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CNIE	BCLIE	I2CIE	NVMIE	ADIE	U1TXIE	U1RXIE	SPI1IE
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
T3IE	T2IE	OC2IE	IC2IE	T1IE	OC1IE	IC1IE	INT0IE
bit 7							bit 0

bit 15 **CNIE:** Input Change Notification Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 14 **BCLIE:** I^{2}C Bus Collision Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 13 **I2CIE:** I^{2}C Transfer Complete Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 12 **NVMIE:** Non-Volatile Memory Write Complete Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 11 **ADIE:** A/D Conversion Complete Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 10 **U1TXIE:** UART1 Transmitter Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 9 **U1RXIE:** UART1 Receiver Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 8 **SPI1IE:** SPI1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 7 **T3IE:** Timer3 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 6 **T2IE:** Timer2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **OC2IE:** Output Compare Channel 2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 4 **IC2IE:** Input Capture Channel 2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 3 **T1IE:** Timer1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 2 **OC1IE:** Output Compare Channel 1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

REGISTER 8-6: IEC0 - INTERRUPT ENABLE CONTROL REGISTER0 (Continued)

bit 1 **IC1IE:** Input Capture Channel 1 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 0 **INT0IE:** External Interrupt 0 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

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

REGISTER 8-7: IEC1: INTERRUPT ENABLE CONTROL REGISTER1

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IC6IE	IC5IE	IC4IE	IC3IE	C1IE	SPI2IE	U2TXIE	U2RXIE
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IE	T5IE	T4IE	OC4IE	OC3IE	IC8IE	IC7IE	INT1IE
bit 7							bit 0

bit 15 **IC6IE:** Input Capture Channel 6 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 14 **IC5IE:** Input Capture Channel 5 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 13 **IC4IE:** Input Capture Channel 4 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 12 **IC3IE:** Input Capture Channel 3 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 11 **C1IE:** CAN1 (Combined) Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 10 **SPI2IE:** SPI2 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 9 **U2TXIE:** UART2 Transmitter Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 8 **U2RXIE:** UART2 Receiver Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 7 **INT2IE:** External Interrupt 2 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 6 **T5IE:** Timer5 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **T4IE:** Timer4 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 4 **OC4IE:** Output Compare Channel 4 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 3 **OC3IE:** Output Compare Channel 3 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 2 **IC8IE:** Input Capture Channel 8 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

REGISTER 8-7: IEC1 - INTERRUPT ENABLE CONTROL REGISTER1 (Continued)

bit 1 **IC7IE:** Input Capture Channel 7 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 0 **INT1IE:** External Interrupt 1 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

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

REGISTER 8-8: IEC2: INTERRUPT ENABLE CONTROL REGISTER2

Upper Half:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLTBIE	FLTAIE	LVDIE	DCIIE	QEIIE
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as ‘0’

bit 12 **FLTBIE:** Fault B Input Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 11 **FLTAIE:** Fault A Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 10 **LVDIE:** Programmable Low Voltage Detect Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 9 **DCIIE:** Data Converter Interface Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 8 **QEIIE:** Quadrature Encoder Interface Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 7 **PWMIE:** Motor Control Pulse Width Modulation Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 6 **C2IE:** CAN2 (Combined) Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 5 **INT4IE:** External Interrupt 4 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 4 **INT3IE:** External Interrupt 3 Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 3 **OC8IE:** Output Compare Channel 8 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 2 **OC7IE:** Output Compare Channel 7 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 1 **OC6IE:** Output Compare Channel 6 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

bit 0 **OC5IE:** Output Compare Channel 5 Interrupt Enable bit
1 = Interrupt request enabled
0 = Interrupt request not enabled

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

REGISTER 8-9: IPC0: INTERRUPT PRIORITY CONTROL REGISTER0

Upper Half:

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	T1IP2	T1IP1	T1IP0	—	OC1IP2	OC1IP1	OC1IP0
bit 15							bit 8

Lower Half:

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	IC1IP2	IC1IP1	IC1IP0	—	INT0IP2	INT0IP1	INT0IP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **T1IP<2:0>:** Timer1 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **OC1IP<2:0>:** Output Compare Channel 1 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **IC1IP<2:0>:** Input Capture Channel 1 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **INT0IP<2:0>:** External Interrupt 0 Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-10: IPC1: INTERRUPT PRIORITY CONTROL REGISTER1

Upper Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	T3IP2	T3IP1	T3IP0	—	T2IP2	T2IP1	T2IP0
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	OC2IP2	OC2IP1	OC2IP0	—	IC2IP2	IC2IP1	IC2IP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **T3IP<2:0>:** Timer3 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **T2IP<2:0>:** Timer2 Interrupt Priority
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **OC2IP<2:0>:** Output Compare Channel 2 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **IC2IP<2:0>:** Input Capture Channel 2 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-11: IPC2: INTERRUPT PRIORITY CONTROL REGISTER2

Upper Half:

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	ADIP2	ADIP1	ADIP0	—	U1TXIP2	U1TXIP1	U1TXIP0
bit 15							bit 8

Lower Half:

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	U1RXIP2	U1RXIP1	U1RXIP0	—	SPI1IP2	SPI1IP1	SPI1IP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **ADIP<2:0>:** A/D Conversion Complete Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **U1TXIP<0>:** UART1 Transmitter Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **U1RXIP<2:0>:** UART1 Receiver Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **SPI1IP<2:0>:** SPI1 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-12: IPC3: INTERRUPT PRIORITY CONTROL REGISTER3

Upper Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	CNIP2	CNIP1	CNIP0	—	BCLIP2	BCLIP1	BCLIP0
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	I2CIP2	I2CIP1	I2CIP0	—	NVMIP2	NVMIP1	NVMIP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **CNIP<2:0>:** Input Change Notification Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
.
.
.
001 = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **BCLIP<2:0>:** I2C Bus Collision Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
.
.
.
001 = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **I2CIP<2:0>:** I2C Transfer Complete Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
.
.
.
001 = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **NVMIP<2:0>:** Non-Volatile Memory Write Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
.
.
.
001 = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-13: IPC4: INTERRUPT PRIORITY CONTROL REGISTER4

Upper Half:

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	OC3IP2	OC3IP1	OC3IP0	—	IC8IP2	IC8IP1	IC8IP0
bit 15							bit 8

Lower Half:

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	IC7IP2	IC7IP1	IC7IP0	—	INT1IP2	INT1IP1	INT1IP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **OC3IP<2:0>:** Output Compare Channel 3 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **IC8IP<2:0>:** Input Capture Channel 8 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **IC7IP<2:0>:** Input Capture Channel 7 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **INT1IP<2:0>:** External Interrupt 1 Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-14: IPC5: INTERRUPT PRIORITY CONTROL REGISTER5

Upper Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	INT2IP2	INT2IP1	INT2IP0	—	T5IP2	T5IP1	T5IP0
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	T4IP2	T4IP1	T4IP0	—	OC4IP2	OC4IP1	OC4IP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **INT2IP<2:0>:** External Interrupt 2 Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **T5IP<2:0>:** Timer5 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **T4IP<2:0>:** Timer4 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **OC4IP<2:0>:** Output Compare Channel 4 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-15: IPC6: INTERRUPT PRIORITY CONTROL REGISTER6

Upper Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	C1IP2	C1IP1	C1IP0	—	SPI2IP2	SPI2IP1	SPI2IP0
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	U2TXIP2	U2TXIP1	U2TXIP0	—	U2RXIP2	U2RXIP1	U2RXIP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **C1IP<2:0>:** CAN1 (combined) Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
•
•
•
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **SPI2IP<2:0>:** SPI2 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
•
•
•
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **U2TXIP<2:0>:** UART2 Transmitter Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
•
•
•
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **U2RXIP<2:0>:** UART2 Receiver Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
•
•
•
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-16: IPC7: INTERRUPT PRIORITY CONTROL REGISTER7

Upper Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	IC6IP2	IC6IP1	IC6IP0	—	IC5IP2	IC5IP1	IC5IP0
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	IC4IP2	IC4IP1	IC4IP0	—	IC3IP2	IC3IP1	IC3IP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **IC6IP<2:0>:** Input Capture Channel 6 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
▪
▪
▪
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **IC5IP<2:0>:** Input Capture Channel 5 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
▪
▪
▪
`001` = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **IC4IP<2:0>:** Input Capture Channel 4 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
▪
▪
▪
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **IC3IP<2:0>:** Input Capture Channel 3 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
▪
▪
▪
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-17: IPC8: INTERRUPT PRIORITY CONTROL REGISTER8

Upper Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	OC8IP2	OC8IP1	OC8IP0	—	OC7IP2	OC7IP1	OC7IP0
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	OC6IP2	OC6IP1	OC6IP0	—	OC5IP2	OC5IP1	OC5IP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **OC8IP<2:0>:** Output Compare Channel 8 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **OC7IP<2:0>:** Output Compare Channel 7 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **OC6IP<2:0>:** Output Compare Channel 6 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **OC5IP<2:0>:** Output Compare Channel 5 Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-18: IPC9: INTERRUPT PRIORITY CONTROL REGISTER9

Upper Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	PWMIP2	PWMIP1	PWMIP0	—	C2IP2	C2IP1	C2IP0
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	INT4IP2	INT4IP1	INT4IP0	—	INT3IP2	INT3IP1	INT3IP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **PWMIP<2:0>:** Motor Control Pulse Width Modulation Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
.
.
.
001 = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **C2IP<2:0>:** CAN2 (combined) Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
.
.
.
001 = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **INT4IP<2:0>:** External Interrupt 4 Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
.
.
.
001 = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **INT3IP<2:0>:** External Interrupt 3 Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
.
.
.
001 = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-19: IPC10: INTERRUPT PRIORITY CONTROL REGISTER10

Upper Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	FLTAIP2	FLTAIP1	FLTAIP0	—	LVDIP2	LVDIP1	LVDIP0
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
—	DCIIP2	DCIIP1	DCIIP0	—	QEIIP2	QEIIP1	QEIIP0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **FLTAIP<2:0>:** Fault A Input Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **LVDIP<2:0>:** Programmable Low Voltage Detect Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 7 **Unimplemented:** Read as ‘0’

bit 6-4 **DCIIP<2:0>:** Data Converter Interface Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

bit 3 **Unimplemented:** Read as ‘0’

bit 2-0 **QEIIP<2:0>:** Quadrature Encoder Interface Interrupt Priority bits
`111` = Interrupt is priority 7 (highest priority interrupt)
.
.
.
`001` = Interrupt is priority 1
`000` = Interrupt is priority 0 (lowest interrupt priority)

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

REGISTER 8-20: IPC11: INTERRUPT PRIORITY CONTROL REGISTER11

Upper Half:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Half:							
U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	FLTBIP2	FLTBIP1	FLTBIP0
bit 7							bit 0

bit 15-3 **Unimplemented:** Read as ‘0’

bit 2-0 **FLTBIP<2:0>:** Fault B Input Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
.
.
.
001 = Interrupt is priority 1
000 = Interrupt is priority 0 (lowest interrupt priority)

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

NOTES:

9.0 I/O PORTS

All of the device pins (except VDD, VSS, $\overline{\text{MCLR}}$, and OSC1/CLKIN) are shared between the peripherals and the parallel I/O ports.

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

9.1 Parallel I/O (PIO) Ports

When a peripheral is enabled, the use of any associated 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 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 is 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 PORTA is shown in Table 9-1.

The TRISA (Data Direction Control) register controls the direction of the RA<7:0> pins, as well as the INTx pins and the VREF pins. TRISA is a read/write register. The LATA register supplies data to the outputs, and is readable/writable. Reading the PORTA register yields the state of the input pins, while writing the PORTA register yields the contents of the LATA register.

FIGURE 9-1: BLOCK DIAGRAM OF A DEDICATED PORT STRUCTURE

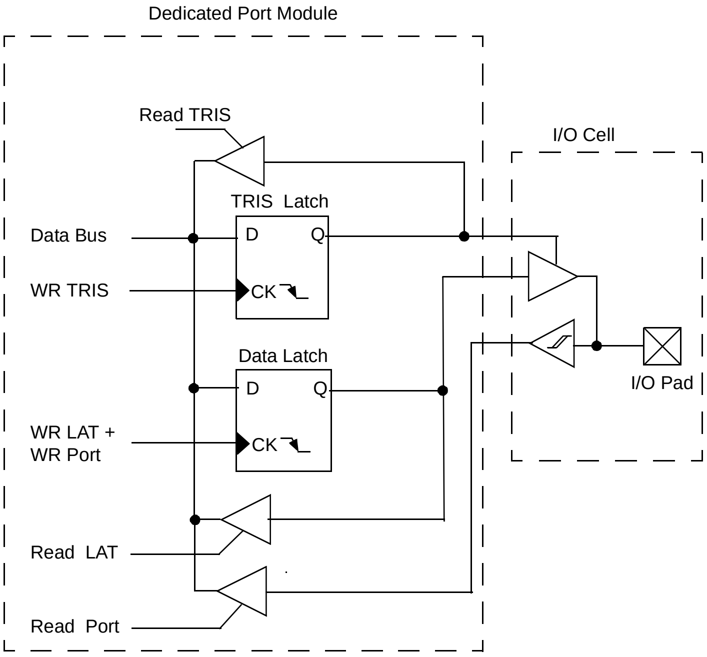


TABLE 9-1: PORTA REGISTER MAP

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
TRISA	02C0	TRISA15	TRISA14	TRISA13	TRISA12	TRISA11	TRISA10	TRISA9	—	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 111u 1111 1111
PORTA	02C2	RA15	RA14	RA13	RA12	RA11	RA10	RA9	—	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	0000 000u 0000 0000
LATA	02C4	LATA15	LATA14	LATA13	LATA12	LATA11	LATA10	LATA9	—	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	0000 000u 0000 0000

A parallel I/O (PIO) port that shares a pin with another peripheral is always subservient to the other 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 9-2 shows how ports are shared with other peripherals, and the associated I/O cell (pad) to which they are connected. Table 9-2 through Table 9-7 show the formats of the registers for the shared ports, PORTB through PORTG.

> **Note:** The actual bits in use vary between devices.

FIGURE 9-2: BLOCK DIAGRAM OF A SHARED PORT STRUCTURE

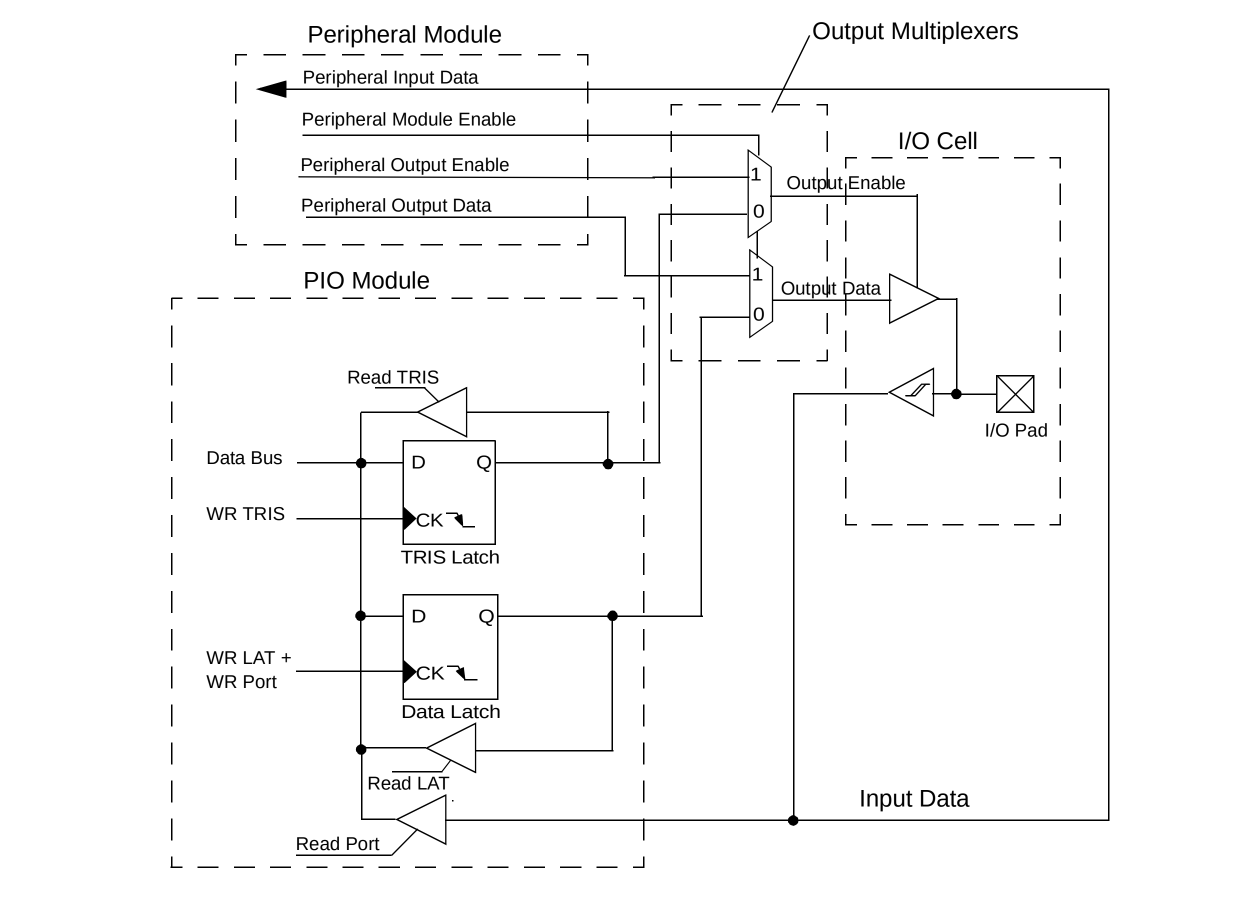


TABLE 9-2: PORTB REGISTER MAP

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	TRISB15	TRISB14	TRISB13	TRISB12	TRISB11	TRISB10	TRISB9	TRISB8	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111 1111 1111
PORTB	02C8	RB15	RB14	RB13	RB12	RB11	RB10	RB9	RB8	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	0000 0000 0000 0000
LATB	02CB	LATB15	LATB14	LATB13	LATB12	LATB11	LATB10	LATB9	LATB8	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	0000 0000 0000 0000

TABLE 9-3: PORTC 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
TRISC	02CC	TRISC15	TRISC14	TRISC13	—	—	—	—	—	—	—	—	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	111u uuuu uuu1 1111
PORTC	02CE	RC15	RC14	RC13	—	—	—	—	—	—	—	—	RC4	RC3	RC2	RC1	RC0	000u uuuu uuu0 0000
LATC	02D0	LATC15	LATC14	LATC13	—	—	—	—	—	—	—	—	LATC4	LATC3	LATC2	LATC1	LATC0	000u uuuu uuu0 0000

TABLE 9-4: PORTD 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
TRISD	02D2	TRISD15	TRISD14	TRISD13	TRISD12	TRISD11	TRISD10	TRISD9	TRISD8	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	1111 1111 1111 1111
PORTD	02D4	RD15	RD14	RD13	RD12	RD11	RD10	RD9	RD8	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	0000 0000 0000 0000
LATD	02D6	LATD15	LATD14	LATD13	LATD12	LATD11	LATD10	LATD9	LATD8	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	0000 0000 0000 0000

TABLE 9-5: PORTE 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
TRISE	02D8	—	—	—	—	—	—	TRISE9	TRISE8	TRISE7	TRISE6	TRISE5	TRISE4	TRISE3	TRISE2	TRISE1	TRISE0	uuuu uu11 1111 1111
PORTE	02DB	—	—	—	—	—	—	RE9	RE8	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0	uuuu uu00 0000 0000
LATE	02DC	—	—	—	—	—	—	LATE9	LATE8	LATE7	LATE6	LATE5	LATE4	LATE3	LATE2	LATE1	LATE0	uuuu uu00 0000 0000

TABLE 9-6: PORTF 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
TRISF	02EE	—	—	TRISF13	TRISF12	—	—	—	TRISF8	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	uu11 uuu1 1111 1111
PORTF	02E0	—	—	RF13	RF12	—	—	—	RF8	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	uu00 uuu0 0000 0000
LATF	02E2	—	—	LATF13	LATF12	—	—	—	LATF8	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0	uu00 uuu0 0000 0000

TABLE 9-7: PORTG 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
TRISG	02E4	TRISG15	TRISG14	TRISG13	TRISG12	—	—	TRISG9	TRISG8	TRISG7	TRISG6	—	—	TRISG3	TRISG2	TRISG1	TRISG0	1111 uu11 11uu 1111
PORTG	02E6	RG15	RG14	RG13	RG12	—	—	RG9	RG8	RG7	RG6	—	—	RG3	RG2	RG1	RG0	0000 uu00 00uu 0000
LATG	02E8	LATG15	LATG14	LATG13	LATG12	—	—	LATG9	LATG8	LATG7	LATG6	—	—	LATG3	LATG2	LATG1	LATG0	0000 uu00 00uu 0000

9.2 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 up to 24 external signals (CN0 through CN23) that may be selected (enabled) for generating an interrupt request on a change of state.

TABLE 9-8: INPUT CHANGE NOTIFICATION REGISTER MAP (BITS 15-8)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	RESET State
CNEN1	00C0	CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE	CN9IE	CN8IE	0000 0000 0000 0000
CNEN2	00C2	—	—	—	—	—	—	—	—	0000 0000 0000 0000
CNPU1	00C4	CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE	CN9PUE	CN8PUE	0000 0000 0000 0000
CNPU2	00C6	—	—	—	—	—	—	—	—	0000 0000 0000 0000

TABLE 9-9: 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	CN23IE	CN22IE	CN21IE	CN20IE	CN19IE	CN18IE	CN17IE	CN16IE	0000 0000 0000 0000
CNPU1	00C4	CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE	0000 0000 0000 0000
CNPU2	00C6	CN23PUE	CN22PUE	CN21PUE	CN20PUE	CN19PUE	CN18PUE	CN17PUE	CN16PUE	0000 0000 0000 0000

REGISTER 9-1: TRISA: PORTA DATA DIRECTION REGISTER

Upper Half:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
TRISA15	TRISA14	TRISA13	TRISA12	TRISA11	TRISA10	TRISA9	—
bit 15							bit 8

Lower Half:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
bit 7							bit 0

bit 15-9 **TRISA<15:9>:** PORTA Data Direction Control bits
`1` = I/O configured as an input
`0` = I/O configured as an output

bit 8 **Unimplemented:** Read as ‘0’

bit 7-0 **TRISA<7:0>:** PORTA Data Direction Control bits
`1` = I/O configured as an input
`0` = I/O configured as an output

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

REGISTER 9-2: PORTA: READ PIN/WRITE PORTA LATCH REGISTER

Upper Half:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	U-0
RA15	RA14	RA13	RA12	RA11	RA10	RA9	—
bit 15							bit 8

Lower Half:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
bit 7							bit 0

bit 15-9 **RA<15:9>:** PORTA PIO Read Pin/Write Latch Data bits

bit 8 **Unimplemented:** Read as ‘0’

bit 7-0 **RA<7:0>:** PORTA PIO Read Pin/Write Latch Data bits

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

REGISTER 9-3: LATA: READ/WRITE PORTA LATCH REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
LATA15	LATA14	LATA13	LATA12	LATA11	LATA10	LATA9	—
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
bit 7							bit 0

bit 15-9 **LATA<15:9>:** PORTA Read/Write Latch Data bits

bit 8 **Unimplemented:** Read as ‘0’

bit 7-0 **LATA<7:0>:** PORTA Read/Write Latch Data bits

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

REGISTER 9-4: TRISB: PORTB PIO DATA DIRECTION REGISTER

Upper Half:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISB15	TRISB14	TRISB13	TRISB12	TRISB11	TRISB10	TRISB9	TRISB8
bit 15							bit 8

Lower Half:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
bit 7							bit 0

bit 15-0 **TRISB<15:0>:** PORTB PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

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

REGISTER 9-5: PORTB: READ PORTB PIO PIN/WRITE PORTB PIO LATCH REGISTER

Upper Half:

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
RB15	RB14	RB13	RB12	RB11	RB10	RB9	RB8
bit 15							bit 8

Lower Half:

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
bit 7							bit 0

bit 15-0 **RB<15:0>:** PORTB PIO Read Pin/Write Latch Data bits

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

REGISTER 9-6: LATB: READ/WRITE PORTB PIO LATCH REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
LATB15	LATB14	LATB13	LATB12	LATB11	LATB10	LATB9	LATB8
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
bit 7							bit 0

bit 15-0 **LATB<15:0>:** PORTB PIO Read/Write Latch Data bits

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

REGISTER 9-7: TRISC: PORTC PIO DATA DIRECTION REGISTER

Upper Half:							
R/W-1	R/W-1	R/W-1	U-0	U-0	U-0	U-0	U-0
TRISC15	TRISC14	TRISC13	—	—	—	—	—
bit 15							bit 8

Lower Half:							
U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
bit 7							bit 0

bit 15-13 **TRISC<15:13>:** PORTC PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

bit 12-5 **Unimplemented:** Read as ‘0’

bit 4-0 **TRISC<4:0>:** PORTC PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

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

REGISTER 9-8: PORTC: READ PORTC PIO PIN/WRITE PORTC PIO LATCH REGISTER

Upper Half:							
R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0	U-0
RC15	RC14	RC13	—	—	—	—	—
bit 15							bit 8

Lower Half:							
U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	RC4	RC3	RC2	RC1	RC0
bit 7							bit 0

bit 15-13 **RC<15:13>:** PORTC PIO Read Pin/Write Latch Data bits

bit 12-5 **Unimplemented:** Read as ‘0’

bit 4-0 **RC<4:0>:** PORTC PIO Read Pin/Write Latch Data bits

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

REGISTER 9-9: LATC: READ/WRITE PORTC PIO LATCH REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
LATC15	LATC14	LATC13	—	—	—	—	—
bit 15							bit 8

Lower Half:

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	LATC4	LATC3	LATC2	LATC1	LATC0
bit 7							bit 0

bit 15-13 **LATC<15:13>:** PORTC PIO Read/Write Latch Data bits

bit 12-5 **Unimplemented:** Read as ‘0’

bit 4-0 **LATC<4:0> :** PORTC PIO Read/Write Latch Data bits

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

REGISTER 9-10: TRISD: PORTD PIO DATA DIRECTION REGISTER

Upper Half:

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISD15	TRISD14	TRISD13	TRISD12	TRISD11	TRISD10	TRISD9	TRISD8
bit 15							bit 8

Lower Half:

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
bit 7							bit 0

bit 15-0 **TRISD<15:0>:** PORTD PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

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

REGISTER 9-11: PORTD: READ PORTD PIO PIN/WRITE PORTD PIO LATCH REGISTER

Upper Half:

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
RD15	RD14	RD13	RD12	RD11	RD10	RD9	RD8
bit 15							bit 8

Lower Half:

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
bit 7							bit 0

bit 15-0 **RD<15:0>:** PORTD PIO Read Pin/Write Latch Data bits

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

REGISTER 9-12: LATD: READ/WRITE PORTD PIO LATCH REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
LATD15	LATD14	LATD13	LATD12	LATD11	LATD10	LATD9	LATD8
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0
bit 7							bit 0

bit 15-0 **LATD<15:0>:** PORTD PIO Read/Write Latch Data bits

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

REGISTER 9-13: TRISE: PORTE PIO DATA DIRECTION REGISTER

Upper Half:							
U-0	U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-1
—	—	—	—	—	—	TRISE9	TRISE8
bit 15							bit 8

Lower Half:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISE7	TRISE6	TRISE5	TRISE4	TRISE3	TRISE2	TRISE1	TRISE0
bit 7							bit 0

bit 15-10 **Unimplemented:** Read as ‘0’

bit 9-0 **TRISE<9:0>:** PORTE PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

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

REGISTER 9-14: PORTE: READ PORTE PIO PIN/WRITE PORTE PIO LATCH REGISTER

Upper Half:							
U-0	U-0	U-0	U-0	U-0	U-0	R/W-x	R/W-x
—	—	—	—	—	—	RE9	RE8
bit 15							bit 8

Lower Half:							
R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0
bit 7							bit 0

bit 15-10 **Unimplemented:** Read as ‘0’

bit 9-0 **RE<9:0>:** PORTE PIO Read Pin/Write Latch Data bits

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

REGISTER 9-15: LATE: READ/WRITE PORTE PIO LATCH REGISTER

Upper Half:							
U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	LATE9	LATE8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
LATE7	LATE6	LATE5	LATE4	LATE3	LATE2	LATE1	LATE0
bit 7							bit 0

bit 15-10 **Unimplemented:** Read as ‘0’

bit 9-0 **LATE<9:0>:** PORTE PIO Read/Write Latch Data bits

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

REGISTER 9-16: TRISF: PORTF PIO DATA DIRECTION REGISTER

Upper Half:							
U-0	U-0	R/W-1	R/W-1	U-0	U-0	U-0	R/W-1
—	—	TRISF13	TRISF12	—	—	—	TRISF8
bit 15							bit 8

Lower Half:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0
bit 7							bit 0

bit 15-14 **Unimplemented:** Read as ‘0’

bit 13-12 **TRISF<13:12>:** PORTF PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

bit 11-9 **Unimplemented:** Read as ‘0’

bit 8-0 **TRISF<8:0>:** PORTF PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

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

REGISTER 9-17: PORTF: READ PORTF PIO PIN/WRITE PORTF PIO LATCH REGISTER

Upper Half:

U-0	U-0	R/W-x	R/W-x	U-0	U-0	U-0	R/W-1
—	—	RF13	RF12	—	—	—	RF8
bit 15							bit 8

Lower Half:

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0
bit 7							bit 0

bit 15-14 **Unimplemented:** Read as ‘0’

bit 13-12 **RF<13:12>:** PORTF PIO Read Pin/Write Latch Data bits

bit 11-9 **Unimplemented:** Read as ‘0’

bit 8-0 **RF<8:0>:** PORTF PIO Read Pin/Write Latch Data bits

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

REGISTER 9-18: LATF: READ/WRITE PORTF PIO LATCH REGISTER

Upper Half:

U-0	U-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0
—	—	LATF13	LATF12	—	—	—	LATF8
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0
bit 7							bit 0

bit 15-14 **Unimplemented:** Read as ‘0’

bit 13-12 **LATF<13:12>:** PORTF PIO Read/Write Latch Data bits

bit 11-9 **Unimplemented:** Read as ‘0’

bit 8-0 **LATF<8:0>:** PORTF PIO Read/Write Latch Data bits

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

REGISTER 9-19: TRISG: PORTG PIO DATA DIRECTION REGISTER

Upper Half:							
R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	R/W-1	R/W-1
TRISG15	TRISG14	TRISG13	TRISG12	—	—	TRISG9	TRISG8
bit 15							bit 8

Lower Half:							
R/W-1	R/W-1	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
TRISG7	TRISG6	—	—	TRISG3	TRISG2	TRISG1	TRISG0
bit 7							bit 0

bit 15-12 **TRISG<15:12>:** PORTG PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

bit 11-10 **Unimplemented:** Read as ‘0’

bit 9-6 **TRISG<9:6>:** PORTG PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

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

bit 3-0 **TRISG<3:0>:** PORTG PIO Data Direction Control bits
1 = I/O configured as an input
0 = I/O configured as an output

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

REGISTER 9-20: PORTG: READ PORTG PIO PIN/WRITE PORTG PIO LATCH REGISTER

Upper Half:							
R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	R/W-x	R/W-x
RG15	RG14	RG13	RG12	—	—	RG9	RG8
bit 15							bit 8

Lower Half:							
R/W-x	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
RG7	RG6	—	—	RG3	RG2	RG1	RG0
bit 7							bit 0

bit 15-12 **RG<15:12>:** PORTG PIO Read Pin/Write Latch Data bits

bit 11-10 **Unimplemented:** Read as ‘0’

bit 9-6 **RG<9:6>:** PORTG PIO Read Pin/Write Latch Data bits

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

bit 3-0 **RG<3:0>:** PORTG PIO Read Pin/Write Latch Data bits

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

REGISTER 9-21: LATG: READ/WRITE PORTG PIO LATCH REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
LATG15	LATG14	LATG13	LATG12	—	—	LATG9	LATG8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
LATG7	LATG6	—	—	LATG3	LATG2	LATG1	LATG0
bit 7							bit 0

bit 15-12 **LATG<15:12>:** PORTG PIO Read/Write Latch Data bits

bit 11-10 **Unimplemented:** Read as ‘0’

bit 9-6 **LATG<9:6>:** PORTG PIO Read/Write Latch Data bits

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

bit 3-0 **LATG<3:0>:** PORTG PIO Read/Write Latch Data bits

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

REGISTER 9-22: CNEN1: INPUT CHANGE NOTIFICATION INTERRUPT ENABLE REGISTER1

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE	CN9IE	CN8IE
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE
bit 7							bit 0

bit 15-0 **CN<15:0>IE:** Input Change Notification Interrupt Enable bit
1 = Interrupt on input change is enabled
0 = Interrupt on input change not enabled

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

REGISTER 9-23: CNEN2: INPUT CHANGE NOTIFICATION INTERRUPT ENABLE REGISTER2

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN23IE	CN22IE	CN21IE	CN20IE	CN19IE	CN18IE	CN17IE	CN16IE
bit 7							bit 0

bit 15-8 **Unimplemented:** Read as ‘0’

bit 7-0 **CN<23:16>IE:** Input Change Notification Interrupt Enable bits
1 = Interrupt on input change is enabled
0 = Interrupt on input change not enabled

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

REGISTER 9-24: CNPU1: INPUT CHANGE NOTIFICATION PULL-UP ENABLE REGISTER1

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE	CN9PUE	CN8PUE
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE
bit 7							bit 0

bit 15-0 **CN<15:0>PUE :** Input Change Notification Pull-up Enable bit
1 = Pull-up on input change is enabled
0 = Pull-up on input change not enabled

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

REGISTER 9-25: CNPU2: INPUT CHANGE NOTIFICATION PULL-UP ENABLE REGISTER2

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CN23PUE	CN22PUE	CN21PUE	CN20PUE	CN19PUE	CN18PUE	CN17PUE	CN16PUE
bit 7							bit 0

bit 15-8 **Unimplemented:** Read as ‘0’

bit 7-0 **CN<23:16>PUE:** Input Change Notification Pull-up Enable bit
1 = Pull-up on input change is enabled
0 = Pull-up on input change not enabled

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

NOTES:

10.0 TIMER1 MODULE

This section describes the 16-bit General Purpose (GP) Timer1 module and associated operational modes. Figure 10-1 depicts the simplified block diagram of the 16-bit Timer1 Module.

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 10-1 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.

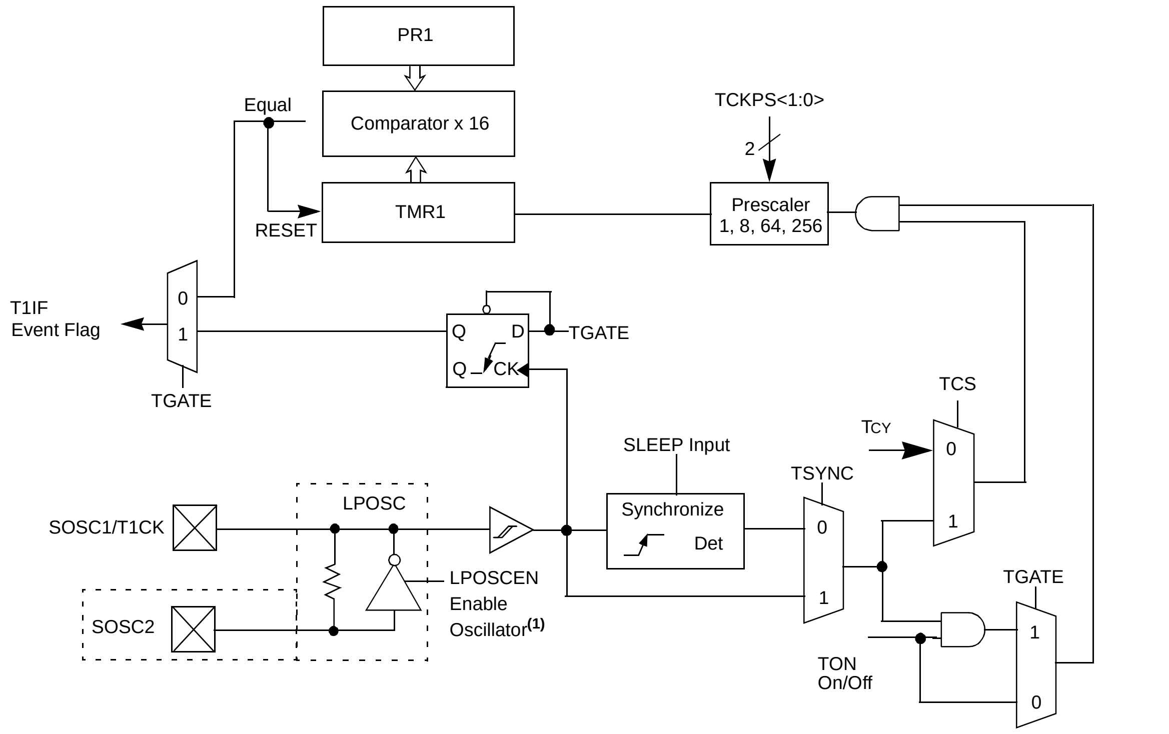
16-bit Synchronous Counter Mode: In the 16-bit Synchronous Counter mode, the timer increments on the rising edge of the applied external clock signal, which is synchronized with the internal phase clocks. The timer counts up to a match value preloaded in PR1, then resets to 0 and continues.

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 not stop incrementing, independent of the TSIDL bit. The TSIDL bit is ignored for this Timer mode.

FIGURE 10-1: 16-BIT TIMER1 MODULE BLOCK DIAGRAM



Note 1: When enable bit LPOSCEN is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

10.1 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.

10.2 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
- a write to the T1CON register
- 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.

10.3 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 = 0) and
- The TSYNC bit 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.

10.4 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.

10.5 Real-Time Clock

The Real-Time Clock (RTC) provides time-of-day and event time stamping capabilities. Key operational features of the RTC are:

- RTC Oscillator operation
- 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.

10.5.1 RTC OSCILLATOR OPERATION

When the 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 or IDLE mode, the timer will continue to increment, provided the external clock is active and the control bits have not been changed. The TSIDL bit is excluded from the control of the timer module for this mode.

10.5.2 REAL-TIME CLOCK INTERRUPTS

When an interrupt event occurs, the respective 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.

TABLE 10-1: 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	Timer 1 Register																0000 0000 0000 0000
PR1	0102	Period Register 1																1111 1111 1111 1111
T1CON	0104	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—	0u0u uuuu u000 u00u

REGISTER 10-1: T1CON: TIMER1 CONTROL REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer1 On bit
1 = Starts 16-bit Timer1
0 = Stops 16-bit Timer1

bit 14 **Unimplemented:** Read as '0’

bit 13 **TSIDL:** Timer1 Stop in IDLE Control bit
1 = Timer will halt in CPU IDLE mode
0 = Timer will continue to operate in CPU IDLE mode

bit 12-7 **Unimplemented:** Read as ’0’

bit 6 **TGATE:** Timer1 Gated Time Accumulation Enable bit
1 = Timer1 gated time accumulation enabled
0 = Timer1 gated time accumulation disabled
(TCS must be set to logic 0 when TGATE = 1)

bit 5-4 **TCKPS<1:0>** Timer1 Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

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

bit 2 **TSYNC:** Timer1 External Clock Input Synchronization Select bit

When TCS = 1:
1 = Synchronize external clock input
0 = Do not synchronize external clock input

When TCS = 0:
This bit is ignored. Timer1 uses the internal clock when TCS = 0.

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

bit 0 **Unimplemented:** Read as '0’

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

11.0 TIMER2/3 MODULE

This section describes the 32-bit General Purpose (GP) Timer modules (Timer2/3) and associated operational modes. Figure 11-1 depicts the simplified block diagram of the 32-bit GP Timer module.

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
- 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. Figure 11-1 presents a simple block diagram of the 32-bit timer.

For 32-bit timer/counter operation, Timer2 is the LS Word (LSW) and Timer3 is the MS Word (MSW) 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. Timer 2 clock and gate inputs are utilized for the 32-bit timer module, but an interrupt is generated with the Timer3 interrupt flag (T3IF).

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.

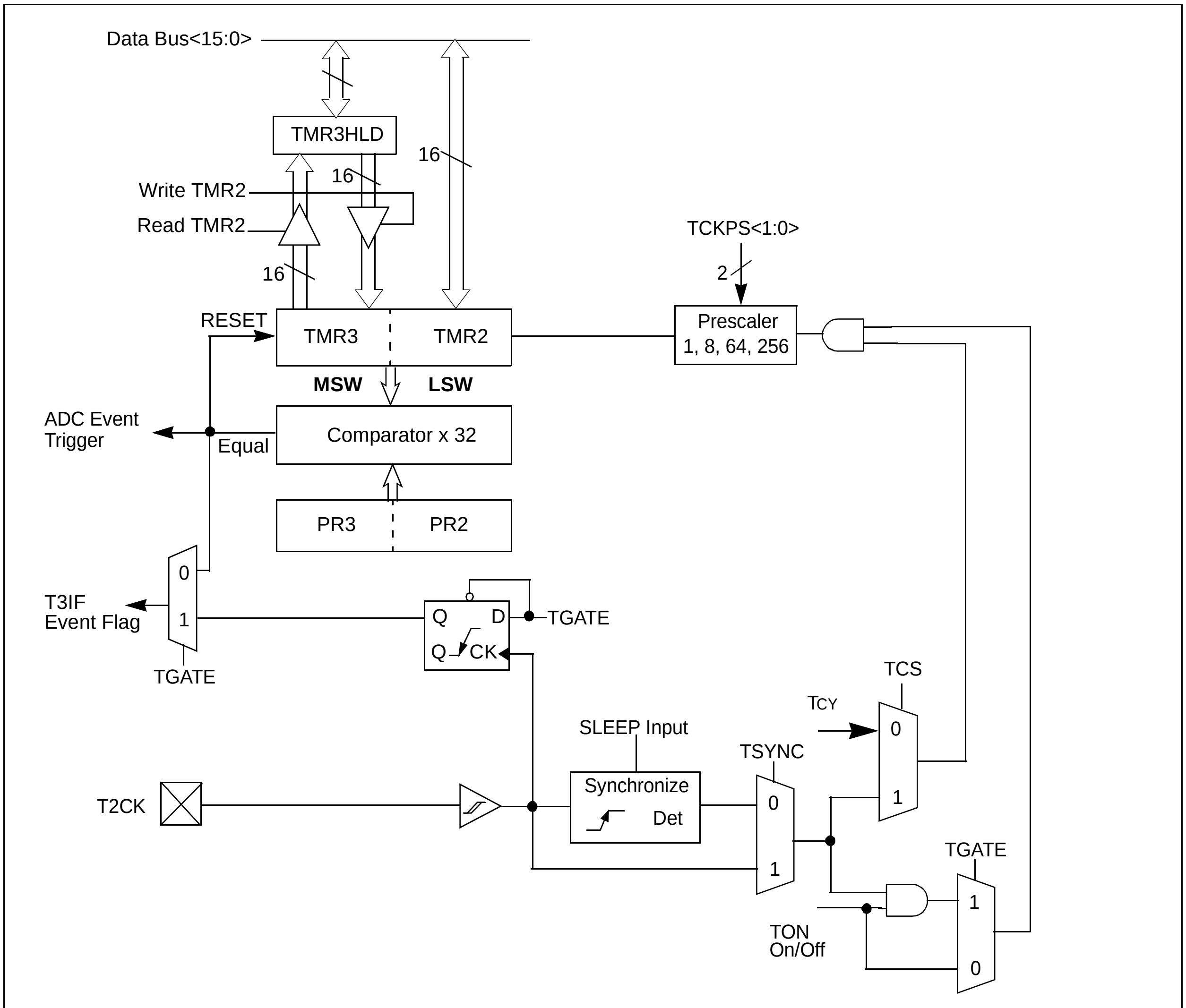
For synchronous 32-bit reads of the Timer2/Timer3 pair, reading the LSB (TMR2 register) will cause the MSB to be read and latched into a 16-bit holding register, termed TMR3HLD.

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.

FIGURE 11-1: 32-BIT TIMER MODULE BLOCK DIAGRAM[1]



Note 1: Timer configuration bit T32, T2CON(<3>), must be set to 1 for a 32-bit timer/counter operation.

11.1 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.

11.2 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.

11.3 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
- a write to the T2CON/T3CON register
- 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.

11.4 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 = 0) and
- The TSYNC bit 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 0x00000000.

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.

11.5 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 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. The timer interrupt flag, T3IF, is located in the IFS0 status register in the Interrupt Controller.

Enabling an interrupt is accomplished via the respective timer interrupt enable bit, T3IE. The timer interrupt enable bit is located in the IEC0 control register.

TABLE 11-1: 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																0000 0000 0000 0000
TMR3HLD	0108	Timer3 Holding Register (For 32 bit timer operations only)																0000 0000 0000 0000
TMR3	010A	Timer3 Register																0000 0000 0000 0000
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	TSYNC	TCS	—	0u0u uuuu u000 000u
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—	0u0u uuuu u000 000u

REGISTER 11-1: T2CON: TIMER2 CONTROL REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS1	TCKPS0	T32	TSYNC	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer2 On bit

When T32 = 1 (in 32-bit Timer mode):
1 = Starts 32-bit TMR3:TMR2
0 = Stops 32-bit TMR3:TMR2

When T32 = 0 (in 16-bit Timer mode):
1 = Starts 16-bit Timer2
0 = Stops 16-bit Timer2

bit 14 **Unimplemented:** Read as '0’

bit 13 **TSIDL:** Timer2 Stop in IDLE Control bit
1 = Timer will halt in CPU IDLE mode
0 = Timer will continue to operate in CPU IDLE mode
(Bit enables 16- and 32-bit timer operation for CPU SLEEP IDLE state)

bit 12-7 **Unimplemented:** Read as ’0’

bit 6 **TGATE:** Timer2 Gated Time Accumulation Enable bit
1 = Timer2 gated time accumulation enabled
0 = Timer2 gated time accumulation disabled
(TCS must be set to logic 0 when TGATE = 1)

bit 5-4 **TCKPS<1:0>** Timer2 Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3 **T32:** 32-bit Timer Mode Select bits
1 = Timer2 and Timer3 form a 32-bit timer
0 = Timer2 and Timer3 form 16-bit timers

Note: For 32-bit timer operation, T3CON control bits do not affect 32-bit timer operation.

bit 2 **TSYNC:** Timer2 External Clock Input Synchronization Select bit

When TCS = 1:
1 = Synchronize external clock input
0 = Do not synchronize external clock input

When TCS = 0:
This bit is ignored. Timer2 uses the internal clock when TCS = 0.

bit 1 **TCS:** Timer2 Clock Source Select bit
1 = External clock from pin T2CK (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **Unimplemented:** Read as '0’

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

REGISTER 11-2: T3CON: TIMER3 CONTROL REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Half:							
U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer3 On bit
1 = Starts 16-bit Timer3
0 = Stops 16-bit Timer3

Note: For 32-bit timer operation, T3CON control bits do not affect 32-bit timer operation.

bit 14 **Unimplemented:** Read as '0'

bit 13 **TSIDL:** Timer3 Stop in IDLE Control bit
1 = Timer will halt in CPU IDLE mode
0 = Timer will continue to operate in CPU IDLE mode

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **TGATE:** Timer3 Gated Time Accumulation Enable bit
1 = Timer3 gated time accumulation enabled
0 = Timer3 gated time accumulation disabled
(TCS must be set to logic 0 when TGATE = 1)

bit 5-4 **TCKPS<1:0>** Timer3 Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

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

bit 2 **TSYNC:** Timer3 External Clock Input Synchronization Select bit

When TCS = 1:
1 = Synchronize external clock input
0 = Do not synchronize external clock input

When TCS = 0:
This bit is ignored. Timer3 uses the internal clock when TCS = 0.

bit 1 **TCS:** Timer3 Clock Source Select bit
1 = External clock from pin T3CKI (on the rising edge)
0 = Internal clock (FOSC/4)

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

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

12.0 TIMER4/5 MODULE

This section describes the second 32-bit General Purpose (GP) Timer module (Timer4/5) and associated operational modes. Figure 12-1 depicts the simplified block diagram of the 32-bit GP Timer Module. The Timer 4/5 module is identical in operation to the Timer 2/3 module.

The Timer4/5 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 (Timer4 and Timer5) with all 16-bit Operating modes.
- Single 32-bit Timer operation
- Single 32-bit Synchronous Counter

Further, the following operational characteristics are supported:

- 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 T4CON and T5CON SFRs. Figure 12-1 presents a simple block diagram of the 32-bit timer.

For 32-bit timer/counter operation, Timer4 is the LS Word (LSW) and Timer5 is the MS Word (MSW) 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).

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 PR5/PR4, then resets to 0 and continues to count.

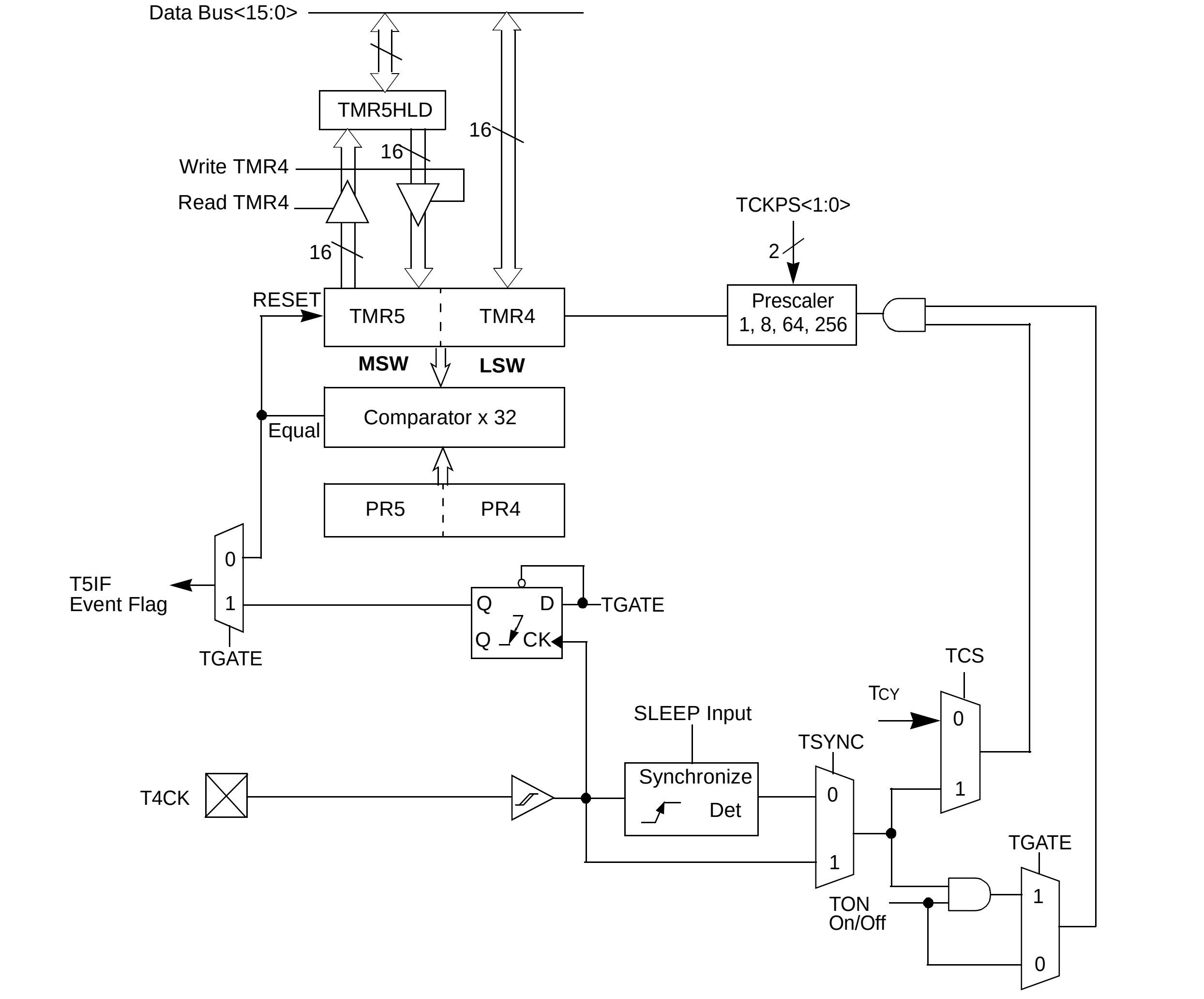
For synchronous 32-bit reads of the Timer4/Timer5 pair, reading the LSB (TMR4 register) will cause the MSB to be read and latched into a 16-bit holding register, termed TMR5HLD.

For synchronous 32-bit writes, the holding register (TMR5HLD) must first be written to. When followed by a write to the TMR4 register, the contents of TMR5HLD will be transferred and latched into the MSB of the 32-bit timer (TMR5).

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 PR5/PR4, 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 (T4CON<13>) bit = 0. If TSIDL = 1, the timer module logic will resume the incrementing sequence upon termination of the CPU IDLE mode.

FIGURE 12-1: 32-BIT TIMER MODULE BLOCK DIAGRAM[1]



Note 1: Timer configuration bit T45, T4CON(<3>), must be set to 1 for a 32-bit timer/counter operation.

12.1 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 (T4CK pin) is asserted high. Control bit TGATE (T4CON<6>) must be set to enable this mode. When in this mode, Timer4 is the originating clock source. The TGATE setting is ignored for Timer5. 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.

12.2 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> (T4CON<5:4> and T5CON<5:4>). For the 32-bit timer operation, the originating clock source is Timer4. The prescaler operation for Timer5 is not applicable in this mode. The prescaler counter is cleared when any of the following occurs:

- a write to the TMR4/TMR5 register
- a write to the T4CON/T5CON register
- device RESET such as POR and BOR

However, if the timer is disabled (TON = 0), then the Timer 4 prescaler cannot be reset, since the prescaler clock is halted.

TMR4/TMR5 is not cleared when T4CON/T5CON is written.

12.3 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 = 0) and
- The TSYNC bit 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 0x00000000.

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.

12.4 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 T5IF bit is asserted and an interrupt will be generated, if enabled. In this mode, the T5IF interrupt flag is used as the source of the interrupt. The T5IF bit must be cleared in software. The timer interrupt flag, T5IF, is located in the corresponding IFS1 Status register in the Interrupt Controller.

Enabling an interrupt is accomplished via the respective timer interrupt enable bit, T5IE. The timer interrupt enable bit, is located in the corresponding IEC1 Control register.

TABLE 12-1: 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	Timer 4 Register																0000 0000 0000 0000
TMR5HLD	0116	Timer 5 Holding Register (For 32 bit operations only)																0000 0000 0000 0000
TMR5	0118	Timer 5 Register																0000 0000 0000 0000
PR4	011A	Period Register 4																1111 1111 1111 1111
PR5	011C	Period Register 5																1111 1111 1111 1111
T4CON	011E	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T45	TSYNC	TCS	—	0u0u uuuu u000 000u
T5CON	0120	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—	0u0u uuuu u000 000u

REGISTER 12-1: T4CON: TIMER4 CONTROL REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Half:

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS1	TCKPS0	T45	TSYNC	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer4 On bit

When T45 = 1 (in 32-bit Timer mode):
1 = Starts 32-bit TMR5:TMR4
0 = Stops 32-bit TMR5:TMR4

When T45 = 0 (in 16-bit Timer mode):
1 = Starts 16-bit Timer4
0 = Stops 16-bit Timer4

bit 14 **Unimplemented:** Read as '0'

bit 13 **TSIDL:** Timer4 Stop in IDLE Control bit
1 = Timer will halt in CPU IDLE mode
0 = Timer will continue to operate in CPU IDLE mode
(Bit enables 16- and 32-bit timer operation for CPU SLEEP IDLE state)

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **TGATE:** Timer4 Gated Time Accumulation Enable bit
1 = Timer4 gated time accumulation enabled
0 = Timer4 gated time accumulation disabled
(TCS must be set to logic 0 when TGATE = 1)

bit 5-4 **TCKPS<1:0>** Timer4 Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3 **T45:** 32-bit Timer Mode Select bits
1 = Timer4 and Timer5 form a 32-bit timer
0 = Timer4 and Timer5 form 16-bit timers

Note: For 32-bit Timer operation, T5CON control bits do not affect 32-bit timer operation.

bit 2 **TSYNC:** Timer4 External Clock Input Synchronization Select bit

When TCS = 1:
1 = Synchronize external clock input
0 = Do not synchronize external clock input

When TCS = 0:
This bit is ignored. Timer4 uses the internal clock when TCS = 0.

bit 1 **TCS:** Timer4 Clock Source Select bit
1 = External clock from pin T4CK (on the rising edge)
0 = Internal clock (FOSC/4)

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

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

REGISTER 12-2: T5CON: TIMER5 CONTROL REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

Lower Half:

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—
bit 7							bit 0

bit 15 **TON:** Timer5 On bit
1 = Starts 16-bit Timer5
0 = Stops 16-bit Timer5

Note: For 32-bit Timer operation, T5CON control bits do not affect 32-bit timer operation.

bit 14 **Unimplemented:** Read as '0’

bit 13 **TSIDL:** Timer5 Stop in IDLE Control bit
1 = Timer will halt in CPU IDLE mode
0 = Timer will continue to operate in CPU IDLE mode

bit 12-7 **Unimplemented:** Read as ’0’

bit 6 **TGATE:** Timer5 Gated Time Accumulation Enable bit
1 = Timer5 gated time accumulation enabled
0 = Timer5 gated time accumulation disabled
(TCS must be set to logic 0 when TGATE = 1)

bit 5-4 **TCKPS<1:0>** Timer5 Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value

bit 3 **Unimplemented:** Read as '0’

bit 2 **TSYNC:** Timer5 External Clock Input Synchronization Select bit

When TCS = 1:
1 = Synchronize external clock input
0 = Do not synchronize external clock input

When TCS = 0:
This bit is ignored. Timer5 uses the internal clock when TCS = 0.

bit 1 **TCS:** Timer5 Clock Source Select bit
1 = External clock from pin T5CK (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **Unimplemented:** Read as '0’

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

13.0 INPUT CAPTURE MODULE

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 13-1 depicts a block diagram of the Input Capture Module. Input capture is useful for such modes as:

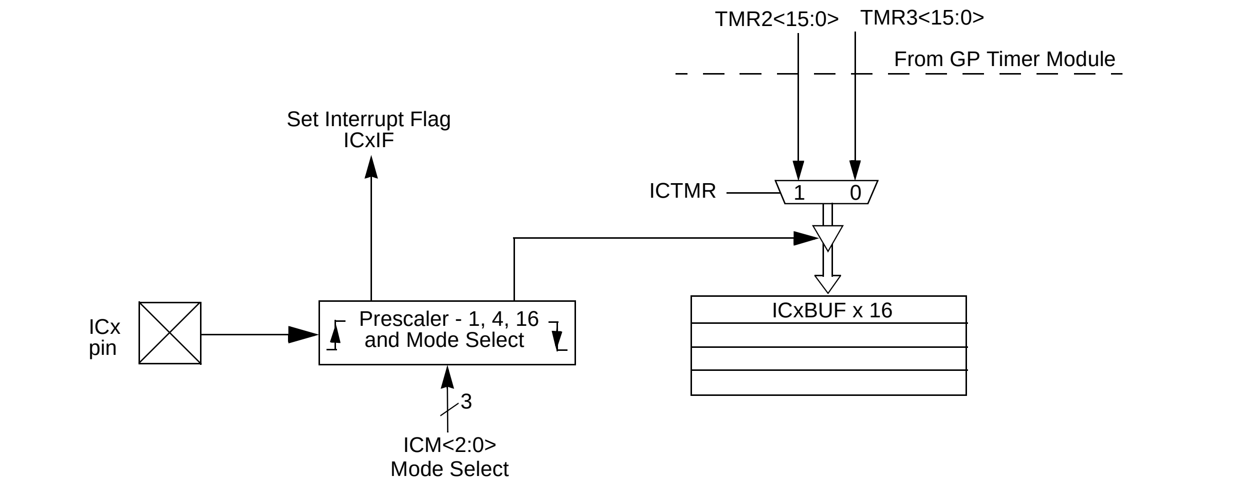
- Frequency/Period/Pulse Measurements
- Additional sources of external interrupts

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 dsPIC devices contain up to 8 capture channels, i.e., the maximum value of N is 8.

FIGURE 13-1: INPUT CAPTURE MODULE BLOCK DIAGRAM



Note 1: Where ‘x’ is shown, reference is made to the registers or bits associated with the respective input capture channels 1 through N.

13.1 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>).

13.1.1 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.

13.1.2 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:

- ICBFNE - Input Capture Buffer Not Empty
- ICOV - Input Capture Overflow

The ICBFNE 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.

13.1.3 TIMER2 AND TIMER3 SELECTION MODE

The input capture module consists of up to four input capture channels. Each 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.

13.1.4 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.

13.2 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.

13.2.1 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 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.

13.2.2 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 ICI<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.

13.3 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 ICI<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 Status 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 IEC Control register.

TABLE 13-1: 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																0000 0000 0000 0000
IC1CON	0142	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			uu0u uuuu 0000 0000
IC2BUF	0144	Input 2 Capture Register																0000 0000 0000 0000
IC2CON	0146	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			uu0u uuuu 0000 0000
IC3BUF	0148	Input 3 Capture Register																0000 0000 0000 0000
IC3CON	014A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			uu0u uuuu 0000 0000
IC4BUF	014C	Input 4 Capture Register																0000 0000 0000 0000
IC4CON	014E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			uu0u uuuu 0000 0000
IC5BUF	0150	Input 5 Capture Register																0000 0000 0000 0000
IC5CON	0152	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			uu0u uuuu 0000 0000
IC6BUF	0154	Input 6 Capture Register																0000 0000 0000 0000
IC6CON	0156	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			uu0u uuuu 0000 0000
IC7BUF	0158	Input 7 Capture Register																0000 0000 0000 0000
IC7CON	015A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			uu0u uuuu 0000 0000
IC8BUF	015C	Input 8 Capture Register																0000 0000 0000 0000
IC8CON	015E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			uu0u uuuu 0000 0000

REGISTER 13-1: ICxCON: INPUT CAPTURE x CONTROL REGISTER

Upper Half:							
U-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	ICSIDL	—	—	—	—	—
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0
ICTMR	ICI1	ICI0	ICOV	ICBNE	ICM2	ICM1	ICM0
bit 7							bit 0

bit 15-14 **Unimplemented:** Read as '0'

bit 13 **ICSIDL:** Input Capture Stop in IDLE Control bit
1 = Input Capture x will halt in CPU IDLE mode
0 = Input Capture x will continue to operate in CPU IDLE mode

bit 12-8 **Unimplemented:** Read as '0'

bit 7 **ICTMR:** Input Capture Timer Select bits
1 = Timer2 is the clock source for Capture x
0 = Timer3 is the clock source for Capture x

bit 6-5 **ICI<1:0>** Select Number of Captures per Interrupt bits
11 = Interrupt on every fourth Capture event
10 = Interrupt on every third Capture event
01 = Interrupt on every second Capture event
00 = Interrupt on every Capture event

bit 4 **ICOV:** Input Capture Overflow Status Flag (Read Only) bit
1 = Input Capture x overflow occurred
0 = No Input Capture x overflow occurred

bit 3 **ICBNE:** Input Capture Buffer Empty Status (Read Only) bit
1 = Input Capture x buffer is not empty, at least one more capture value can be read
0 = Input Capture x buffer is empty

bit 2-0 **ICM<2:0>** Input Capture Mode Select bits
111 = Input Capture x functions as interrupt pin only in CPU SLEEP and IDLE mode
(Rising edge detect only, all other control bits are not applicable)
110 = Unused (module disabled)
101 = Capture mode, every 16th rising edge
100 = Capture mode, every 4th rising edge
011 = Capture mode, every rising edge
010 = Capture mode, every falling edge
001 = Capture mode, every edge (rising and falling)
(ICI<1:0> ignored for this mode)
000 = Input Capture x off

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

Note: There is one ICxCON register for each input capture channel (x = 1, 2, ..., N).

14.0 OUTPUT COMPARE MODULE

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 14-1 depicts a block diagram of the Output Compare module.

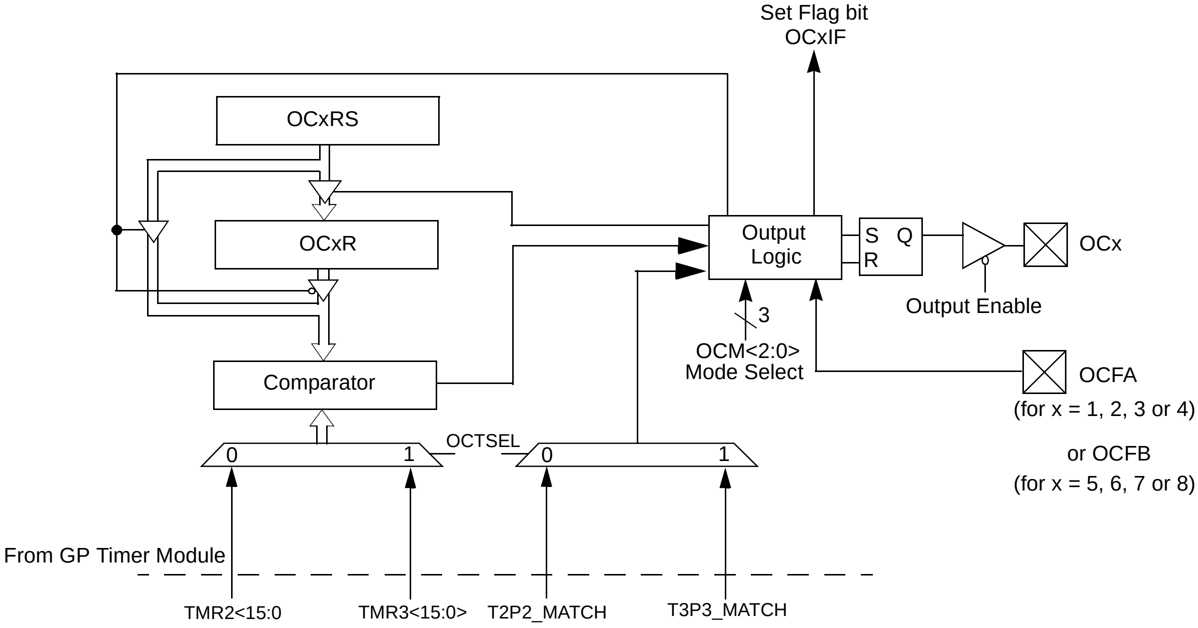
The key operational features of the Output Compare module include:

- 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 dsPIC devices contain up to 8 compare channels, i.e., the maximum value of N is 8.

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. When configured for the PWM mode of operation, the OCxR is the Main latch (read only) and OCxRS is the Secondary latch.

FIGURE 14-1: OUTPUT COMPARE MODE BLOCK DIAGRAM



Note 1: Where ‘x’ is shown, reference is made to the registers associated with the respective output compare channels 1 through N.

14.1 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.

14.2 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.

14.3 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

14.3.1 SINGLE 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 TCY.
- Calculate desired pulse width value base upon TCY.
- 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`.

Figure 14-2 depicts the Dual Compare Single Output Pulse mode.

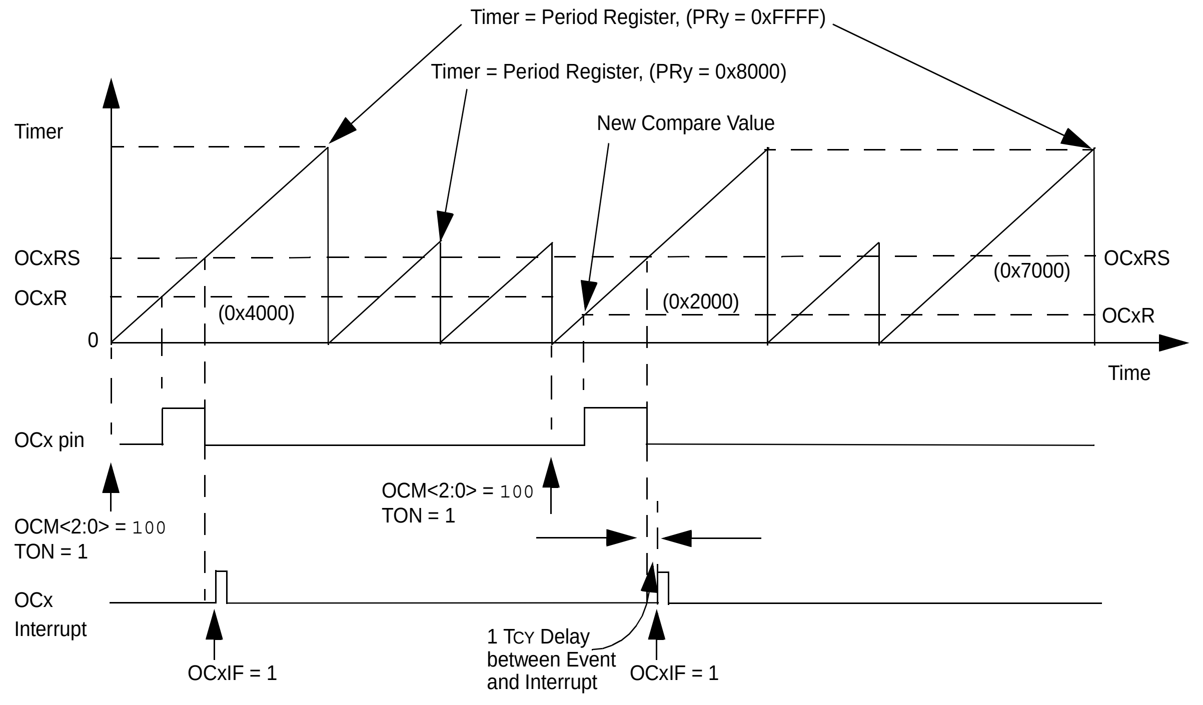
14.3.2 CONTINUOUS PULSE MODE

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 TCY.
- Calculate desired pulse value base upon TCY.
- 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.

Figure 14-3 depicts the Dual Compare Continuous Output Pulse mode.

FIGURE 14-2: DUAL COMPARE, SINGLE OUTPUT PULSE MODE

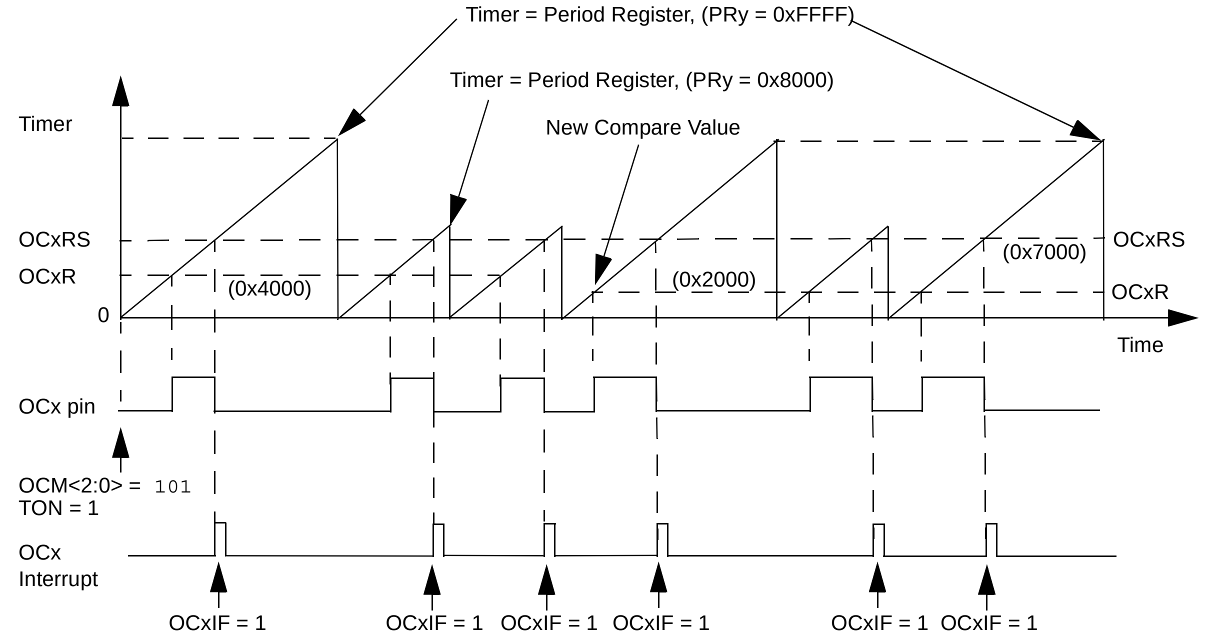


Note 1: Where 'x' is shown, reference is made to the control bits and registers associated with output compare channels 1, 2, 3 ... to N.

2: Where 'y' is shown, reference is made to Timer2 or Timer3.

3: IF bit is generated by interrupt controller module.

FIGURE 14-3: DUAL COMPARE, CONTINUOUS OUTPUT PULSE MODE



Note 1: Where ‘x’ is shown, reference is made to the control bits and registers associated with output compare channels 1, 2, 3 ... to N.

2: Where ‘y’ is shown, reference is made to Timer2 or Timer3.

3: IF bit is asserted by interrupt controller module.

14.4 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.

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.

14.4.1 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/B 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:

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

14.4.2 PWM PERIOD

The PWM period is specified by writing to the PRx register. The PWM period can be calculated using the following formula:

$$\text{PWM period} = [(\text{PRx}) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{TMRx prescale value})$$

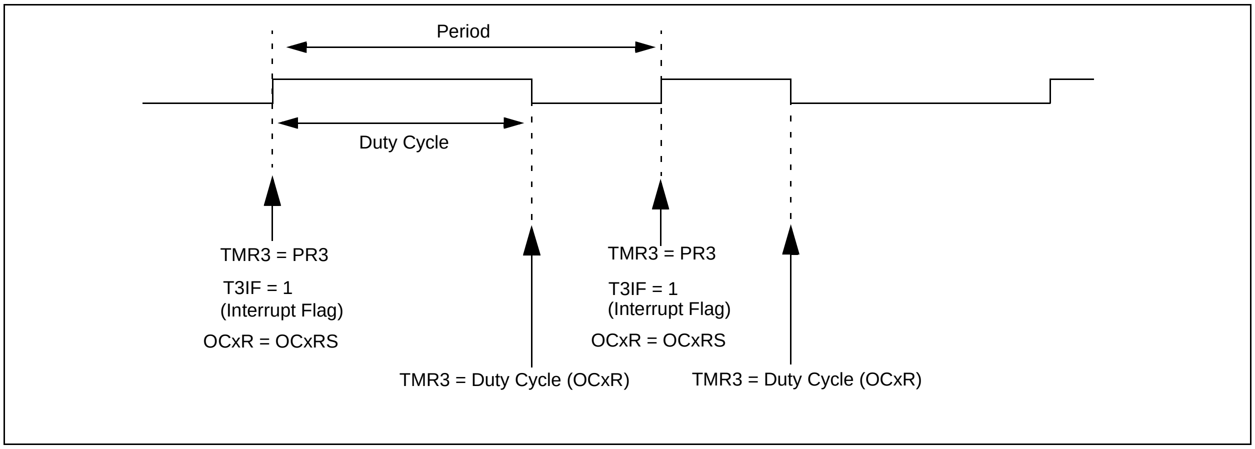
PWM frequency is defined as 1 / [PWM period].

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: if PWM duty cycle = 0%, the OCx pin will not be set).
- The PWM duty cycle is latched from OCxRS into OCxR.
- The corresponding timer interrupt flag is set.

See Figure 14-4 for key PWM period comparisons. Timer3 is referred to in the figure for clarity.

FIGURE 14-4: PWM OUTPUT TIMING



14.5 Output Compare Operation During CPU SLEEP Mode

When the CPU enters the SLEEP mode, limited functionality is provided for the output compare module. During SLEEP, all internal clocks are stopped. Therefore, the output compare module can operate if the timer clock is configured for the External Asynchronous Clock mode. In addition, the module will generate an interrupt signal when a match occurs between the timer (TMRx) and period register (PRx).

If the output compare timer source is configured for internal instruction cycle, or External Synchronized Clock mode 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.

14.6 Output Compare Operation During CPU IDLE Mode

When the CPU enters the IDLE mode, the output compare module operates 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.

The selected time-base for the output compare can be configured for:

- Internal Instruction Cycle
- External Synchronous
- External Asynchronous

14.7 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) bit is asserted and an interrupt will be generated, if enabled. The OCxIF bit is located in the corresponding IFS Status register, and must be cleared in software. The interrupt is enabled via the respective compare interrupt enable (OCxIE) bit, located in the corresponding IEC Control 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 IF bit is located in the IFS0 Status 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 Control register. The output compare interrupt flag is never set during the PWM mode of operation.

TABLE 14-1: 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>			uu0u uuuu uuu0 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	OCTSE	OCM<2:0>			uu0u uuuu uuu0 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>			uu0u uuuu uuu0 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>			uu0u uuuu uuu0 0000
OC5RS	0198	Output Compare 5 Secondary Register																0000 0000 0000 0000
OC5R	019A	Output Compare 5 Main Register																0000 0000 0000 0000
OC5CON	019C	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			uu0u uuuu uuu0 0000
OC6RS	019E	Output Compare 6 Secondary Register																0000 0000 0000 0000
OC6R	01A0	Output Compare 6 Main Register																0000 0000 0000 0000
OC6CON	01A2	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			uu0u uuuu uuu0 0000
OC7RS	01A4	Output Compare 7 Secondary Register																0000 0000 0000 0000
OC7R	01A6	Output Compare 7 Main Register																0000 0000 0000 0000
OC7CON	01A8	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			uu0u uuuu uuu0 0000
OC8RS	01AA	Output Compare 8 Secondary Register																0000 0000 0000 0000
OC8R	01AC	Output Compare 8 Main Register																0000 0000 0000 0000
OC8CON	01AE	—	—	OCSIDL	—	—	—	—	—	—	—	—	OCFLT	OCTSEL	OCM<2:0>			uu0u uuuu uuu0 0000

REGISTER 14-1: OCxCON: OUTPUT COMPARE CONTROL x REGISTER

Upper Half:

U-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
—	—	OCSIDL	—	—	—	—	—
bit 15							bit 8

Lower Half:

U-0	U-0	U-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	OCFLT	OCTSEL	OCM2	OCM1	OCM0
bit 7							bit 0

bit 15-14 **Unimplemented:** Read as '0’

bit 13 **OCSIDL:** Output Compare Stop in IDLE Control bit
`1` = Output Compare x will halt in CPU IDLE mode
`0` = Output Compare x will continue to operate in CPU IDLE mode

bit 12-5 **Unimplemented:** Read as '0’

bit 4 **OCFLT:** PWM Fault Condition Status bit
`1` = PWM fault condition has occurred (cleared in HW only)
`0` = No PWM fault condition has occurred
(Status bit is unimplemented in non-PWM mode)

bit 3 **OCTSEL:** Output Compare Timer Select bits
`1` = Timer3 is the clock source for Compare x
`0` = Timer2 is the clock source for Compare x

bit 2-0 **OCM<2:0>**: Output Compare Mode Select bits
`111` = PWM mode on OCx, fault pin enabled, (T2IF/T3IF bit is set for PWM, OCxIF is set for fault)
`110` = PWM mode on OCx, fault pin disabled, (T2IF/T3IF bit is set for PWM)
`101` = Initialize OCx pin Low, generate continuous output pulses on OCx pin, (OCxIF bit is set)
`100` = Initialize OCx pin Low, generate single output pulse on OCx pin, (OCxIF bit is set)
`011` = Initialize OCx pin Low, Compare x toggles OCx pin, (OCxIF bit is set)
`010` = Initialize OCx pin High, Compare x forces OCx pin Low, (OCxIF bit is set)
`001` = Initialize OCx pin Low, Compare x forces OCx pin High, (OCxIF bit is set)
`000` = Output Compare x off

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

Note: There is one OCxCON register for each output compare channel (x = 1, 2, ..., N).

15.0 QUADRATURE ENCODER INTERFACE (QEI) MODULE

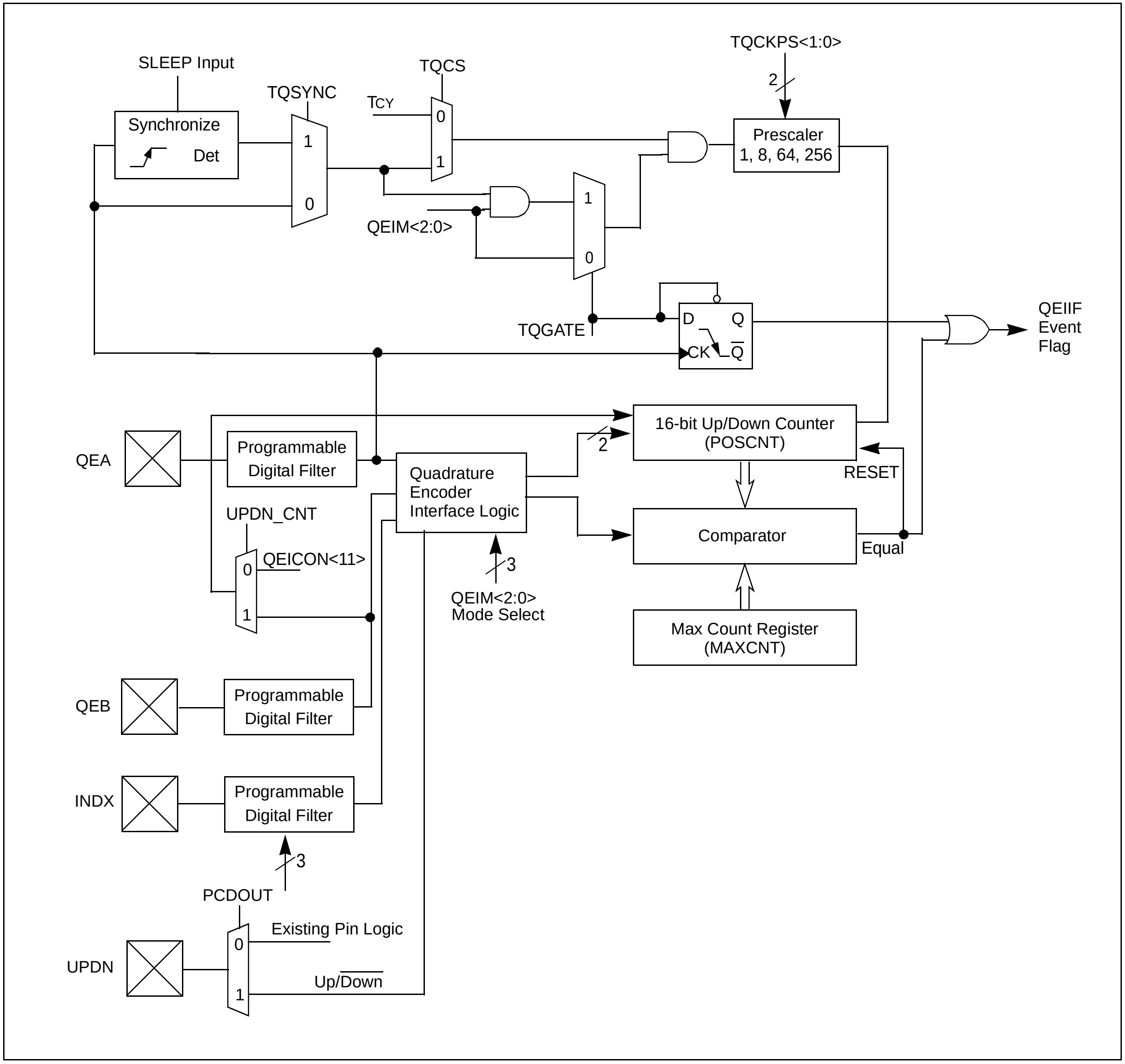
This section describes the Quadrature Encoder Interface (QEI) module and associated operational modes. The QEI module provides the interface to incremental encoders for obtaining motor positioning data. Incremental encoders are very useful in motor control applications.

The Quadrature Encoder Interface (QEI) is a key feature requirement for several motor control applications, such as Switched Reluctance (SR) and AC Induction Motor (ACIM). The operational features of the QEI are, but not limited to:

- 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 15-1 depicts the Quadrature Encoder Interface block diagram.

FIGURE 15-1: QUADRATURE ENCODER INTERFACE BLOCK DIAGRAM



15.1 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.

15.2 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.

15.3 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 a SFR bit UPDN (QEICON<11>) as a read only bit. To place the state of this signal on an I/O pin, the SFR bit PCDOUT (QEICON<6>) must be 1.

15.4 Position Measurement Mode

There are two Measurement modes which are supported and are termed x2 and x4. These modes are selected by the EIM<2:0> mode select bits located in SFR QEICON<10:8>.

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.

15.5 Programmable Digital Noise Filters

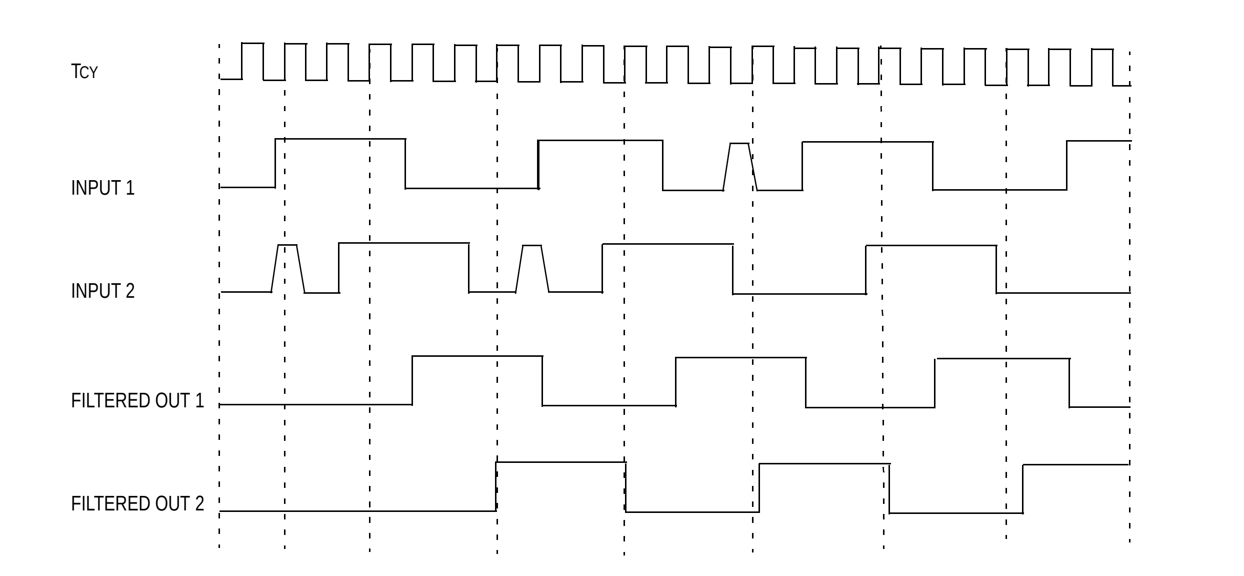
The digital noise filter section is responsible for rejecting noise on the incoming capture or quadrature signals. Schmitt Trigger inputs and a three-clock cycle delay filter combine to reject low level noise and large, short duration noise spikes that typically occur in noise prone applications, such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

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

Figure 15-2 demonstrates the effect of the digital noise filter.

FIGURE 15-2: SIGNAL RELATIONSHIP THROUGH FILTER, 1:1 CLOCK DIVIDE



15.6 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 for a simple 16-bit timer/counter. The setup and control for the auxiliary timer is accomplished through the QEICON SFR register. This timer functions identical 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 occur, 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 QEICON<11> = 1, the timer will count up. When QEICON<11> = 0, the timer will count down.

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

15.7 QEI Module Operation During CPU SLEEP Mode

15.7.1 QEI OPERATION DURING CPU SLEEP MODE

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

15.7.2 TIMER OPERATION DURING CPU SLEEP MODE

During CPU SLEEP mode, the timer will operate if:

- The timer module is enabled (TQON = 1), and
- The timer clock source is selected as external (TQCS = 0), and
- The TQSYNC bit 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.

15.8 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.

15.8.1 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.

15.8.2 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.

15.9 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
- Timer period match event (overflow/underflow)
- Gate accumulation event

The QEI interrupt flag bit, QEIIF, is asserted upon the 16-bit up/down position counter rollover/underflow, or timer period match (POSCNT = MAXCNT). The QEIIF bit must be cleared in software. QEIIF is located in the IFS2 Status register.

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

TABLE 15-1: 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	—	—	QEISIDL	INDX	UPDN	QEIM2	QEIM1	QEIM0	SWPAB	PCDOUT	TQGATE	TQCKPS1	TQCKPS0	TQSYNC	TQCS	UPDN_CNT	uu00 0000 0000 0000
DFLTCON	0124	—	—	—	—	—	—	—	—	QEOUT	QECK2	QECK1	QECK0	INDOUT	INDCK2	INDCK1	INDCK0	uuuu uuuu 0000 0000
POSCNT	0126	Position Counter<15:0>																0000 0000 0000 0000
MAXCNT	0128	Maximun Count<15:0>																0000 0000 0000 0000

REGISTER 15-1: QEICON: QEI CONTROL REGISTER

Upper Half:							
U-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	QEISIDL	INDX	UPDN	QEIM2	QEIM1	QEIM0
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SWPAB	PCDOUT	TQGATE	TQCKPS1	TQCKPS0	TQSYNC	TQCS	UPDN_CNT
bit 7							bit 0

bit 15-14 **Unimplemented:** Read as '0’

bit 13 **QEISIDL:** Module Stop in IDLE Control bit
1 = Module will halt in CPU IDLE mode
0 = Module will continue to operate CPU IDLE mode

bit 12 **INDX:** Index Pin State Status bit (Read Only)
1 = Index pin is high
0 = Index pin is low

bit 11 **UPDN:** Position Counter Direction Status bit
1 = Position counter direction is positive (+)
0 = Position counter direction is negative (-
(Read only bit in QEI mode. Read/Write bit in Timer Operation mode.)

bit 10-8 **QEIM<2:0>:** Quadrature Encoder Interface Mode Select bits
111 = Quadrature Encoder Interface enabled (x4 mode) with Position Counter Reset by match (MAXCNT)
110 = Quadrature Encoder Interface enabled (x4 mode) with Index Pulse Reset of position counter
101 = Quadrature Encoder Interface enabled (x2 mode) with Position Counter Reset by match (MAXCNT)
100 = Quadrature Encoder Interface enabled (x2 mode) with Index Pulse Reset of position counter
011 = Unused (module disabled)
010 = Unused (module disabled)
001 = Starts 16-bit Timer
000 = Quadrature Encoder Interface/Timer off

bit 7 **SWPAB:** Phase A and Phase B Input Swap Select bit
1 = Phase A and Phase B inputs swapped
0 = Phase A and Phase B inputs not swapped

bit 6 **PCDOUT:** Position Counter Direction State Output Enable bit
1 = Position counter direction status output enable
(QEI logic controls state of I/O pin)
0 = Position counter direction status output disabled
(Normal I/O pin operation.)

bit 5 **TQGATE:** Timer Gated Time Accumulation Enable bit
1 = Timer gated time accumulation enabled
0 = Timer gated time accumulation disabled

bit 4-3 **TQCKPS<1:0>:** Timer Input Clock Prescale Select bits
11 = 1:256 prescale value
10 = 1:64 prescale value
01 = 1:8 prescale value
00 = 1:1 prescale value
(Prescaler utilized for 16-bit Timer mode only.)

REGISTER 15-1: QEICON: QEI CONTROL REGISTER (Continued)

bit 2 **TQSYNC:** Timer External Clock Input Synchronization Select bit

When TQCS = 1:
1 = Synchronize external clock input
0 = Do not synchronize external clock input

When TQCS = 0:
This bit is ignored. Timer uses the internal clock when TQCS = 0.

bit 1 **TQCS:** Timer Clock Source Select bit
1 = External clock from pin TQCK (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **UPDN_CNT:** Position Counter Direction Selection Control bit
1 = QEB pin state defines position counter direction
0 = Control/Status bit, QEICON<11>, defines timer counter (POSCNT) direction

Note: When configured for QEI mode, this bit is ignored.

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

REGISTER 15-2: DFLTCON: DIGITAL FILTER CONTROL REGISTER

Upper Half:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
QEOUT	QECK2	QECK1	QECK0	INDOUT	INDCK2	INDCK1	INDCK0
bit 7							bit 0

bit 15-8 **Unimplemented:** Read as '0’

bit 7 **QEOUT:** QEA/QEB Digital Filter Output Enable bit
1 = Digital filter outputs enabled
0 = Digital filter outputs disabled (Normal pin operation)

bit 6-4 **QECK<2:0>:** QEA/QEB Digital Filter Clock Divide Select bits
111 = 1:256 Clock divide
110 = 1:128 Clock divide
101 = 1:64 Clock divide
100 = 1:32 Clock divide
011 = 1:16 Clock divide
010 = 1:4 Clock divide
001 = 1:2 Clock divide
000 = 1:1 Clock divide

bit 3 **INDOUT:** Index Channel Digital Filter Output Enable bit
1 = Digital filter output is enabled
0 = Digital filter output is disabled (normal pin operation)

bit 2-0 **INDCK<2:0>:** Index Channel Digital Filter Clock Divide Select bits

111 = 1:256 Clock divide
110 = 1:128 Clock divide
101 = 1:64 Clock divide
100 = 1:32 Clock divide
011 = 1:16 Clock divide
010 = 1:4 Clock divide
001 = 1:2 Clock divide
000 = 1:1 Clock divide

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

REGISTER 15-3: POSCNT: 16-BIT POSITION COUNTER REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
POSCNT15	POSCNT14	POSCNT13	POSCNT12	POSCNT11	POSCNT10	POSCNT9	POSCNT8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
POSCNT7	POSCNT6	POSCNT5	POSCNT4	POSCNT3	POSCNT2	POSCNT1	POSCNT0
bit 7							bit 0

bit 15-0 **POSCNT<15:0>:** 16-bit Up_Down Position Counter Register bits
(Serves as Timer/Counter register when in Timer mode)

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

REGISTER 15-4: MAXCNT: MAXIMUM COUNT REGISTER

Upper Half:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
MAXCNT15	MAXCNT14	MAXCNT13	MAXCNT12	MAXCNT11	MAXCNT10	MAXCNT9	MAXCNT8
bit 15							bit 8

Lower Half:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
MAXCNT7	MAXCNT6	MAXCNT5	MAXCNT4	MAXCNT3	MAXCNT2	MAXCNT1	MAXCNT0
bit 7							bit 0

bit 15-0 **MAXCNT<15:0>:** 16-bit Maximum Count Register bits
(Serves as Period register when in Timer mode)

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

NOTES:

16.0 MOTOR CONTROL PWM MODULE

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)

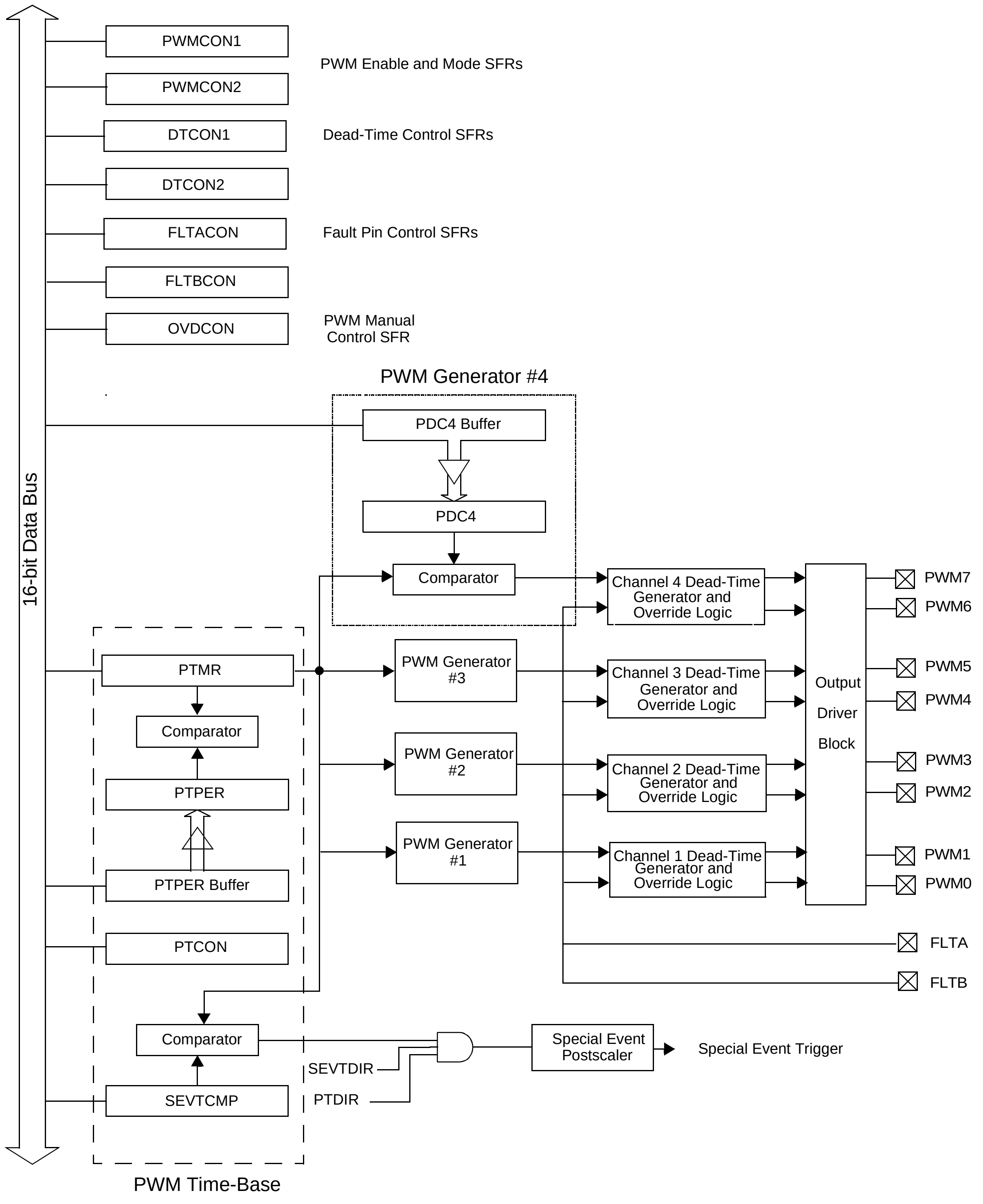
The PWM module has the following features:

- 8 PWM I/O pins with 4 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 I/O pins to a defined state

A simplified block diagram of the PWM module is shown in Figure 16-1. Figure 16-5 and Figure 16-9 show how the module hardware is partitioned for each PWM output pair for the Complementary and Independent Output modes, respectively.

This module contains 4 duty cycle generators, numbered 1 through 4. The module has 8 PWM output pins, numbered 0 through 7. The eight I/O pins are grouped into odd numbered/even numbered pairs. For complementary loads, the even PWM pins are always the complement of the corresponding odd I/O pin.

FIGURE 16-1: PWM MODULE BLOCK DIAGRAM



Note: Details of PWM Generator #1, #2, and #3 not shown for clarity.

The PWM module allows several modes of operation, which are beneficial for specific power control applications. Each mode of operation is described subsequently.

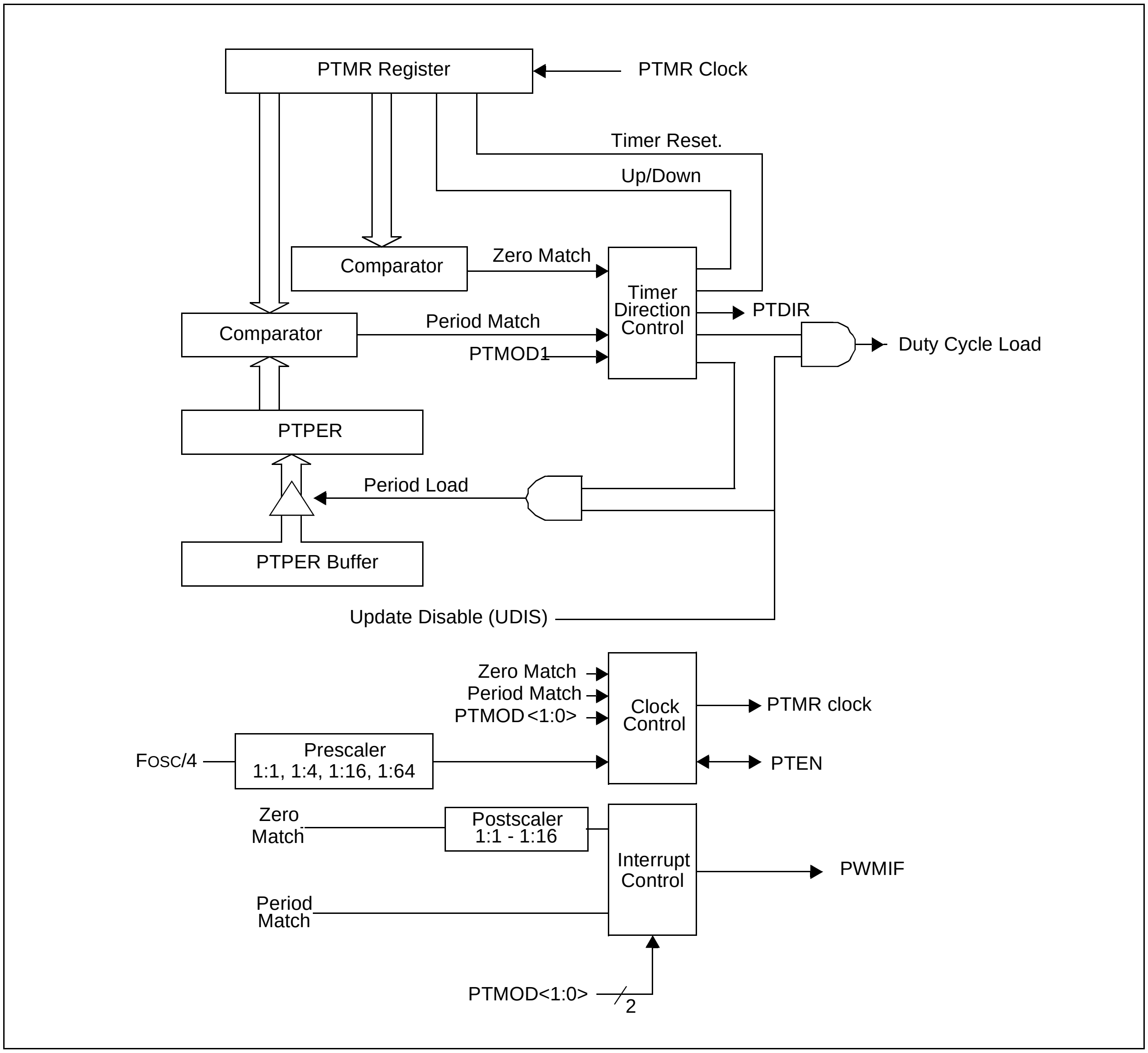
16.1 PWM Time-Base

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<15:0> contains a read only status bit, PDIR, that indicates the present count direction of the PWM time-base. If PTDIR is cleared, PTMR is counting upwards, whereas PTDIR set, indicates that PTMR is counting downwards. The PWM time-base is configured via the PTCON SFR. The time-base is enabled/disabled by setting/clearing the PTEN bit in the PTCON 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 the 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.

FIGURE 16-2: PWM TIME-BASE BLOCK DIAGRAM



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 PTCON SFR. The Up/Down Counting modes support center aligned PWM generation. The Single Shot mode allows the PWM module to support pulse control of certain Electronically Commutative Motors (ECMs).

16.1.1 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.

16.1.2 SINGLE SHOT MODE

In the Single Shot Counting 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.

16.1.3 CONTINUOUS UP/DOWN COUNTING MODES

In the Continuous Up/Down Counting 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 PTCON SFR is read only and indicates the counting direction The PTDIR bit is set when the timer counts downwards.

16.1.4 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.

16.1.5 PWM TIME-BASE POSTSCALER

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

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.

16.2 PWM Time-Base Interrupts

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 PTCON SFR.

16.2.1 FREE RUNNING MODE

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.

16.2.2 SINGLE SHOT MODE

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**.**

16.2.3 CONTINUOUS UP/DOWN COUNTING MODE

In the Up/Down Counting 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.

16.2.4 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. Secondly, 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.

16.3 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.
- Up/Down Counting 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 from the following formula:

EQUATION 16-1: PWM PERIOD

$$P_{PWM} = \frac{4(PTPER + 1)}{F_{OSC} \bullet (PTMRprescalevalue)}$$

If the PWM time-base is configured for one of the Up/Down Count modes, the PWM period will be twice the value provided by Equation 16-1.

The maximum resolution (in bits) for a given device oscillator and PWM frequency can be determined from the following formula:

EQUATION 16-2: PWM RESOLUTION

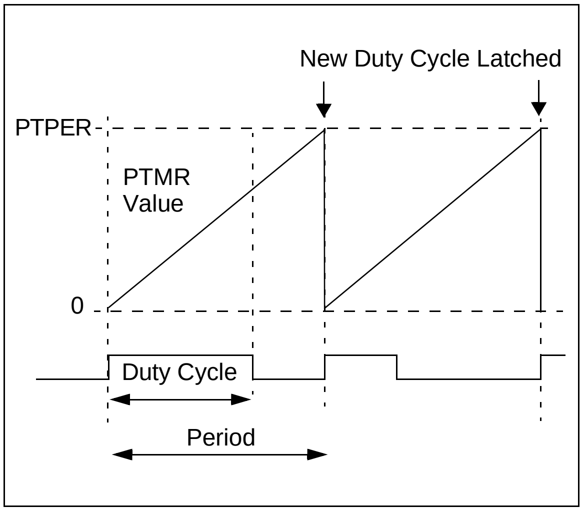
$$\text{Resolution} = \frac{\log\left(\frac{F_{OSC}}{2 \bullet Fpwm}\right)}{\log(2)}$$

16.4 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 for a given PWM channel has a period specified by the value loaded in PTPER and a duty cycle specified by the appropriate duty cycle register (see Figure 16-3). 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 16-3: EDGE ALIGNED PWM



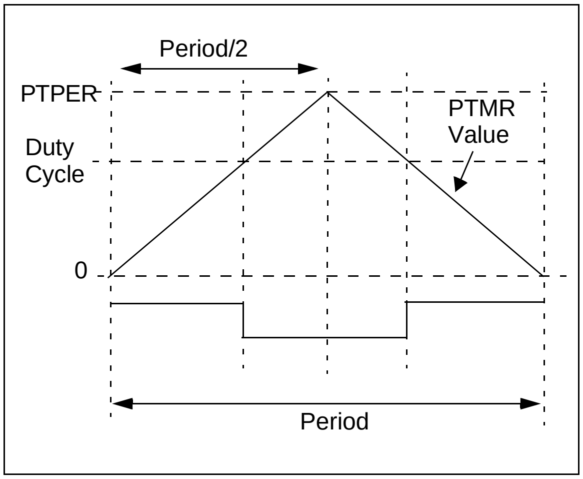
16.5 Center Aligned PWM

Center aligned PWM signals are produced by the module when the PWM time-base is configured in an Up/Down Counting mode (see Figure 16-4).

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 16-4: CENTER ALIGNED PWM



16.6 PWM Duty Cycle Comparison Units

There are four 16-bit special function registers used to specify duty cycle values for the PWM module:

- PDC1
- PDC2
- PDC3
- PDC4

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 LS bit of a duty cycle register determines whether the PWM edge occurs on Q1 or Q3. Thus, the PWM resolution is effectively doubled.

16.6.1 DUTY CYCLE REGISTER BUFFERS

The four PWM duty cycle registers are double buffered to allow glitchless updates of the PWM outputs. For each duty cycle, there is a duty cycle buffer 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 Up/Down Counting 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 Up/Down Counting 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).

16.7 Complementary PWM Operation

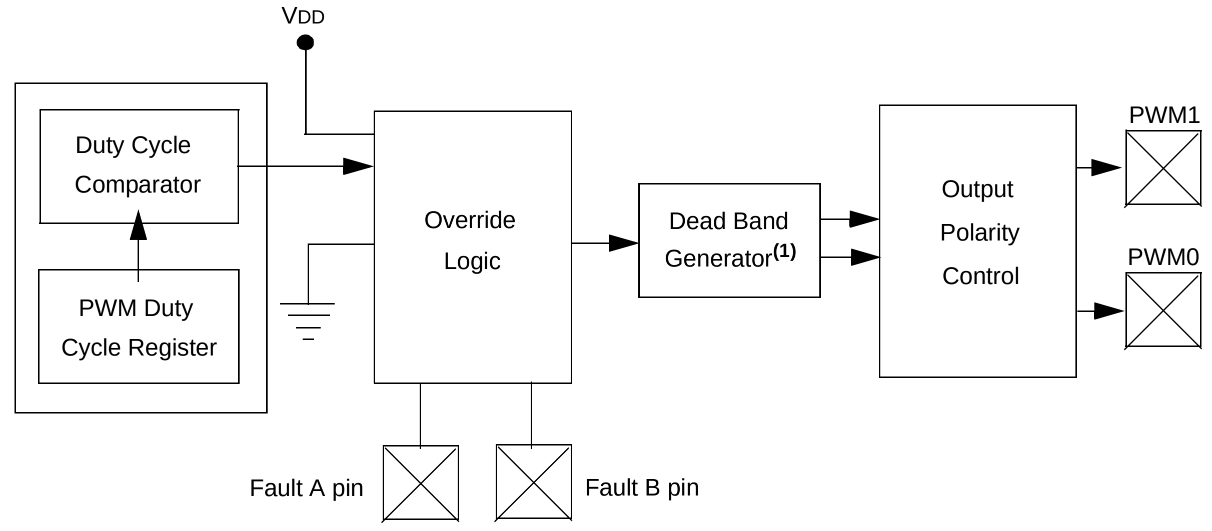
In the Complementary mode of operation, each pair of PWM outputs is fed by a complementary PWM signal. A dead-time may be optionally inserted during device switching, when both outputs are inactive for a short period (Refer to Section 16.8).

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

- PDC1 register controls PWM1/PWM0 outputs
- PDC2 register controls PWM3/PWM2 outputs
- PDC3 register controls PWM5/PWM4 outputs
- PDC4 register controls PWM7/PWM6 outputs

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

FIGURE 16-5: PWM MODULE BLOCK DIAGRAM, ONE OUTPUT PAIR, COMPLEMENTARY MODE



Note 1: In the Complementary mode, the even channel cannot be forced active by a fault or override event, when the odd channel is active.

16.8 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.

16.8.1 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 16-8, each dead-time unit has a rising and falling edge detector connected to the duty cycle comparison output.

16.8.2 DEAD-TIME ASSIGNMENT

The DTCON2 SFR contains control bits that allow the dead-times to be assigned to each of the complementary outputs. Table 16-1 summarizes the function of each dead-time selection control bit.

TABLE 16-1: DEAD-TIME SELECTION BITS

Bit	Function
DTS1R	Selects PWM0/PWM1 rising edge dead-time.
DTS1F	Selects PWM0/PWM1 falling edge dead-time.
DTS2R	Selects PWM2/PWM3 rising edge dead-time.
DTS2F	Selects PWM2/PWM3 falling edge dead-time.
DTS3R	Selects PWM4/PWM5 rising edge dead-time.
DTS3F	Selects PWM4/PWM5 falling edge dead-time.
DTS4R	Selects PWM6/PWM7 rising edge dead-time.
DTS4F	Selects PWM6/PWM7 falling edge dead-time.

16.8.3 DEAD-TIME RANGES

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

Four input clock prescaler selections have been provided to allow a suitable range of dead-times, based on the device operating frequency. The clock prescaler option may be selected independently for each of the two dead-time values. The dead-time clock prescaler values are selected using the DTAPS<1:0> and DTBPS<1:0> control bits in the DTCON1 SFR. The following clock prescaler options may be selected for each of the dead-time values:

- FOSC/4
- FOSC/8
- FOSC/16
- FOSC/32

After the prescaler values are selected, the dead-time for each unit is adjusted by loading two 6-bit unsigned values into the DTCON1 special function register.

The dead-time unit prescalers are 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 or DTCON2 registers.
- On any device RESET.

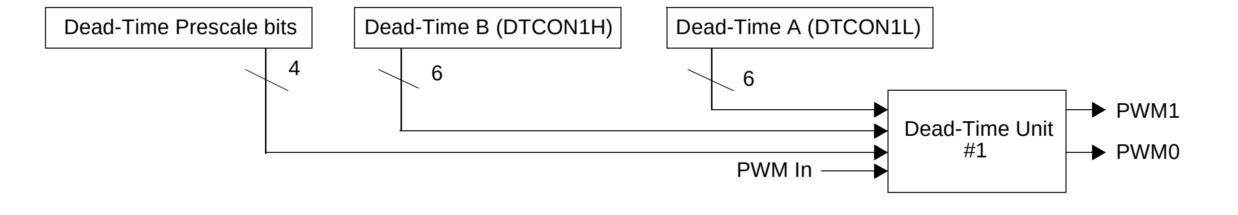
Note: The user should not modify the DTCON1 or DTCON2 values while the PWM module is operating (PTEN = 1). Unexpected results may occur.

16.8.4 DEAD-TIME DISTORTION

For small PWM duty cycles, the ratio of dead-time to the active PWM time may become large. In this case, the inserted dead-time will introduce distortion into waveforms produced by the PWM module. The user can ensure that dead-time distortion is minimized by keeping the PWM duty cycle at least three times larger than the dead-time (see Figure 16-7).

A similar effect occurs for duty cycles at or near 100%. The maximum duty cycle used in the application should be chosen such that the minimum inactive time of the signal is at least three times larger than the dead-time.

FIGURE 16-6: DEAD-TIME CONTROL UNITS BLOCK DIAGRAM



Note: There are 3 other Dead-Time Units, corresponding to PWM7/6, PWM5/4, and PWM 3/2, respectively.

FIGURE 16-7: DEAD-TIME TIMING DIAGRAM

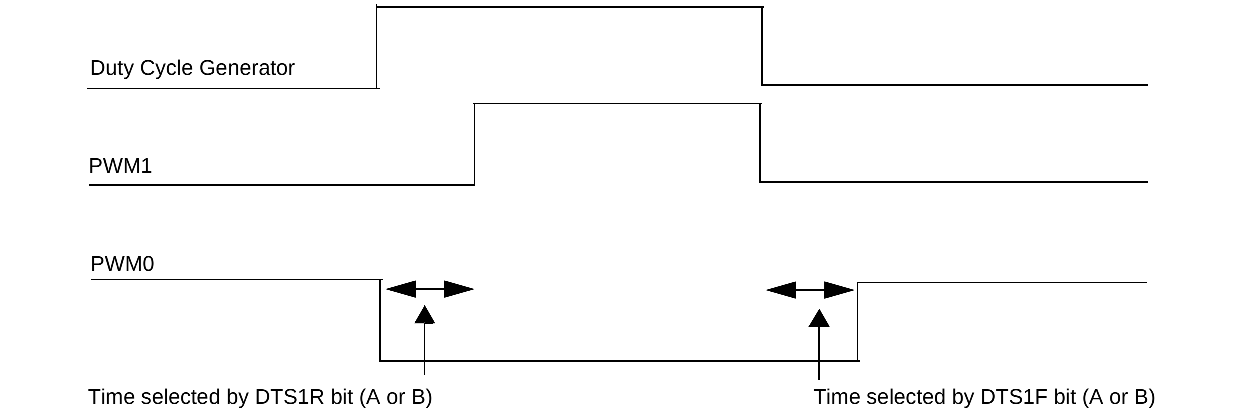
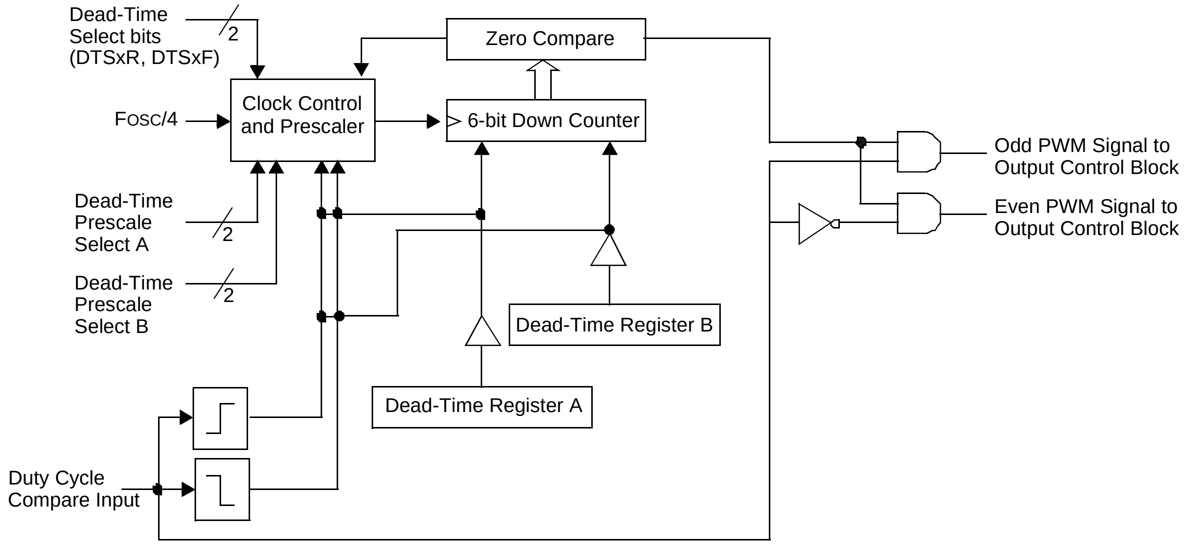

Duty Cycle Generator
PWM1
PWM0
Time selected by DTS1R bit (A or B)
Time selected by DTS1F bit (A or B)

FIGURE 16-8: DEAD-TIME CONTROL UNIT BLOCK DIAGRAM FOR ONE PWM OUTPUT PAIR


Dead-Time Select bits (DTSxR, DTSxF)
2
Zero Compare
FOSC/4
Clock Control and Prescaler
6-bit Down Counter
Dead-Time Prescale Select A
2
Dead-Time Prescale Select B
2
Odd PWM Signal to Output Control Block
Even PWM Signal to Output Control Block
Dead-Time Register B
Dead-Time Register A
Duty Cycle Compare Input

Note: 'x' represents Dead-Time Units 1, 2, 3, and 4.

16.9 Independent PWM Output

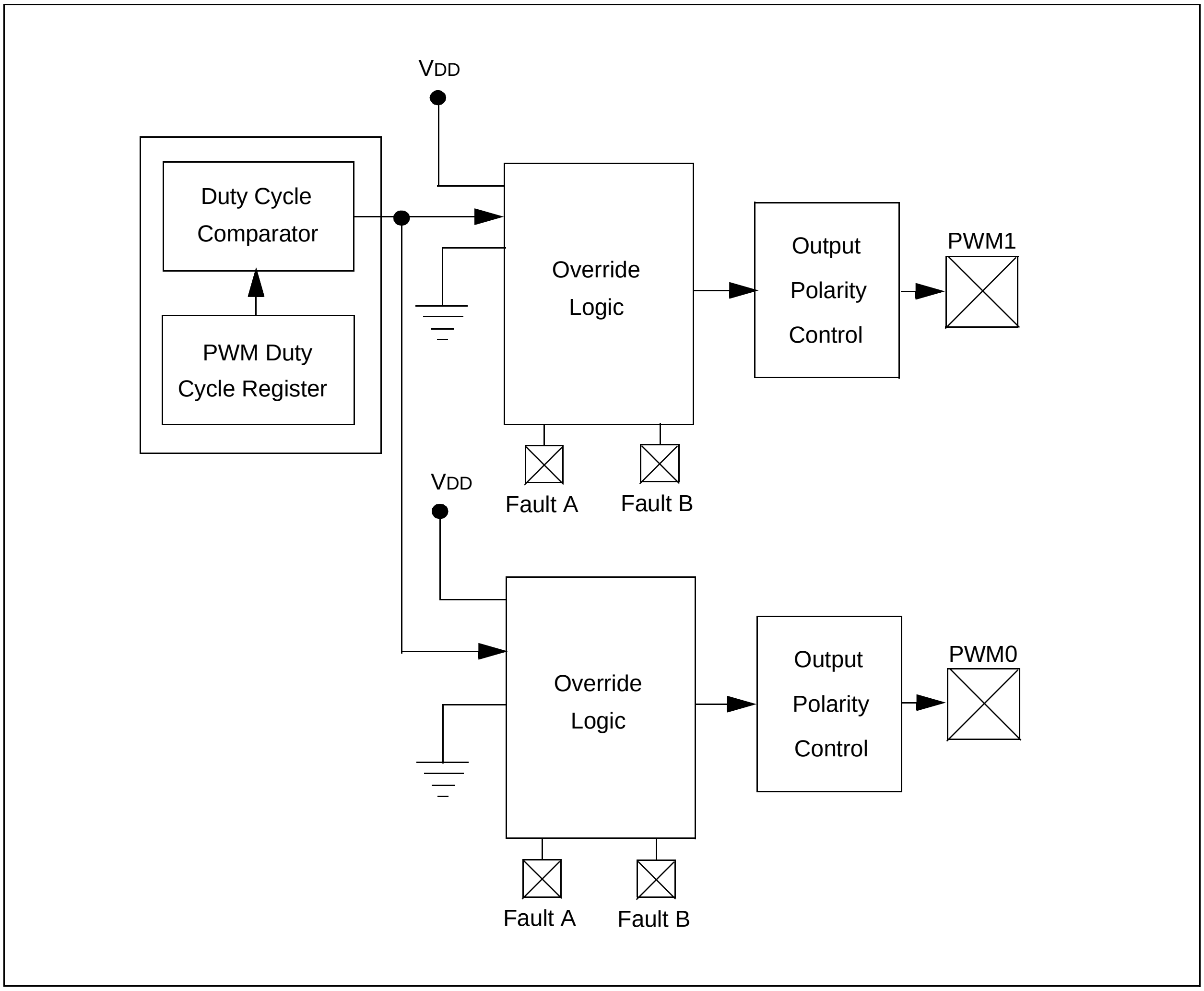
An independent PWM Output mode is required for driving certain types of loads. A particular PWM output pair is in the Independent Output mode when the corresponding PMOD bit in the PWMCON1 register is set. No dead-time control is implemented between adjacent PWM I/O pins when the module is operating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

In the Independent mode, each duty cycle generator is connected to both of the PWM I/O pins in an output pair (see Figure 16-9). 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 mode:

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

Refer to Register 16-11 for details on the OVDCON SFR.

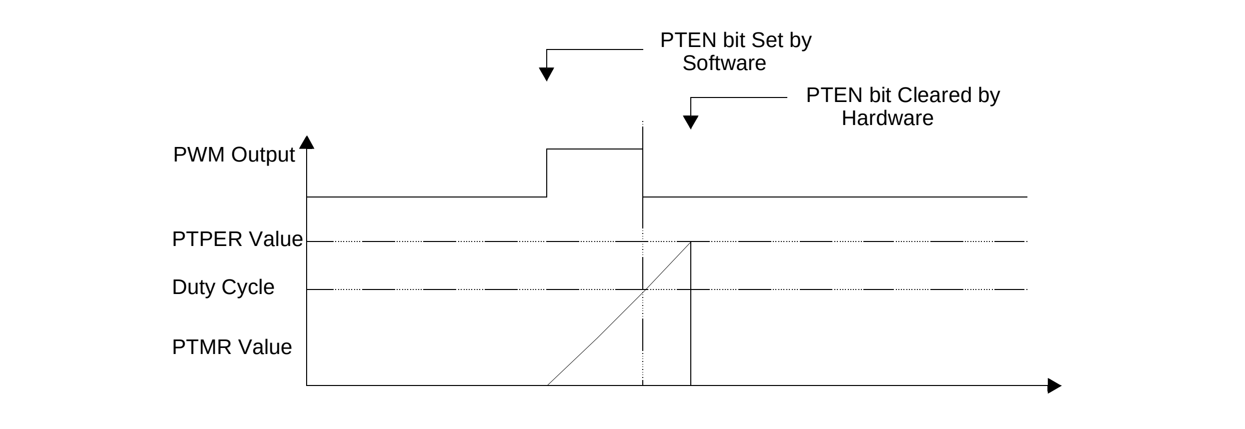
FIGURE 16-9: PWM MODULE BLOCK DIAGRAM, ONE OUTPUT PAIR, INDEPENDENT MODE



16.10 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 (see Figure 16-10).

FIGURE 16-10: SINGLE PULSE PWM OPERATION



16.11 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 eight bits, POVD<7:0>, that determine which PWM I/O pins will be overridden. The lower half of the OVDCON register contains eight bits, POUT<7:0>, that determine the state of the PWM I/O pins when a particular output is overridden, via the POVD bits.

16.11.1 COMPLEMENTARY OUTPUT MODE

When the even numbered pin is driven active via the OVDCON register, the output signal is forced to be the complement of the odd numbered I/O pin in the pair (see Figure 16-5 for details.) However, dead-time insertion is still performed when PWM channels are overridden manually.

16.11.2 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.

16.12 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 configuration bits (see Section 23.0) work in conjunction with the four PWM enable bits (PWMEN<4:1>), located in the PWMCON1 SFR. 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 outputs 1, 3, 5 and 7, whereas the LPOL bit specifies the polarity for outputs 0, 2, 4 and 6.

16.12.1 OUTPUT PIN CONTROL

The PWMEN<4:1> control bits in the PWMCON1 SFR enable each PWM output pin pair for use by the module. Each PWMEN bit controls the following output pairs:

- PWMEN1 enables PWM0/PWM1 output pair
- PWMEN2 enables PWM2/PWM3 output pair
- PWMEN3 enables PWM4/PWM5 output pair
- PWMEN4 enables PWM6/PWM7 output pair

16.13 PWM Fault Pins

There are two fault pins (FLTA and FLTB) associated with the PWM module. When asserted, these pins can optionally drive each of the PWM I/O pins to a defined state.

16.13.1 FAULT PIN ENABLE BITS

The FLTACON and FLTBCON Special Function Registers each have 4 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 or FLTBCON register.

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

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

16.13.2 FAULT STATES

The FLTACON and FLTBCON special function registers have 8 bits each, 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 odd numbered PWM I/O pin always has priority in the Complementary mode, so that both I/O pins cannot be driven active simultaneously. Refer to Figure 16-5 for details.

16.13.3 FAULT PIN PRIORITY

If both fault input pins have been assigned to control a particular PWM I/O pin, the fault state programmed for the Fault A input pin will take priority over the Fault B input pin.

16.13.4 FAULT INPUT MODES

Each of the fault input pins 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/FLTBCON 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 each fault input pin is selected using the FLTAM and FLTBM control bits in the FLTACON and FLTBCON Special Function Registers.

Each of the fault pins can be controlled manually in software.

16.14 PWM Update Lockout

For a complex PWM application, the user may need to write up to four 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 may optionally be enabled so that the user may specify when new duty cycle buffer values are valid. 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.

16.15 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 an Up/Down Counting 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 an Up/Down Counting mode.

16.15.1 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.

16.16 PWM Operation During CPU SLEEP Mode

The Fault A and Fault B input pins have the ability to wake the CPU from SLEEP mode. The PWM module generates an interrupt if either of the fault pins is driven low while in SLEEP.

16.17 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 continues to operate. If PTSIDL = 1, the module will stop operation as long as the CPU remains in IDLE mode.

TABLE 16-2: MOTOR CONTROL 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>		0u00 uuuu 0000 0000
PTMR	01C2	PTDIR	PWM Timer Count Value															0000 0000 0000 0000
PTPER	01C4	—	PWM Time-Base Period Register															u000 0000 0000 0000
SEVTCMP	01C6	SEVTDIR	PWM Special Event Compare Register															0000 0000 0000 0000
PWMCON1	01C8	—	—	—	—	PTMOD4	PTMOD3	PTMOD2	PTMOD1	—	—	—	—	PWMEN4	PWMEN3	PWMEN2	PWMEN1	uuuu 1111 uuuu 0000
PWMCON2	01CA	—	—	—	—	SEVOPS<3:0>				—	—	—	—	—	—	OSYNC	UDIS	uuuu 0000 uuuu uu00
DTCON1	01CC	DTBPS<1:0>		Dead-Time B Value						DTAPS<1:0>		Dead-Time A Value						0000 0000 0000 0000
DTCON2	01CE	—	—	—	—	—	—	—	—	DTS4R	DTS4F	DTS3R	DTS3F	DTS2R	DTS2F	DTS1R	DTS1F	uuuu uuuu 0000 0000
FLTACON	01D0	FAOV7	FAOV6	FAOV5	FAOV4	FAOV3	FAOV2	FAOV1	FAOV0	FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1	0000 0000 0uuu 0000
FLTBCON	01D2	FBOV7	FBOV6	FBOV5	FBOV4	FBOV3	FBOV2	FBOV1	FBOV0	FLTCM	—	—	—	FBEN4	FBEN3	FBEN2	FBEN1	0000 0000 0uuu 0000
OVDCON	01D4	POVD7	POVD6	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0	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
PDC4	01DC	PWM Duty Cycle #4 Register																0000 0000 0000 0000

REGISTER 16-1: PTCON: PWM TIME-BASE CONTROL REGISTER

Upper Half:							
R/W-0	R-0	R/W-0	U-0	U-0	U-0	U-0	U-0
PTEN	—	PTSIDL	—	—	—	—	—
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTOPS3	PTOPS2	PTOPS1	PTOPS0	PTCKPS1	PTCKPS0	PTMOD1	PTMOD0
bit 7							bit 0

bit 15 **PTEN:** PWM Time-Base Timer Enable bit
1 = PWM time-base is ON
0 = PWM time-base is OFF

bit 14 **Unimplemented**: Read as ‘0’. User software should write 0 to this bit

bit 13 **PTSIDL:** PWM Time-Base Stop in IDLE Mode bit
1 = PWM time-base halts in CPU IDLE mode
0 = PWM time-base runs in CPU IDLE mode

bit 12-8 **Unimplemented**: Read as ‘0’. User software should write 0 to these bits.

bit 7-4 **PTOPS<3:0>:** PWM Time-Base Output Postscale Select bits
1111 = 1:16 Postscale
1110 = 1:8 Postscale
| |
| |
0000 = 1:1 Postscale

bit 3-2 **PTCKPS<1:0>:** PWM Time-Base Input Clock Prescale Select bits
11 = PWM time-base input clock is FOSC/256 (1:64 prescale)
10 = PWM time-base input clock is FOSC/64 (1:16 prescale)
01 = PWM time-base input clock is FOSC/16 (1:4 prescale)
00 = PWM time-base input clock is FOSC/4 (1:1 prescale)

bit 1-0 **PTMOD<1:0>:** PWM Time-Base Mode Select bits
11 = PWM time-base operates in a Continuous Up/Down mode with interrupts for double PWM updates
10 = PWM time-base operates in a Continuous Up/Down counting mode
01 = PWM time-base configured for Single Shot mode
00 = PWM time-base operates in a Free Running mode

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

REGISTER 16-2: PTMR: PWM TIME-BASE COUNT REGISTER

Upper Half:							
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTDIR	PTMR14	PTMR13	PTMR12	PTMR11	PTMR10	PTMR9	PTMR8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTMR7	PTMR6	PTMR5	PTMR4	PTMR3	PTMR2	PTMR1	PTMR0
bit 7							bit 0

bit 15 **PTDIR:** PWM Time-Base Count Direction Status bit
1 = PWM time-base counts down
0 = PWM time-base counts up

bit 14-0 **PTMR<14:0>:** PWM Time-Base Count Value bits

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

REGISTER 16-3: PTPER: PWM TIME-BASE PERIOD REGISTER

Upper Half:							
U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	PTPER14	PTPER13	PTPER12	PTPER11	PTPER10	PTPER9	PTPER8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTPER7	PTPER6	PTPER5	PTPER4	PTPER3	PTPER2	PTPER1	PTPER0
bit 7							bit 0

bit 15 **Unimplemented**: Read as ‘0’. User software should write 0 to this bit.

bit 14-0 **PTPER<14:0>:** PWM Time-Base Period Value bits

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

REGISTER 16-4: SEVTCMP: SPECIAL EVENT TRIGGER COMPARE COUNT REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTDIR	SEVTCMP14	SEVTCMP13	SEVTCMP12	SEVTCMP11	SEVTCMP10	SEVTCMP9	SEVTCMP8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEVTCMP7	SEVTCMP6	SEVTCMP5	SEVTCMP4	SEVTCMP3	SEVTCMP2	SEVTCMP1	SEVTCMP0
bit 7							bit 0

bit 15 **SEVTDIR:** Special Event Trigger Time-Base Direction bit
1 = A special event trigger will occur when the PWM time-base is counting downwards
0 = A special event trigger will occur when the PWM time-base is counting upwards

bit 14-0 **SEVTCMP<14:0>:** Special Event Compare Count Value bits

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

REGISTER 16-5: PWMCON1: PWM CONTROL REGISTER1

Upper Half:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	PMOD4	PMOD3	PMOD2	PMOD1
bit 15							bit 8

Lower Half:							
U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	—	PWMEN4	PWMEN3	PWMEN2	PWMEN1
bit 7							bit 0

bit 15-12 **Unimplemented**: Read as ‘0’. User software should write 0 to these bits.

bit 11-8 **PMOD<4:1>:** PWM I/O Pair Mode bits
1 = PWM I/O pin pair is in the Independent Output mode
0 = PWM I/O pin pair is in the Complementary Output mode

bit 7-4 **Unimplemented**: Read as ‘0’. User software should write 0 to these bits.

bit 3-0 **PWMEN<4:1>:** PWM I/O Pair Enable bits
1 = PWM I/O pin pair is enabled for PWM output
0 = PWM I/O pin pair disabled. I/O pins becomes general purpose I/O.

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

REGISTER 16-6: PWMCON2: PWM CONTROL REGISTER2

Upper Half:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	SEVOPS3	SEVOPS2	SEVOPS1	SEVOPS0
bit 15							bit 8

Lower Half:							
U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	OSYNC	UDIS
bit 7							bit 0

bit 15-12 **Unimplemented**: Read as ‘0’. User software should write 0 to these bits.

bit 11-8 **SEVOPS<3:0>:** PWM Special Event Trigger Output Postscale Select bits
`1111` = 1:16 Postscale
`1110` = 1:8 Postscale
||
||
`0000` = 1:1 Postscale

bit 7-2 **Unimplemented**: Read as ‘0’. User software should write 0 to these bits.

bit 1 **OSYNC:** Output Override Synchronization bit
`1` = Output overrides via the OVDCON register are synchronized to the PWM time-base
`0` = Output overrides via the OVDCON register are asynchronous

bit 0 **UDIS:** PWM Update Disable bit
`1` = Updates from duty cycle and period buffer registers are disabled
`0` = Updates from duty cycle and period buffer registers are enabled

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

REGISTER 16-7: DTCON1: DEAD-TIME CONTROL REGISTER1

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTBPS1	DTBPS0	DTB5	DTB4	DTB3	DTB2	DTB1	DTB0
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTAPS1	DTAPS0	DTA5	DTA4	DTA3	DTA2	DTA1	DTA0
bit 7							bit 0

bit 15-14 **DTBPS<1:0>:** Dead-Time Unit B Prescale Select bits
11 = Clock source for dead-time Unit B is FOSC/32
10 = Clock source for dead-time Unit B is FOSC/16
01 = Clock source for dead-time Unit B is FOSC/8
00 = Clock source for dead-time Unit B is FOSC/4

bit 13-8 **DTB<5:0>:** Unsigned 6-bit Dead-Time Value bits for Dead-Time Unit B

bit 7-6 **DTAPS<1:0>:** Dead-Time Unit A Prescale Select bits
11 = Clock source for dead-time Unit A is FOSC/32
10 = Clock source for dead-time Unit A is FOSC/16
01 = Clock source for dead-time Unit A is FOSC/8
00 = Clock source for dead-time Unit A is FOSC/4.

bit 5-0 **DTA<5:0>:** Unsigned 6-bit Dead-Time Value bits for Dead-Time Unit A

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

REGISTER 16-8: DTCON2: DEAD-TIME CONTROL REGISTER2

Upper Half:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTS4R	DTS4F	DTS3R	DTS3F	DTS2R	DTS2F	DTS1R	DTS1F
bit 7							bit 0

bit 15-8 **Unimplemented**: Read as ‘0’. User software should write 0 to these bits.

bit 7 **DTS4R:** Dead-Time Select bit
1 = Dead-time is provided from Unit B
0 = Dead-time is provided from Unit A

bit 6 **DTS4F:** Dead-Time Select bit
1 = Dead-time is provided from Unit B
0 = Dead-time is provided from Unit A

bit 5 **DTS3R:** Dead-Time Select bit
1 = Dead-time is provided from Unit B
0 = Dead-time is provided from Unit A

bit 4 **DTS3F:** Dead-Time Select bit
1 = Dead-time is provided from Unit B
0 = Dead-time is provided from Unit A

bit 3 **DTS2R:** Dead-Time Select bit
1 = Dead-time is provided from Unit B
0 = Dead-time is provided from Unit A

bit 2 **DTS2F:** Dead-Time Select bit
1 = Dead-time is provided from Unit B
0 = Dead-time is provided from Unit A

bit 1 **DTS1R:** Dead-Time Select bit
1 = Dead-time is provided from Unit B
0 = Dead-time is provided from Unit A

bit 0 **DTS1F:** Dead-Time Select bit
1 = Dead-time is provided from Unit B
0 = Dead-time is provided from Unit A

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

REGISTER 16-9: FLTACON: FAULT A CONTROL REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FAOV7	FAOV6	FAOV5	FAOV4	FAOV3	FAOV2	FAOV1	FAOV0
bit 15							bit 8

Lower Half:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1
bit 7							bit 0

bit 15-8 **FAOV<7:0>:** Fault Input A PWM Override Value bits
1 = The PWM output pin is driven ACTIVE on an external fault input event
0 = The PWM output pin is driven INACTIVE on an external fault input event

bit 7 **FLTAM:** Fault A Mode bit
1 = The Fault A input pin functions in the Cycle-by-Cycle Limit mode
0 = The Fault A input pin latches all control pins to the programmed states in FAOV7:FAOV0

bit 6-4 **Unimplemented**. Read as ‘0’. User software should write 0 to these bits.

bit 3-0 **FAEN<4:1>:** Fault Input A Enable bits
1 = PWM I/O pin pair is controlled by Fault Input A
0 = PWM I/O pin pair is not controlled by Fault Input A

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

REGISTER 16-10: FLTBCON: FAULT B CONTROL REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FBOV7	FBOV6	FBOV5	FBOV4	FBOV3	FBOV2	FBOV1	FBOV0
bit 15							bit 8

Lower Half:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
FLTBM	—	—	—	FBEN4	FBEN3	FBEN2	FBEN1
bit 7							bit 0

bit 15-8 **FBOV<7:0>:** Fault Input B PWM Override Value bits
1 = The PWM output pin is driven ACTIVE on an external fault input event
0 = The PWM output pin is driven INACTIVE on an external fault input event

bit 7 **FLTBM:** Fault B Mode bit
1 = The Fault B input pin functions in the Cycle-by-Cycle Limit mode
0 = The Fault B input pin latches all control pins to the programmed states in FBOV7:FBOV0

bit 6-4 **Unimplemented:** Read as 0. User software should write 0 to these bits.

bit 3-0 **FBEN<4:1>:** Fault Input B Enable bits
1 = PWM I/O pin pair is controlled by Fault Input A
0 = PWM I/O pin pair is not controlled by Fault Input A

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

REGISTER 16-11: OVDCON: OUTPUT OVERRIDE CONTROL REGISTER

Upper Half:							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
POVD7	POVD6	POVD5	POVD4	POVD3	POVD2	POVD1	POVD0
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
bit 7							bit 0

bit 15-8 **POVD<7:0>:** PWM Output Override bits
1 = Output on PWM I/O pin is controlled by the value in the duty cycle register and the PWM time-base
0 = Output on PWM I/O pin is controlled by the value in the corresponding POUT bit

bit 7-0 **POUT<7:0> :** PWM Manual Output bits
1 = Output on PWM I/O pin is driven active HIGH when the corresponding PWM output override bit is cleared
0 = Output on PWM I/O pin is driven active LOW when the corresponding PWM output override bit is cleared

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

REGISTER 16-12: PDC1: PWM DUTY CYCLE 1 REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1DC15	P1DC14	P1DC13	P1DC12	P1DC11	P1DC10	P1DC9	P1DC8
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1DC7	P1DC6	P1DC5	P1DC4	P1DC3	P1DC2	P1DC1	P1DC0
bit 7							bit 0

bit 15-0 **P1DC<15:0>:** PWM Duty Cycle #1 Value bits

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

REGISTER 16-13: PDC2: PWM DUTY CYCLE 2 REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P2DC15	P2DC14	P2DC13	P2DC12	P2DC11	P2DC10	P2DC9	P2DC8
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P2DC7	P2DC6	P2DC5	P2DC4	P2DC3	P2DC2	P2DC1	P2DC0
bit 7							bit 0

bit 15-0 **P2DC<15:0>:** PWM Duty Cycle #2 Value bits

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

REGISTER 16-14: PDC3: PWM DUTY CYCLE 3 REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P3DC15	P3DC14	P3DC13	P3DC12	P3DC11	P3DC10	P3DC9	P3DC8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P3DC7	P3DC6	P3DC5	P3DC4	P3DC3	P3DC2	P3DC1	P3DC0
bit 7							bit 0

bit 15-0 **P3DC<15:0>:** PWM Duty Cycle #3 Value bits

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

REGISTER 16-15: PDC4: PWM DUTY CYCLE 4 REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P4DC15	P4DC14	P4DC13	P4DC12	P4DC11	P4DC10	P4DC9	P4DC8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P4DC7	P4DC6	P4DC5	P4DC4	P4DC3	P4DC2	P4DC1	P4DC0
bit 7							bit 0

bit 15-0 **P4DC<15:0>:** PWM Duty Cycle #4 Value bits

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

NOTES:

17.0 SPI MODULE

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

17.1 Operating Function Description

Each SPI module consists of a 16-bit shift register, SPIxSR (where x = 1 or 2), used for shifting data in and out, and a buffer register, SPIxBUF. A control register, SPIxCON, configures the module. Additionally, a status register, SPIxSTAT, indicates various status conditions.

The serial interface consists of 4 pins: SDIx (serial data input), SDOx (serial data output), SCKx (shift clock input or output), and SSx (active low slave select).

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

A series of eight (8) or sixteen (16) clock pulses shifts out bits from the SPIxSR to SDOx pin and simultaneously shifts in data from SDIx pin. An interrupt is generated when the transfer is complete and the corresponding interrupt flag bit (SPI1IF or SPI2IF) is set. This interrupt can be disabled through an interrupt enable bit (SPI1IE or SPI2IE).

The receive operation is double buffered. When a complete byte is received, it is transferred from SPIxSR to SPIxBUF.

If the receive buffer is full when new data is being transferred from SPIxSR to SPIxBUF, the module will set the SPIROV bit, indicating an overflow condition. The transfer of the data from SPIxSR to SPIxBUF 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 SPIxBUF is read by user software.

Transmit writes are also double buffered. The user writes to SPIxBUF. When the master or slave transfer is completed, the SPIxSR is swapped with SPIxBUF. The received data is thus placed in SPIxBUF and the transmit data in SPIxSR is ready for the next transfer.

In Master mode, the clock is generated by prescaling the system clock. Data is transmitted as soon as SPIBUF is written to. 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 SCK. Again, the interrupt is generated when the last bit is latched in. If SSx control is enabled, then transmission and reception are enabled only when SSx = low. The SDOx output will be disabled in SSx mode with SSx high.

The clock provided to the module is (FOSC/4). This clock is then prescaled by the primary (PPRE<1:0>) and the 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.

17.1.1 WORD AND BYTE COMMUNICATION

A control bit, MODE16 (SPIxCON<10>), allows the module to communicate in either 16-bit or 8-bit mode. Figure 17-5 through Figure 17-7 demonstrate the functionality of the module in 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 SPI module always gets reset to start a new communication 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 SPIxSR for 8-bit operation, and data is transmitted out of bit15 of the SPIxSR for 16-bit operation. In both modes, data is shifted into bit 0 of the SPIxSR.

17.1.2 SDOx DISABLE

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

17.2 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 SSx pin to perform the frame synchronization pulse (FSYNC) function. The control bit SPIFSD determines whether the SSx 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 starts transmitting only on the subsequent transmit edge of the SPI clock.

17.2.1 MODE: SPI MASTER, FRAME MASTER

With framed SPI enabled, the clock will be output continuously, regardless of whether the module is transmitting. The FRMEN bit is high, and the SPIFSD bit is low. When the SPIxBUF is written, the SSx pin will be driven high on the next transmit edge of the SPI clock. The SSx pin will be high for one SPI clock cycle. Data will start transmitting on the next transmit edge of the SPI clock, as shown in Figure 17-1.

17.2.2 MODE: SPI MASTER, FRAME SLAVE

The FRMEN bit is high, and the SPIFSD bit is high. The SSx pin is an input, and it is sampled on the sample edge of the SPI clock. When it is sampled high, data will be transmitted on the subsequent transmit edge of the SPI clock, as shown in Figure 17-2. An interrupt will be generated when the transmission is complete. The user must make sure that the correct data is loaded into the SPIxBUF for transmission before the SSx signal is received.

17.2.3 MODE: SPI SLAVE, FRAME MASTER

With framed SPI enabled, FRMEN = high, the input SPI clock will be continuous in Slave mode. the SSx pin will be an output when the SPIFSD bit is low. Therefore, when the SPIxBUF is written, the module will drive the SSx pin high on the next transmit edge of the SPI clock. The SSx pin will be driven high for one SPI clock cycle. Data will start transmitting on the next SPI clock transmit edge.

17.2.4 MODE: SPI SLAVE, FRAME SLAVE

The FRMEN bit is high, and the SPIFSD bit is high. Therefore, both the SCK and SSx pins will be inputs. The SSx pin will be sampled on the sample edge of the SPI clock. When SSx is sampled high, data will be transmitted on the next transmit edge of SCK.

FIGURE 17-1: SPI MASTER, FRAME MASTER

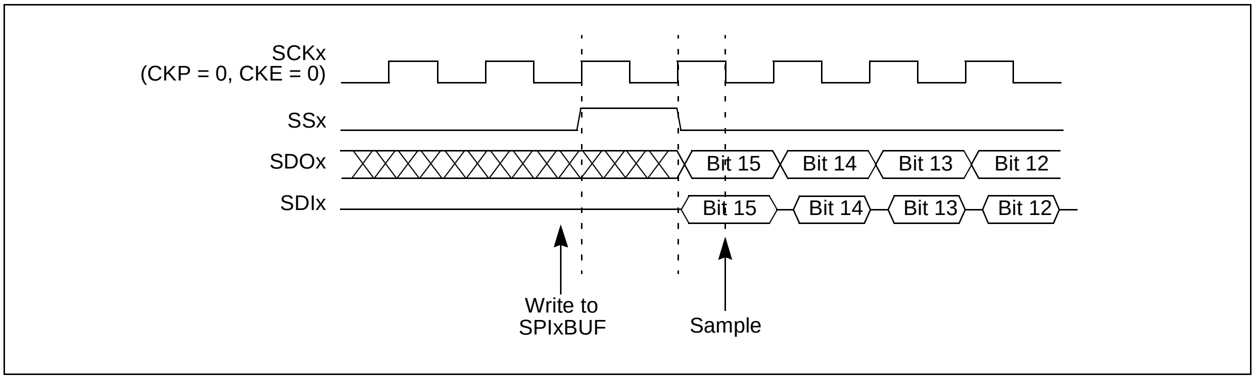


FIGURE 17-2: SPI MASTER, FRAME SLAVE

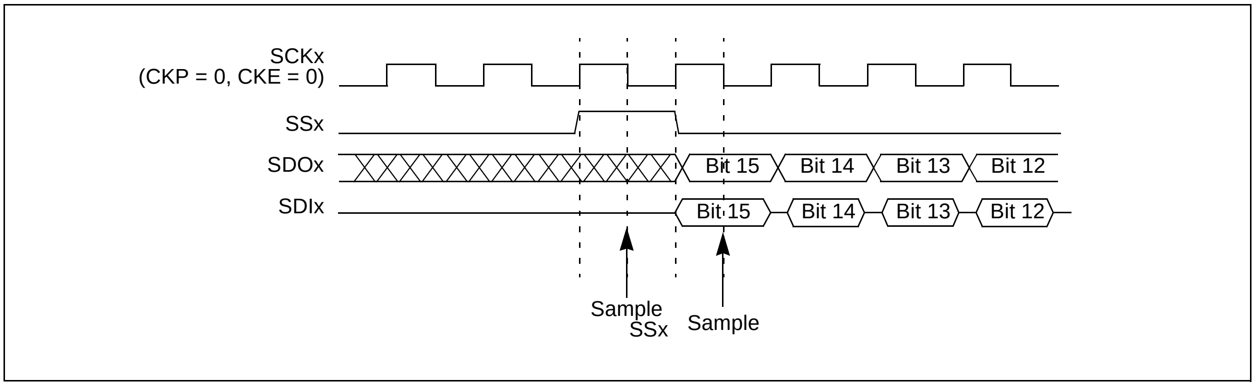
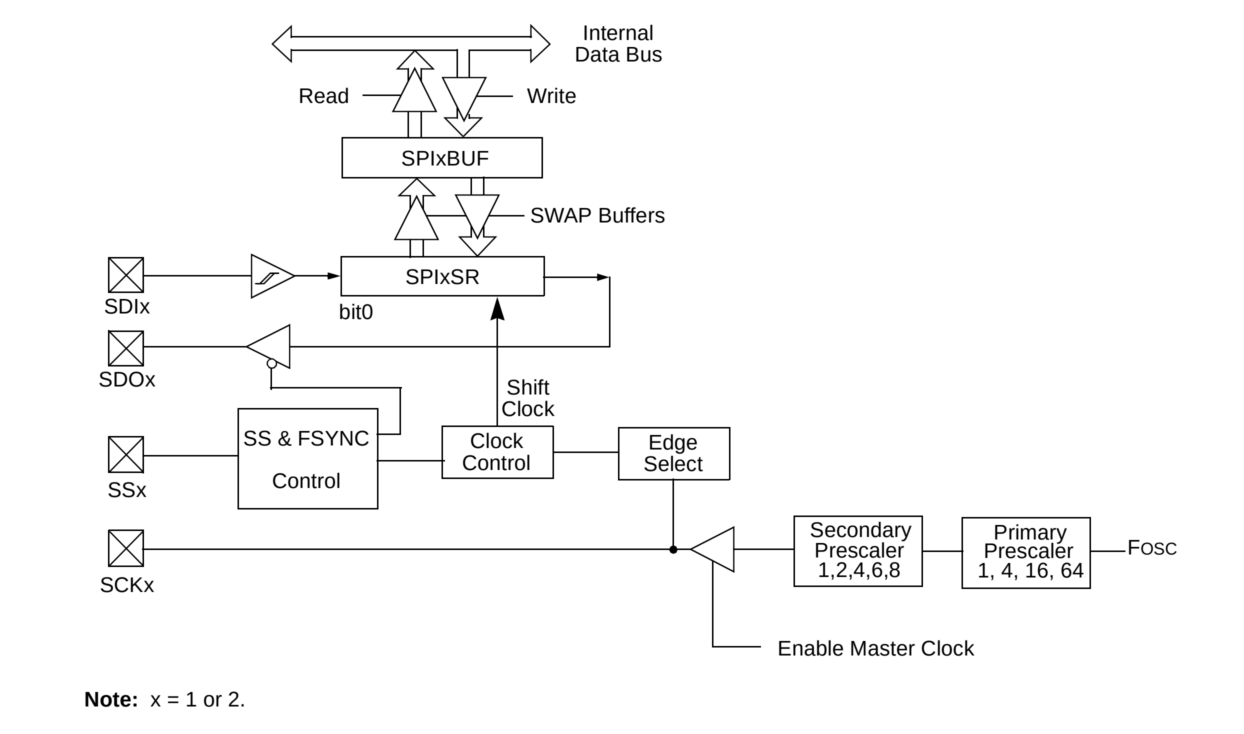
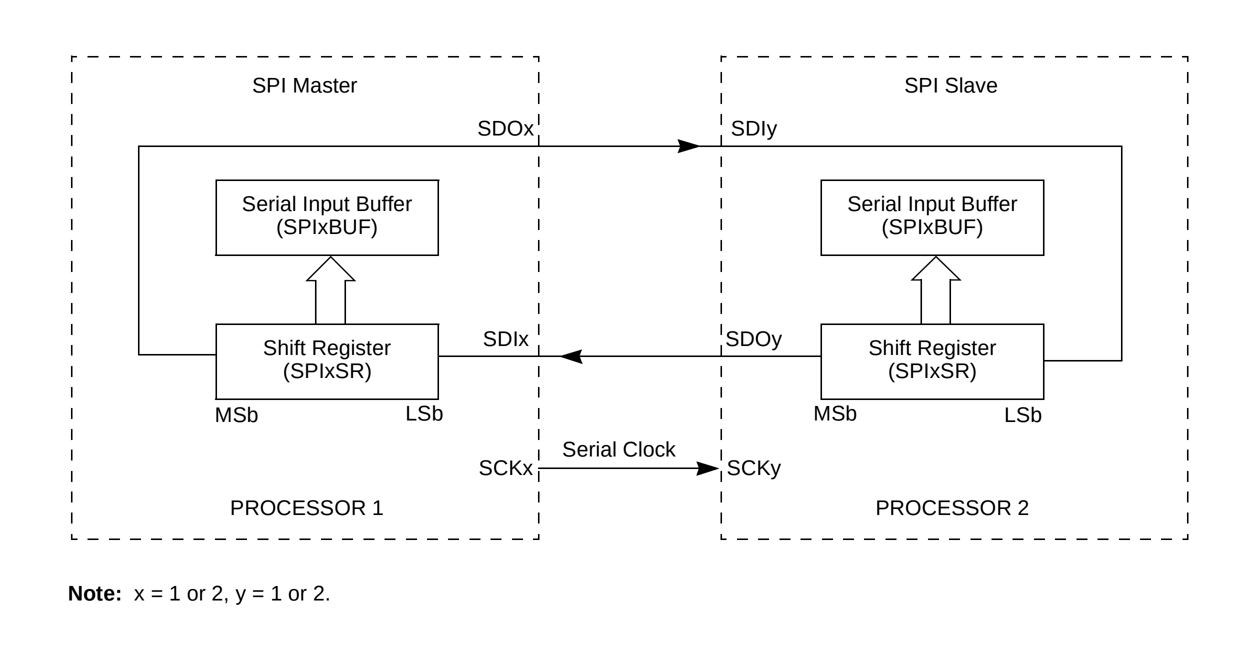


FIGURE 17-3: SPI BLOCK DIAGRAM



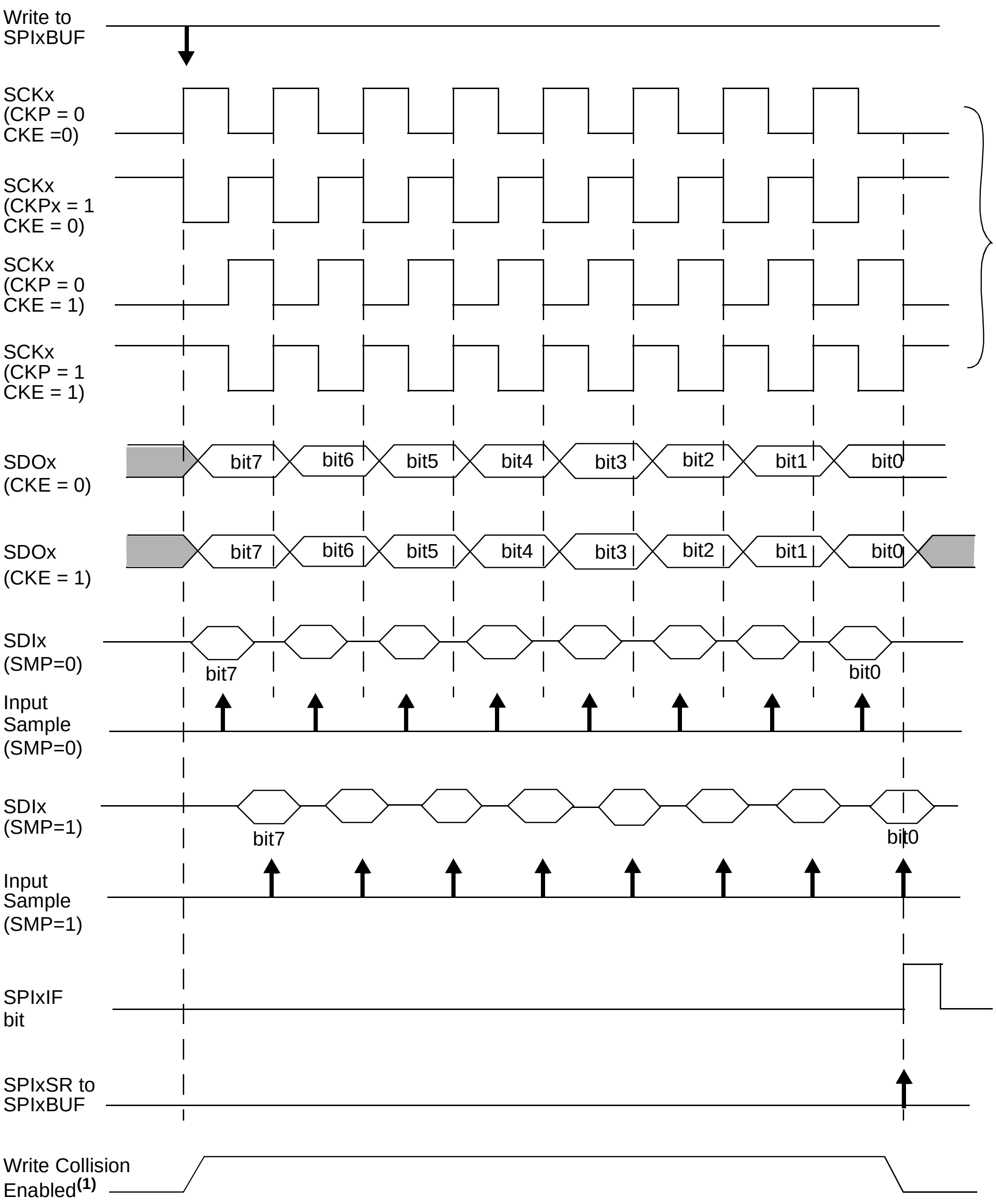
Note: x = 1 or 2.

FIGURE 17-4: SPI MASTER/SLAVE CONNECTION



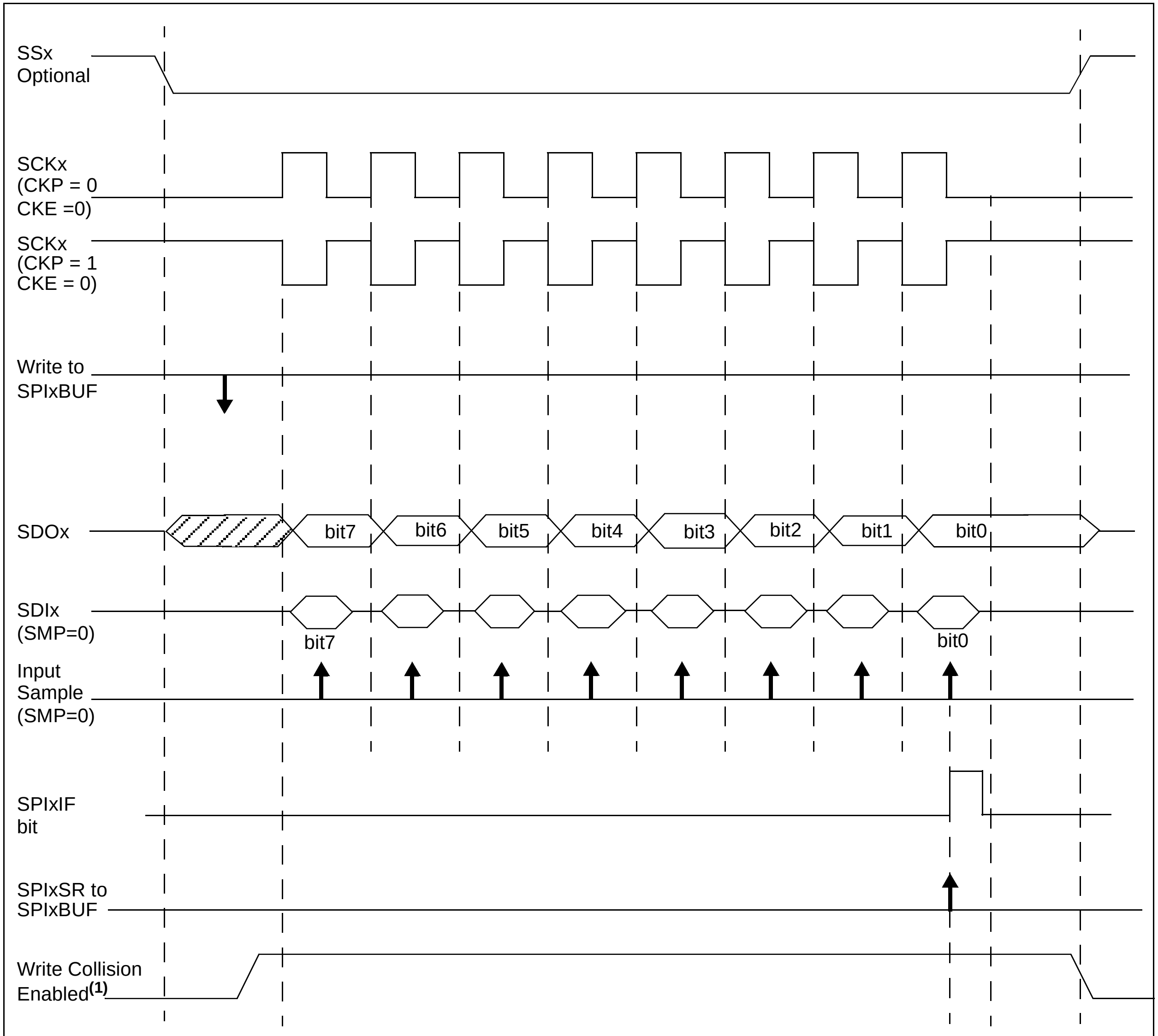
Note: x = 1 or 2, y = 1 or 2.

FIGURE 17-5: SPI MODE TIMING (8-BIT MASTER MODE)



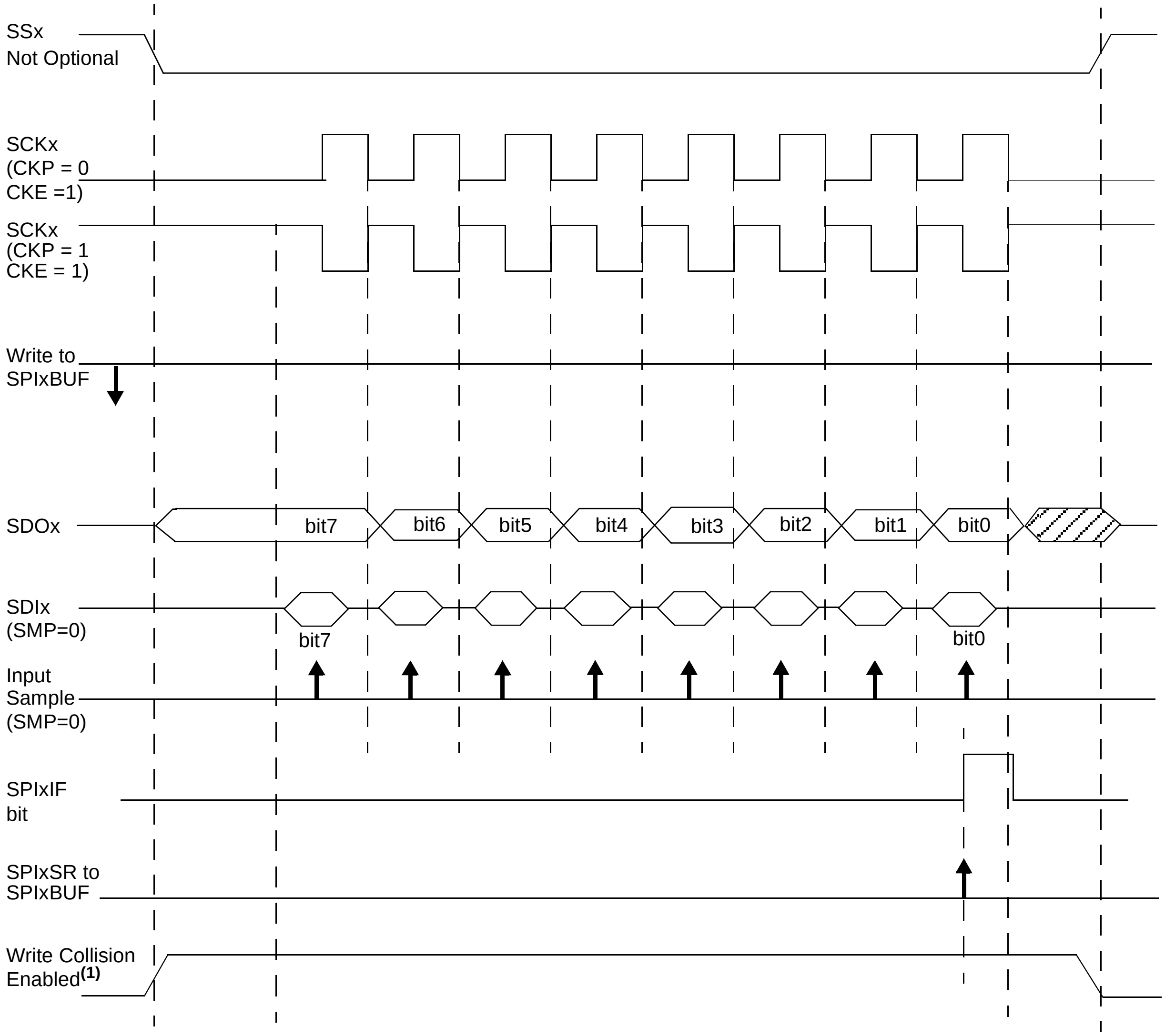
Note 1: When Write Collision is enabled (= 1), an attempt by the CPU to write to SPIxBUF will set the SWCOL status bit, and SPIxBUF will not be written.

FIGURE 17-6: SPI MODE TIMING (8-BIT SLAVE MODE WITH CKE = 0)



Note 1: When Write Collision is enabled (= 1), an attempt by the CPU to write to SPIxBUF will set the SWCOL status bit, and SPIxBUF will not be written.

FIGURE 17-7: SPI MODE TIMING (8-BIT SLAVE MODE WITH CKE = 1)

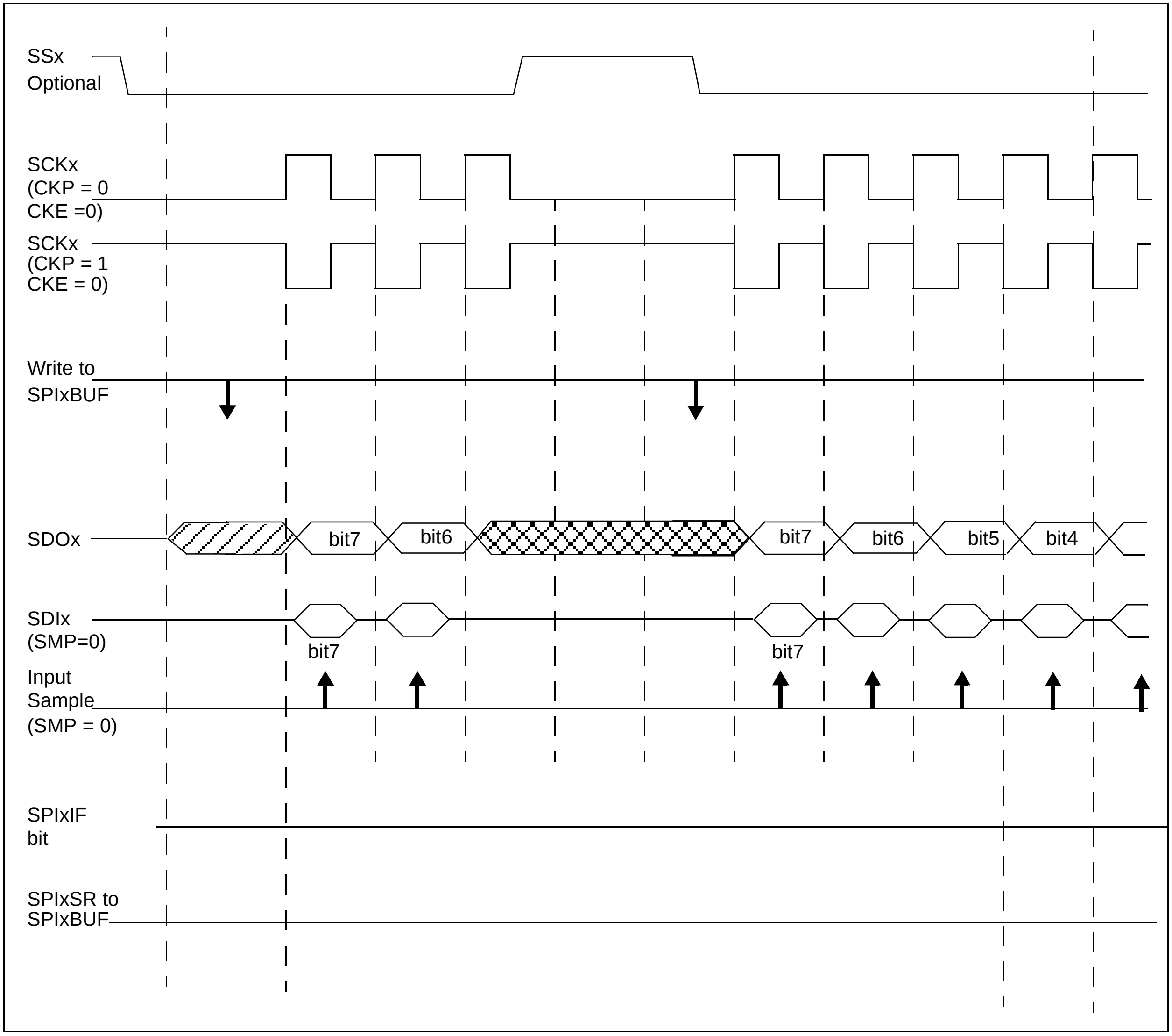


Note 1: When Write Collision is enabled (= 1), an attempt by the CPU to write to SPIxBUF will set the SWCOL status bit, and SPIxBUF will not be written.

17.3 Slave Select Synchronization

The SSx pin allows a Synchronous Slave mode. The SPI must be configured in SPI Slave mode, with SSx pin control enabled (SSEN = 1). When the SSx pin is low, transmission and reception are enabled, and the SDOx pin is driven. When SSx pin goes high, the SDOx pin is no longer driven. Also, the SPI module is re-synchronized, all counters and control circuitry are reset; therefore, when the SSx pin is asserted low again, transmission/reception will begin at the Most Significant bit, even if SSx had been de-asserted in the middle of a transmit/receive (see Figure 17-8).

FIGURE 17-8: SLAVE SYNCHRONIZATION TIMING (MODE 16 = 0)



17.4 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.

17.5 SPI Operation During CPU IDLE Mode

When the device enters IDLE mode, all clock sources remain functional. The SPISIDL bit selects if the SPI module will stop on IDLE or continue on IDLE. If SPISIDL = 0, the module will continue operation when the CPU enters IDLE mode. If SPISIDL = 1, the module will stop when the CPU enters IDLE mode.

TABLE 17-1: SPI1 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	—	SPISIDL	—	—	—	—	—	—	SPIROV	—	—	—	—	—	SBF	0u0u uuuu u0uu uuu0
SPI1CON	0222	—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	u00u 0000 0000 0000
SPI1BUF	0224	Transmit and Receive Buffer																0000 0000 0000 0000

TABLE 17-2: SPI2 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
SPI2STAT	0226	SPIEN	—	SPISIDL	—	—	—	—	—	—	SPIROV	—	—	—	—	—	SBF	0u0u uuuu u0uu uuu0
SPI2CON	0228	—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	u00u 0000 0000 0000
SPI2BUF	022A	Transmit and Receive Buffer																0000 0000 0000 0000

REGISTER 17-1: SPIxSTAT: SPI STATUS REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
SPIEN	—	SPISIDL	—	—	—	—	—
bit 15							bit 8

Lower Half:

U-0	R/W-0 HS	U-0	U-0	U-0	U-0	U-0	R-0 HS,HC
—	SPIROV	—	—	—	—	—	SBF
bit 7							bit 0

bit 15 **SPIEN:** SPI Enable bit
1 = Enables module and configures SCK, SDOx, SDIx and SS as serial port pins
0 = Disables module. All pins controlled by PORT functions. Power consumption is minimal.

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **SPISIDL:** Stop in IDLE Mode bit
1 = Discontinue module operation when device enters a IDLE mode
0 = Continue module operation in IDLE mode

bit 12-7 **Unimplemented:** Read as ‘0’

bit 6 **SPIROV:** Receive Overflow Flag bit
1 = A new byte/word is completely received, and the CPU has not read the previous data in the SPIBUF register
0 = No overflow

bit 5-1 **Unimplemented:** Read as ‘0’

bit 0 **SBF:** SPI Buffer Full Status bit (cleared in hardware when SPIBUF is read or written)
1 = Receive complete, SPIBUF is full
0 = Receive is not complete, SPIBUF is empty

Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
HC = Cleared by Hardware	HS = Set by Hardware	-n = Value at POR
1 = bit is set	0 = bit is cleared	x = bit is unknown

REGISTER 17-2: SPIxCON: SPI CONTROL REGISTER

Upper Half:							
U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0
bit 7							bit 0

bit 15 **Unimplemented:** Read as ‘0’

bit 14 **FRMEN:** Framed SPI Support bit
1 = Framed SPI support enabled
0 = Framed SPI support disabled

bit 13 **SPIFSD:** Frame Sync Pulse Direction Control bit
1 = Frame sync pulse input (slave)
0 = Frame sync pulse output (master)

bit 12 **Unimplemented:** Read as ‘0’

bit 11 **DISSDO:** Disable SDOx Pin bit
1 = SDOx pin is not used by module. Pin controlled by PORT function.
0 = SDOx pin is controlled by the module

bit 10 **MODE16:** Word/Byte Communication Select bit
1 = Communication is word wide (16-bits)
0 = Communication is byte wide (8-bits)

bit 9 **SMP:** SPI Data Input Sample Phase bit
Master mode:
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time
Slave mode:
SMP must be cleared when SPI is used in Slave mode

bit 8 **CKE:** SPI Clock Edge Select bit
1 = Transmit happens on transition from active clock state to IDLE clock state
0 = Transmit happens on transition from IDLE clock state to active clock state

bit 7 **SSEN:** Slave Select Enable bit (Slave Mode)
1 = $\overline{\text{SS}}$ pin used for Slave mode
0 = $\overline{\text{SS}}$ pin not used by module. Pin controlled by PORT function.

bit 6 **CKP:** Clock Polarity Select bit
1 = IDLE state for clock is a high level; active state is a low level
0 = IDLE state for clock is a low level; active state is a high level

bit 5 **MSTEN:** Master Mode Enable bit
1 = Master mode
0 = Slave mode

bit 4-2 **SPRE<2:0>:** Secondary Prescale bits (Master Mode)
111 = Secondary prescale 1:1
110 = Secondary prescale 2:1
101 = Reserved, do not use
100 = Secondary prescale 4:1
011 = Reserved, do not use
010 = Secondary prescale 6:1
001 = Reserved, do not use
000 = Secondary prescale 8:1

REGISTER 17-2: SPIxCON: SPI CONTROL REGISTER (Continued)

bit 1-0 **PPRE<1:0>:** Primary Prescale bits (Master Mode)
`11` = Primary prescale 1:1
`10` = Primary prescale 4:1
`01` = Primary prescale 16:1
`00` = Primary prescale 64:1

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

NOTES:

18.0 I^{2}C MODULE

The Inter-Integrated Circuit^{TM} (I^{2}C^{TM}) module provides complete hardware support for both Slave and Multi-Master modes of the I^{2}C serial communication standard, with a 16-bit interface.

This module offers the following key features:

- Inter-Integrated Circuit (I^{2}C) interface
- I^{2}C interface supports both Master and Slave modes.
- I^{2}C Slave mode supports 7 and 10 bit address.
- I^{2}C Master mode supports 7 and 10 bit address.
- I^{2}C port allows bi-directional transfers between master and slaves.
- Serial clock synchronization for I^{2}C port can be used as a handshake mechanism to suspend and resume serial transfer (SCLREL control).
- I^{2}C supports Multi-Master mode; detects bus collision and will arbitrate accordingly.

18.1 Operating Function Description

The hardware fully implements all the master and slave functions of the I^{2}C Standard and Fast mode specifications, as well as 7 and 10-bit addressing.

Thus, the I^{2}C module can operate either as a slave or a master on an I^{2}C bus.

18.1.1 VARIOUS I^{2}C MODES

The following operating modes are supported:

- I^{2}C Slave mode (7-bit address)
- I^{2}C Slave mode (10-bit address)
- I^{2}C Master mode (7 or 10-bit address)

See the programmer’s model in Figure 18-3.

FIGURE 18-1: I^{2}C BLOCK DIAGRAM (I^{2}C RECEIVE)

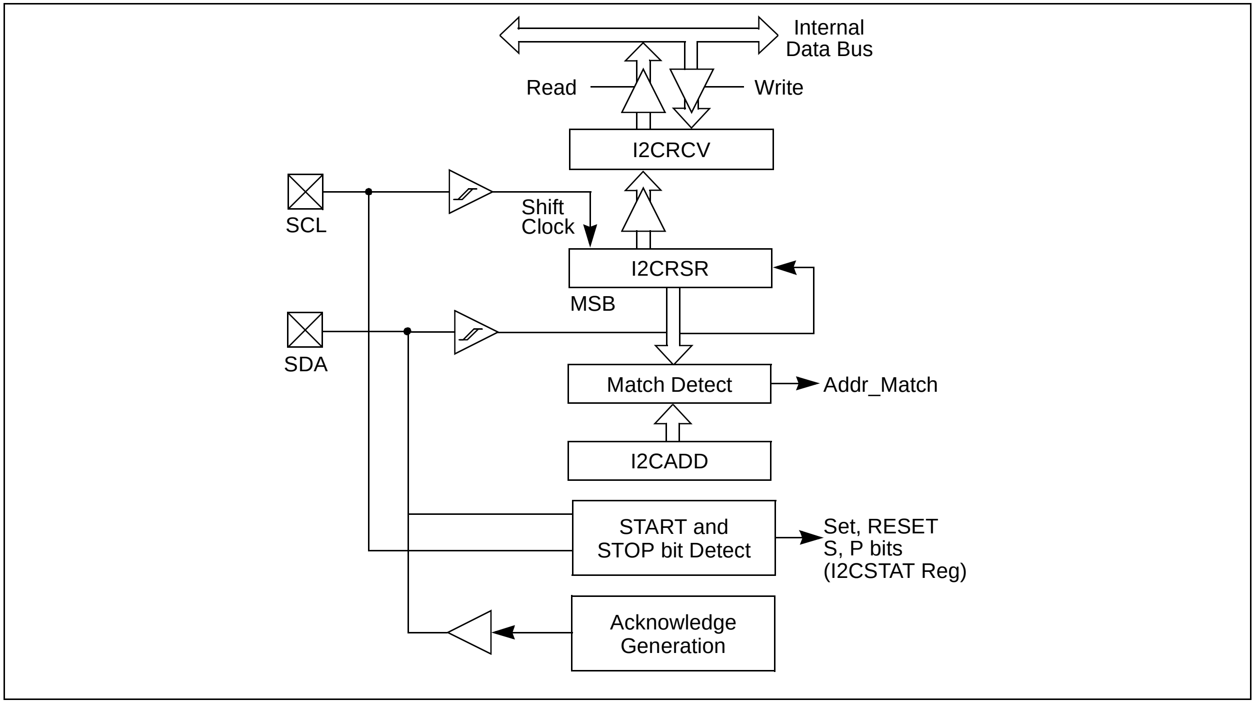


FIGURE 18-2: I^{2}C BLOCK DIAGRAM (TRANSMIT)

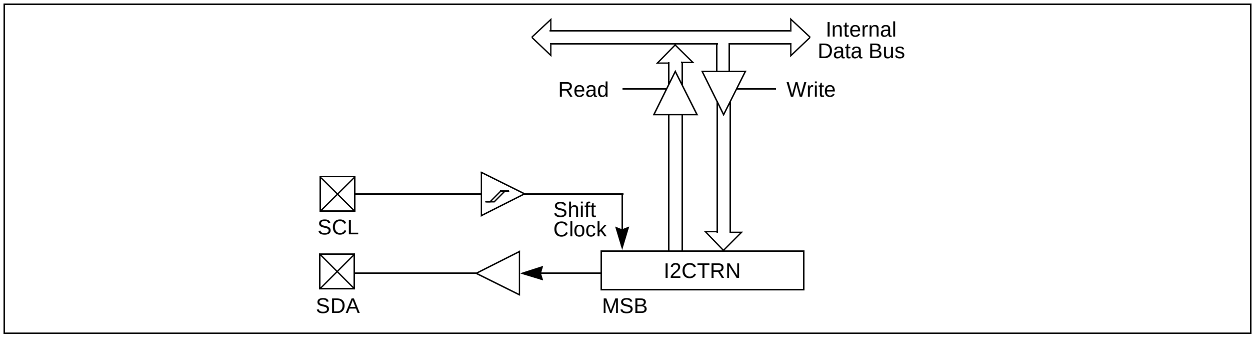
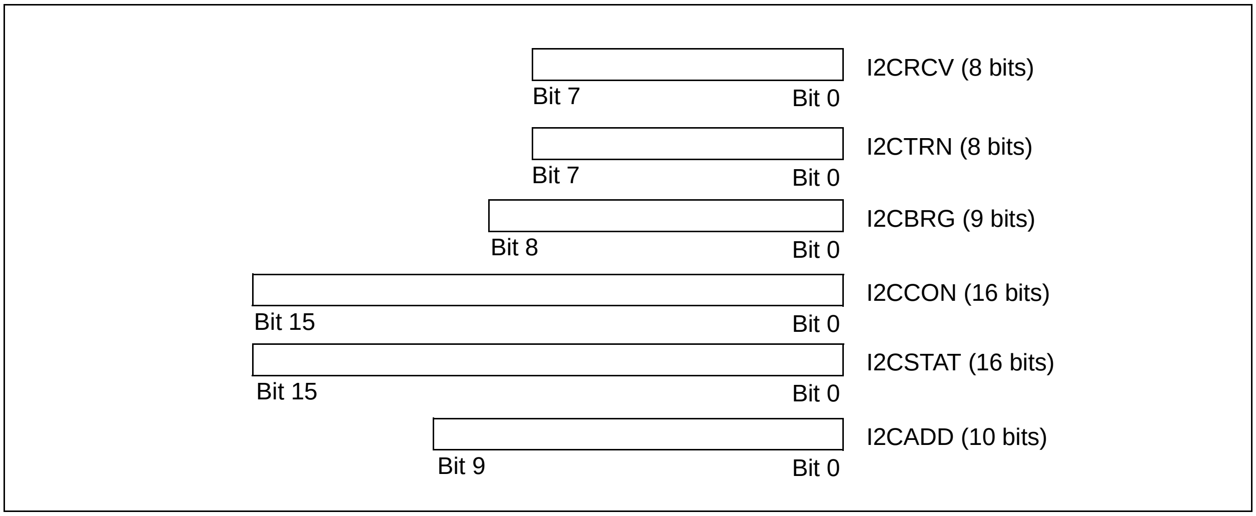


FIGURE 18-3: PROGRAMMER'S MODEL



18.1.2 PIN CONFIGURATION IN I^{2}C MODE

I^{2}C has a 2-pin interface: pin SCL is clock and pin SDA is data.

18.1.3 I^{2}C 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 to or read from. This register is the receive buffer, as shown in Figure 16-1. I2CTRN is the transmit register to which bytes are written during a transmit operation, as shown in Figure 16-2.

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 the I2CIF 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.

18.2 I^{2}C 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 assumed to be a 7-bit address. When an address is received, it is compared to the Least Significant 7 bits 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 '`1 1 1 1 0 A9 A8`' (where `A9`, `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.

18.3 I^{2}C 7-bit Slave Mode Operation

Once enabled (I2CEN = 1), the slave module will wait for a START bit to occur (i.e., the I^{2}C 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), bits I2CADD<6:0> 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 Acknowledge will be sent, and on the falling edge of the ninth bit ($\overline{\text{ACK}}$ bit) the I2CIF interrupt pulse is generated. The address match does not affect the contents of the I2CRCV buffer or the RBF bit.

18.3.1 SLAVE MODE TRANSMISSION

If the R_W bit received is a ’1’, then the serial port will go into Transmit mode. It will send $\overline{\text{ACK}}$ on the ninth bit and then hold SCL to ’0’ until the CPU responds by writing to I2CTRN. SCL is released 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 $\overline{\text{ACK}}$ received from the master.

The $\overline{\text{ACK}}$ bit from master is latched on the ninth clock pulse. If $\overline{\text{ACK}}$ = 1, then the Transmit mode ends and the serial port resumes looking for another START bit.

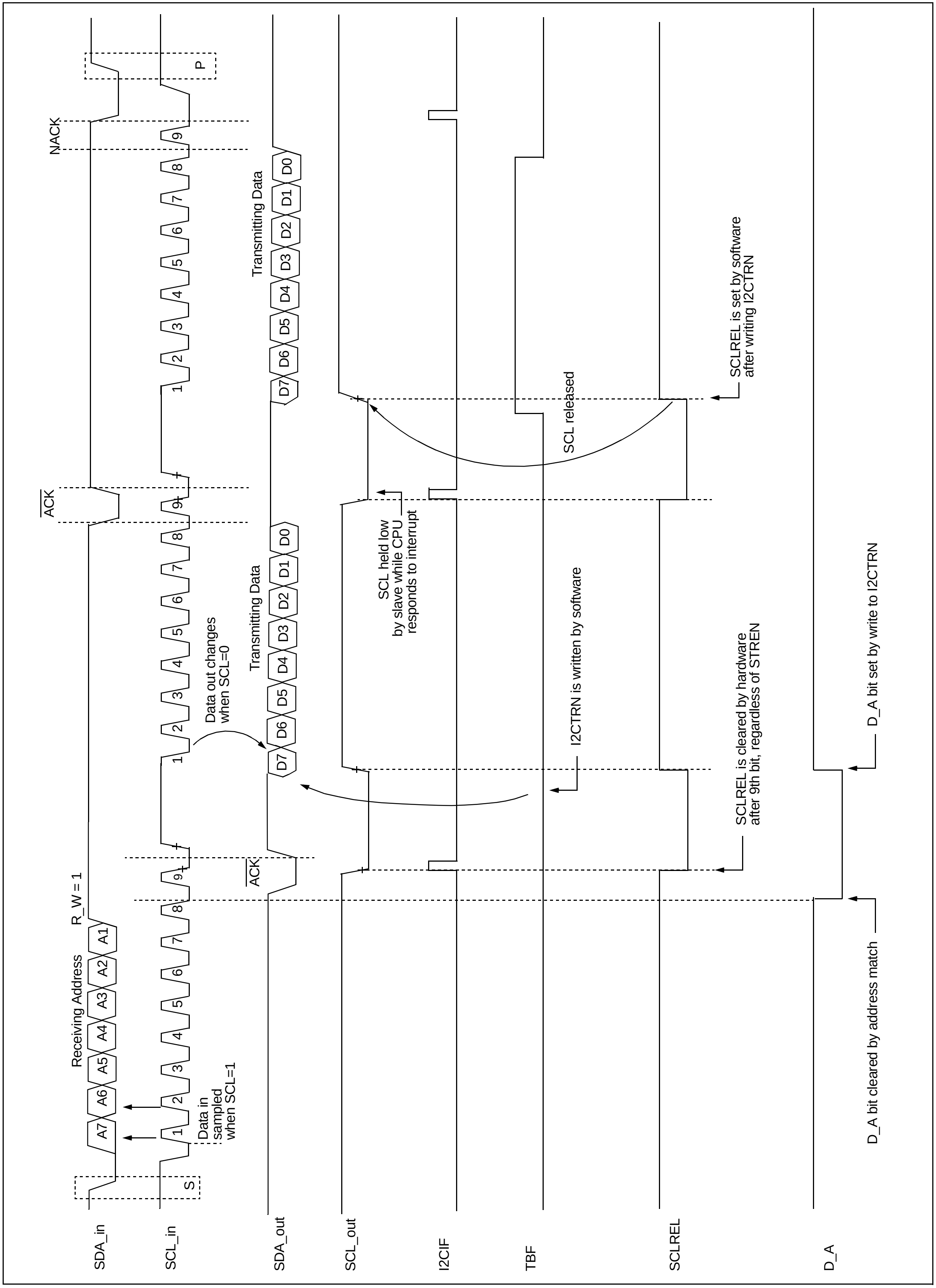
If $\overline{\text{ACK}}$ = 0, then it will again hold SCL low until SCLREL is set by user software. The user must write I2CTRN prior to setting SCLREL.

TBF Status Flag: During transmit, the TBF bit (I2CSTAT<0>) is set when the CPU writes to I2CTRN, and TBF is cleared in hardware when all 8 bits are shifted out.

IWCOL Status Flag: If the user attempts to write a byte to the I2CTRN register when TBF = 1 (i.e., I2CTRN is still shifting out previous data byte), then IWCOL is set. IWCOL must be cleared in software.

R_W Status Flag: Latches and holds the R_W bit received following the last address match, which indicates whether the data transfer was an input or an output.

FIGURE 18-4: I^{2}C SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)


SDA_in
SCL_in
SDA_out
SCL_out
I2CIF
TBF
SCLREL
D_A
S
Receiving Address
A7
A6
A5
A4
A3
A2
A1
R_W = 1
1
2
3
4
5
6
7
8
9
Data in sampled when SCL=1
ACK
Data out changes when SCL=0
Transmitting Data
D7
D6
D5
D4
D3
D2
D1
D0
ACK
NACK
P
SCL held low by slave while CPU responds to interrupt
I2CTRN is written by software
SCL released
SCLREL is cleared by hardware after 9th bit, regardless of STREN
SCLREL is set by software after writing I2CTRN
D_A bit cleared by address match
D_A bit set by write to I2CTRN

18.3.2 SLAVE MODE 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. $\overline{\text{ACK}}$ 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 $\overline{\text{ACK}}$ 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 Acknowledge is not sent ($\overline{\text{ACK}}$ = 1) and the I2CRCV is updated.

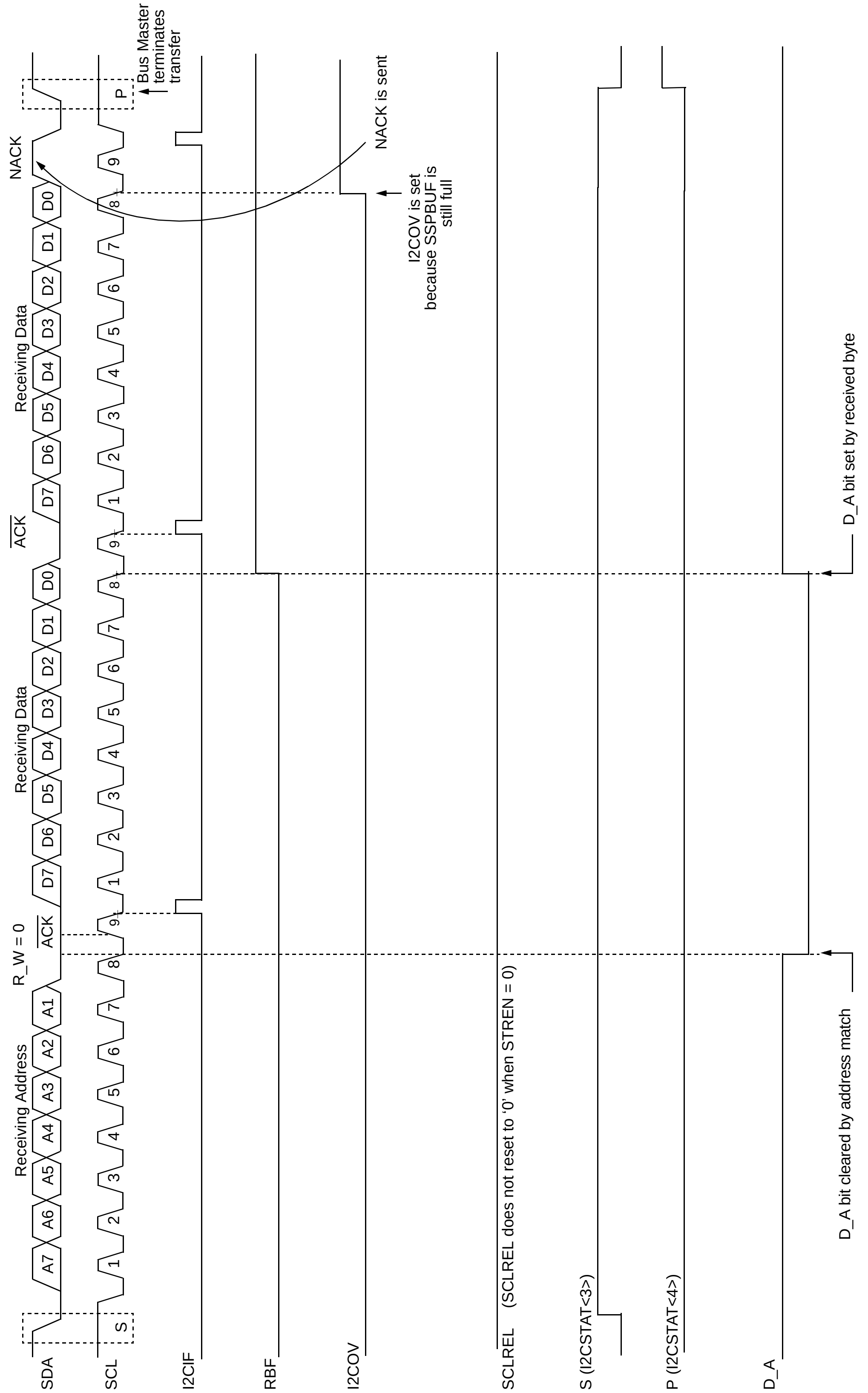
RBF Status Flag: RBF is set when an address or data byte is loaded into I2CRCV from I2CRSR. It is cleared when I2CRCV is read.

I2COV Status Flag: I2COV is set when 8 bits are received into the I2CRSR, and the RBF flag has not been cleared from a previous reception.

R_W Status Flag: Latches and holds the R_W bit received following the last address match, which indicates whether the data transfer was an input or an output.

D_A Status Flag: Indicates whether the last byte received was data or an address.

FIGURE 18-5: I^{2}C SLAVE MODE TIMING WITH STREN = 0 (RECEPTION, 7-BIT ADDRESS)



18.4 I^{2}C 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 I^{2}C 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 only 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.

Then, the low byte of the address is received and compared against 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.

18.4.1 10-BIT 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.

18.4.2 10-BIT SLAVE RECEPTION

Once addressed, the master can, without generating a STOP bit, generate a Repeated START bit and reset the high byte of the address and R_W = 1, thus initiating a slave transmit operation.

FIGURE 18-6: I^{2}C SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)

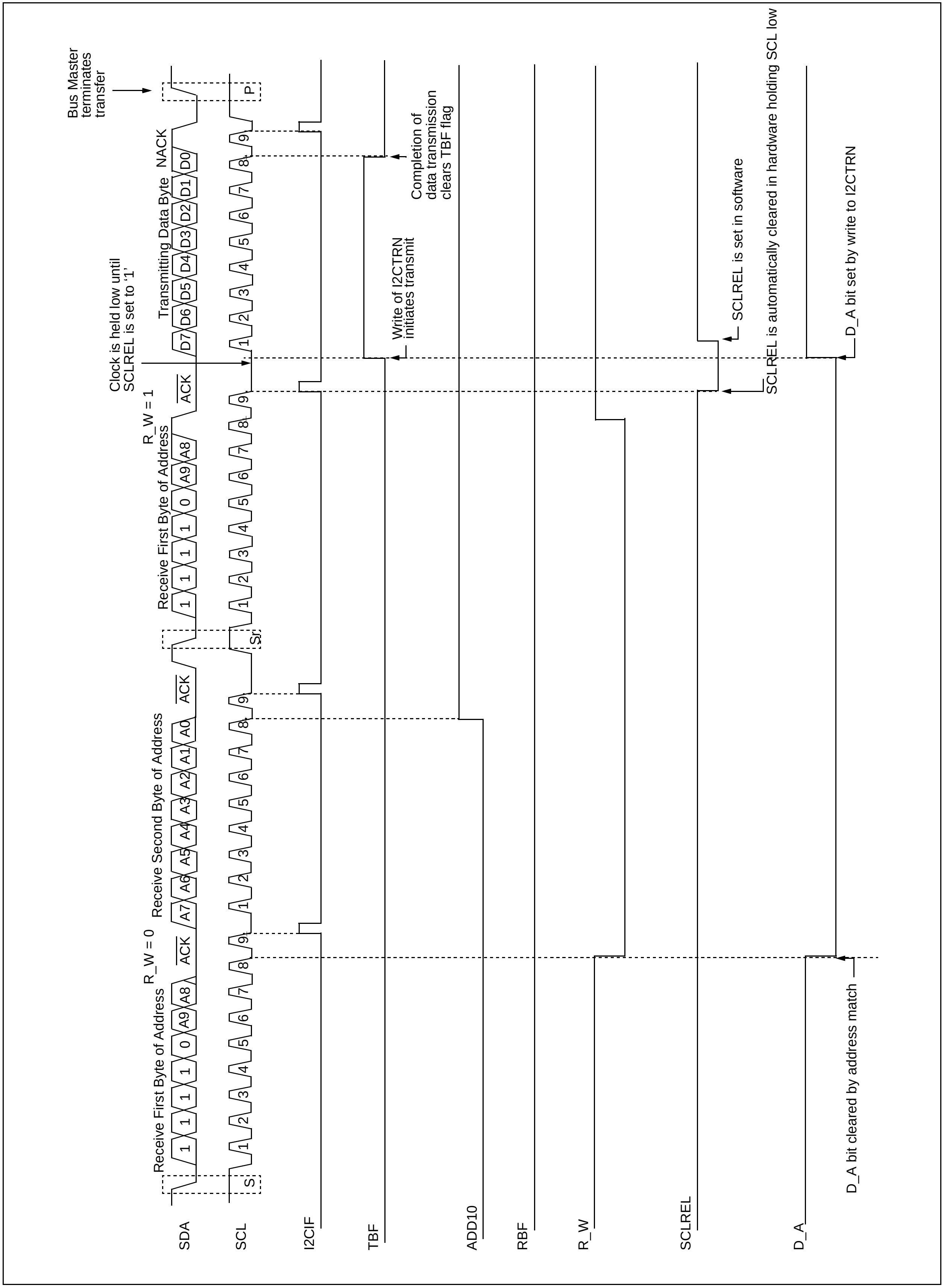
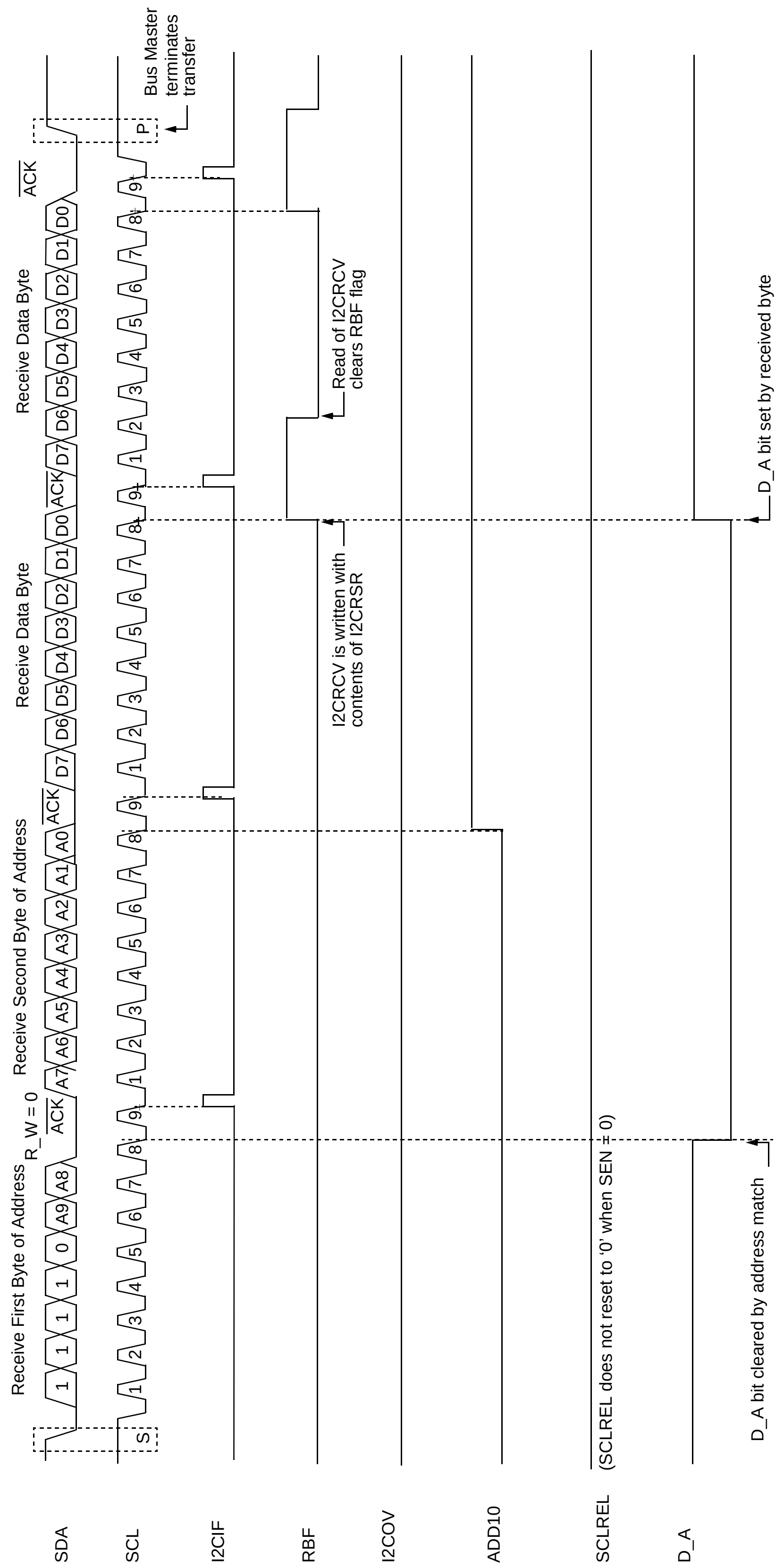

SDA
SCL
I2CIF
TBF
ADD10
RBF
R_W
SCLREL
D_A
Receive First Byte of Address
R_W = 0
Receive Second Byte of Address
R_W = 1
Receive First Byte of Address
Clock is held low until SCLREL is set to '1'
Transmitting Data Byte
NACK
Bus Master terminates transfer
S
1 1 1 1 0 A9 A8 ACK
A7 A6 A5 A4 A3 A2 A1 A0 ACK
Sr
1 1 1 1 0 A9 A8 ACK
D7 D6 D5 D4 D3 D2 D1 D0
P
1 2 3 4 5 6 7 8 9
Write of I2CTRN initiates transmit
Completion of data transmission clears TBF flag
SCLREL is set in software
SCLREL is automatically cleared in hardware holding SCL low
D_A bit set by write to I2CTRN
D_A bit cleared by address match

FIGURE 18-7: I^{2}C SLAVE MODE TIMING WITH STREN = 0 (RECEPTION, 10-BIT ADDRESS)


SDA
SCL
I2CIF
RBF
I2COV
ADD10
SCLREL
D_A
Receive First Byte of Address
R_W = 0
Receive Second Byte of Address
Receive Data Byte
Receive Data Byte
ACK
ACK
ACK
ACK
S
P
1 1 1 1 0 A9 A8
A7 A6 A5 A4 A3 A2 A1 A0
D7 D6 D5 D4 D3 D2 D1 D0
D7 D6 D5 D4 D3 D2 D1 D0
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
Bus Master terminates transfer
I2CRCV is written with contents of I2CRSR
Read of I2CRCV clears RBF flag
D_A bit set by received byte
D_A bit cleared by address match
(SCLREL does not reset to '0' when SEN = 0)

18.5 Automatic Clock Stretch

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

18.5.1 TRANSMIT MODE 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. This occurs regardless of the state of the STREN bit.

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.

18.5.2 RECEIVE MODE 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.

18.5.3 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.

18.5.4 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 is described earlier.

FIGURE 18-8: I^{2}C SLAVE MODE TIMING WITH STREN = 1 (RECEPTION, 7-BIT ADDRESS)

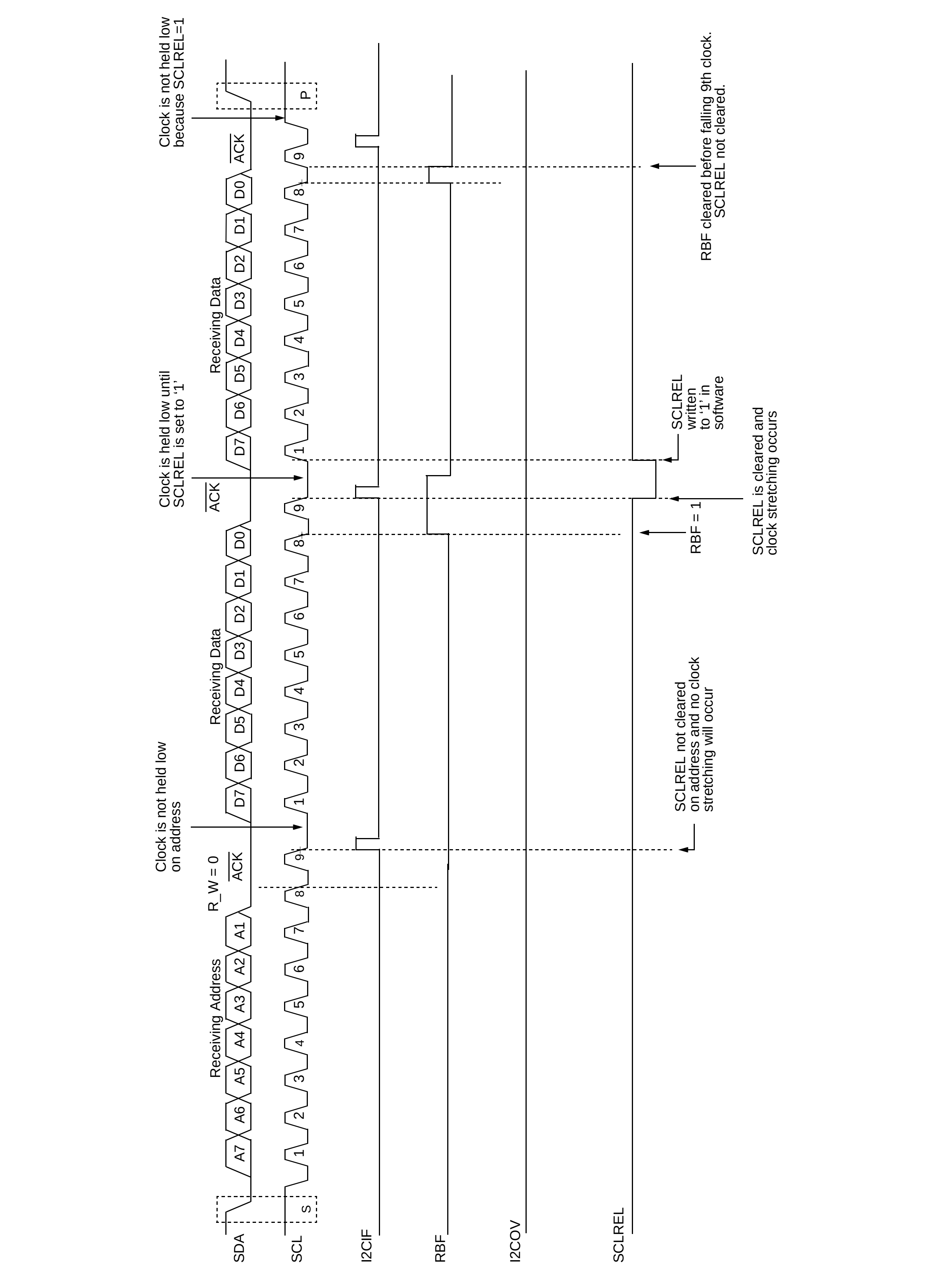
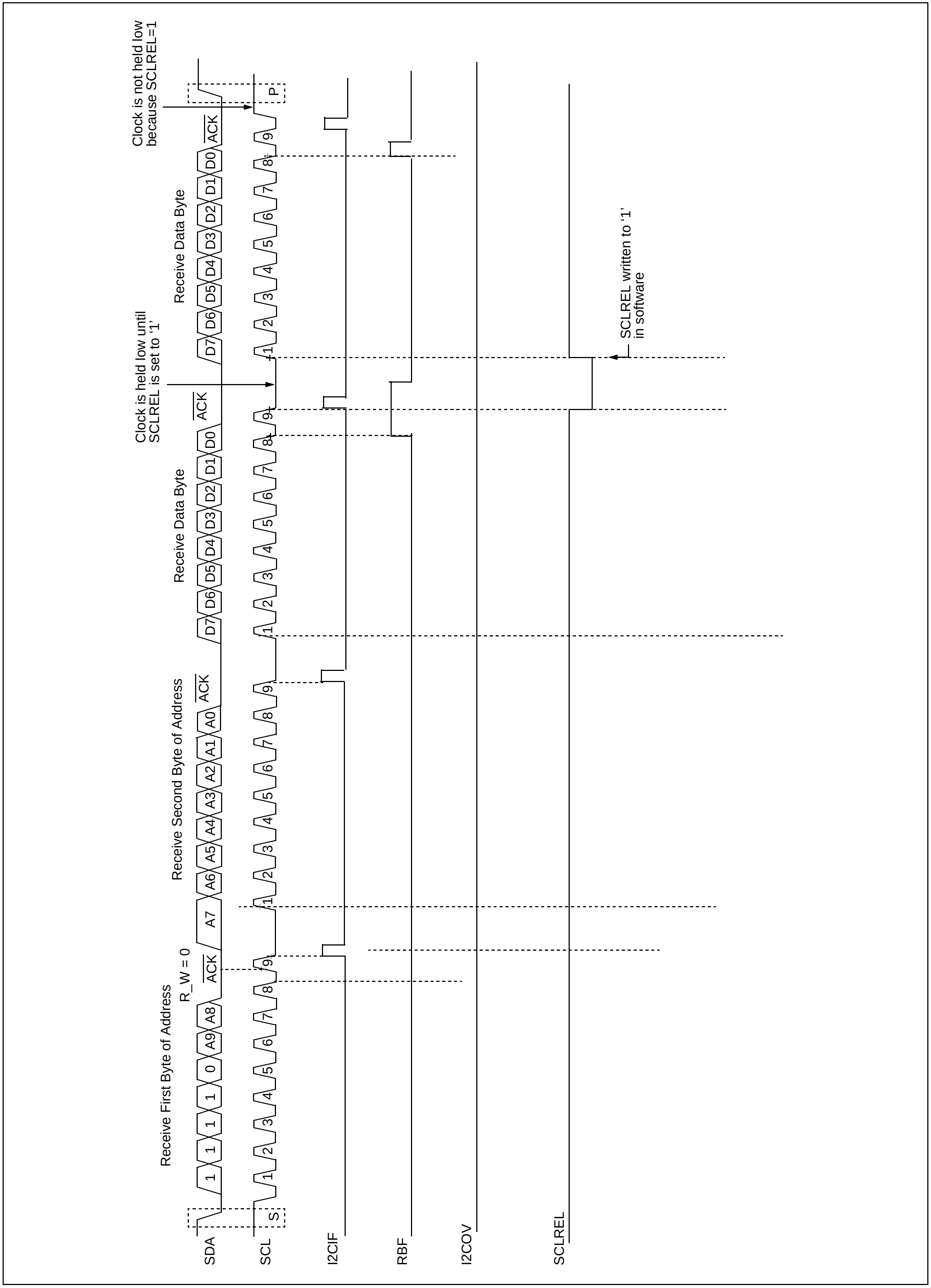

SDA
SCL
I2CIF
RBF
I2COV
SCLREL
S
P
Receiving Address
A7
A6
A5
A4
A3
A2
A1
R_W = 0
ACK
Receiving Data
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Receiving Data
D7
D6
D5
D4
D3
D2
D1
D0
ACK
1
2
3
4
5
6
7
8
9
Clock is not held low on address
Clock is held low until SCLREL is set to '1'
Clock is not held low because SCLREL=1
SCLREL not cleared on address and no clock stretching will occur
RBF = 1
SCLREL is cleared and clock stretching occurs
SCLREL written to '1' in software
RBF cleared before falling 9th clock. SCLREL not cleared.

FIGURE 18-9: I²C SLAVE MODE TIMING STREN = 1 (RECEPTION, 10-BIT ADDRESS)


SDA
SCL
I2CIF
RBF
I2COV
SCLREL
S
P
Receive First Byte of Address
R_W = 0
1
1
1
1
0
A9
A8
ACK
Receive Second Byte of Address
A7
A6
A5
A4
A3
A2
A1
A0
ACK
Receive Data Byte
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Receive Data Byte
D7
D6
D5
D4
D3
D2
D1
D0
ACK
1
2
3
4
5
6
7
8
9
Clock is held low until SCLREL is set to '1'
Clock is not held low because SCLREL=1
SCLREL written to '1' in software

18.6 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 is already sampled low, the SCL output will be asserted. The SCL output will remain low until the SCLREL bit is set, and all other devices on the I^{2}C bus have de-asserted 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.

18.7 Interrupts

The I^{2}C module generates two interrupt flags, I2CIF (I^{2}C Transfer Complete Interrupt Flag) and BCLIF (I^{2}C Bus Collision Interrupt Flag). The I2CIF interrupt flag is pulsed high for one TCY on the falling edge of the 9th clock pulse. The BCLIF interrupt flag is pulsed high for one TCY when a bus collision event is detected.

18.8 Slope Control

The I^{2}C 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.

18.9 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.

18.9.1 SLEEP OPERATION

The control bit, SLPEN, dictates the operation of the I^{2}C module when a SLEEP event occurs. With SLPEN = 0 (RESET state), the module continues operation in SLEEP mode, assuming support signals are provided. Interrupts will be generated at the appropriate time. When SLPEN = 1, the module discontinues all functions when a SLEEP mode is entered. Interrupts are not generated while SLEEP is active.

18.10 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 Acknowledge.

The general call address is one of eight addresses reserved for specific purposes by the I^{2}C protocol. It consists of all 0’s with R_W = 0.

The general call address is recognized when the General Call Enable bit (GCEN) is set (I2CCON<15> = 1). Following a START bit detect, 8 bits are shifted into I2CRSR and the address is compared against I2CADD, and is also compared to 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 ($\overline{\text{ACK}}$ bit), the I2CIF interrupt 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.

18.11 I^{2}C Master Mode Support

In Master mode, the user has six options.

- 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 I^{2}C port to receive data.
- Generate an Acknowledge condition at the end of a received byte of data.

18.12 I^{2}C Master Mode Operation

The master device generates all the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a Repeated START condition. Since the Repeated START condition is also the beginning of the next serial transfer, the I^{2}C bus will not be released.

In Master Transmitter mode, serial data is output through 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 Acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the 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 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 Acknowledge bit is transmitted. START and STOP conditions indicate the beginning and end of transmission.

The baud rate generator used for I^2C mode operation is now used to set the SCL clock frequency for either 100 kHz or 400 kHz I^2C operation. 1 MHz operation is also supported. The baud rate generator re-load value is contained in the I2CBRG register. The baud rate generator will automatically begin counting on a write to the I2CTRN. Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK) the internal clock will automatically stop counting and the SCL pin will remain in its last state.

18.12.1 I^2C MASTER MODE 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 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. SCL is held low for 1 baud rate generator rollover count (TBRG). Data should be valid before SCL is released high. When the SCL pin is released high, it is held high for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the TBF flag is cleared and the master releases SDA, allowing the slave device being addressed to respond with an $\overline{\text{ACK}}$ bit during the ninth bit time, if an address match occurs or if data was received properly. If the master receives an Acknowledge, the Acknowledge status bit (ACKSTAT) is cleared. If not, the bit is set. After the ninth clock the I2CIF is set, and the master clock (baud rate generator) is suspended until the next event, thus leaving SCL low and SDA unchanged (see Figure 18-10).

The Transmit Status Flag, TRSTAT (I2CSTAT<14>), indicates that a master transmit is in progress.

TBF Status Flag:

In Transmit mode, the TBF bit (I2CSTAT<0>) is set when the CPU writes to I2CTRN and is cleared when all 8 bits are shifted out.

IWCOL Status Flag:

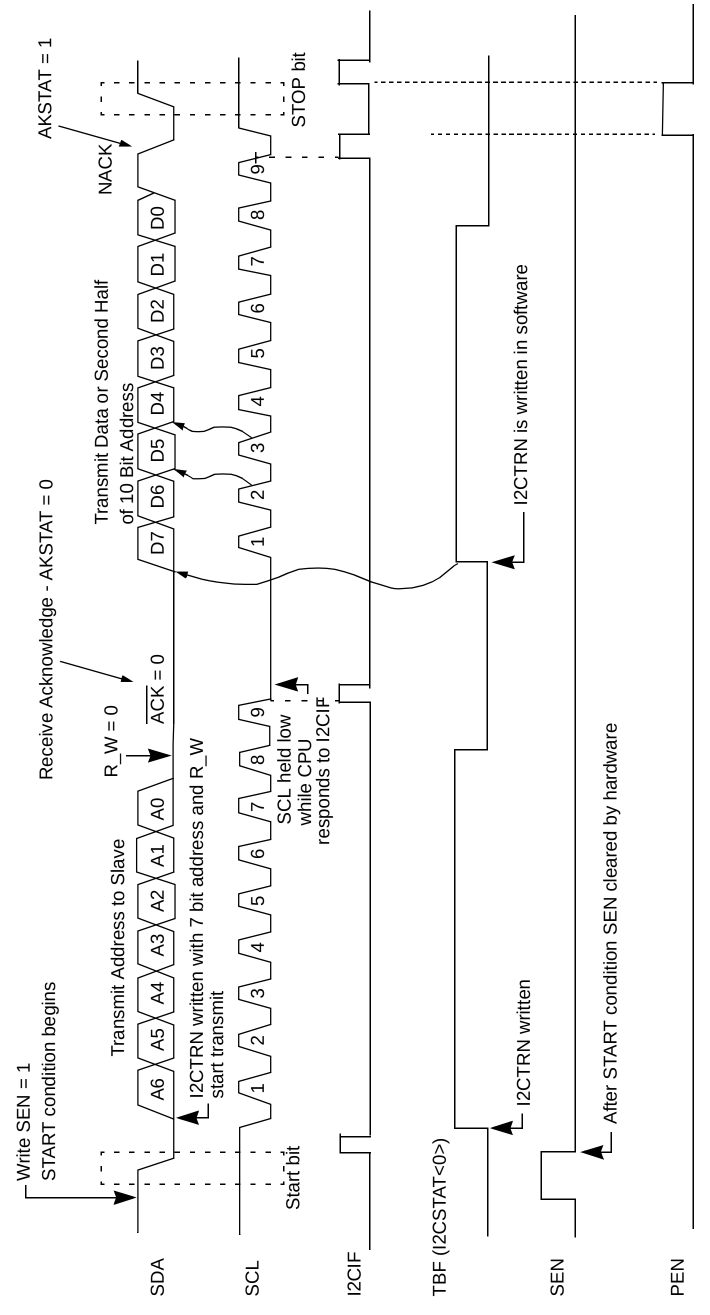
If the user writes the I2CTRN when a transmit is already in progress, then IWCOL is set and the contents of the buffer are unchanged (the write doesn’t occur). It must be cleared in software.

ACKSTAT Status Flag:

In Transmit mode, the ACKSTAT bit (I2CSTAT<15>) is cleared when the slave has sent an Acknowledge ($\overline{\text{ACK}}$ = 0), and is set when the slave does Not Acknowledge (ACK = 1). A slave sends an Acknowledge when it has recognized its address, or when the slave has properly received its data.

The D_A and R_W status bits are not applicable in Master mode.

FIGURE 18-10: I^{2}C MASTER MODE TIMING (TRANSMISSION, 7- OR 10-BIT ADDRESS)



18.12.2 I^{2}C MASTER MODE RECEPTION

Master mode reception is enabled by programming the receive enable (RCEN) bit (I2CCON<11>). The I^{2}C 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 changes (high to low/low to high), and data is shifted in to the I2CRSR on the rising edge of each clock. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the I2CRSR are loaded into the I2CRCV, the RBF and I2CIF bits are set, and the baud rate generator is suspended from counting, holding SCL low. The I^{2}C is now in IDLE state, awaiting the next command. When the I2CRCV is read by the CPU, the RBF flag is automatically cleared. The user can then send an Acknowledge bit at the end of reception, by setting the Acknowledge sequence enable (ACKEN) bit (I2CCON<12>).

RBF Status Flag:

In receive operation, RBF is set when an address or data byte is loaded into I2CRCV from I2CRSR. It is cleared when I2CRCV is read.

I2COV Status Flag:

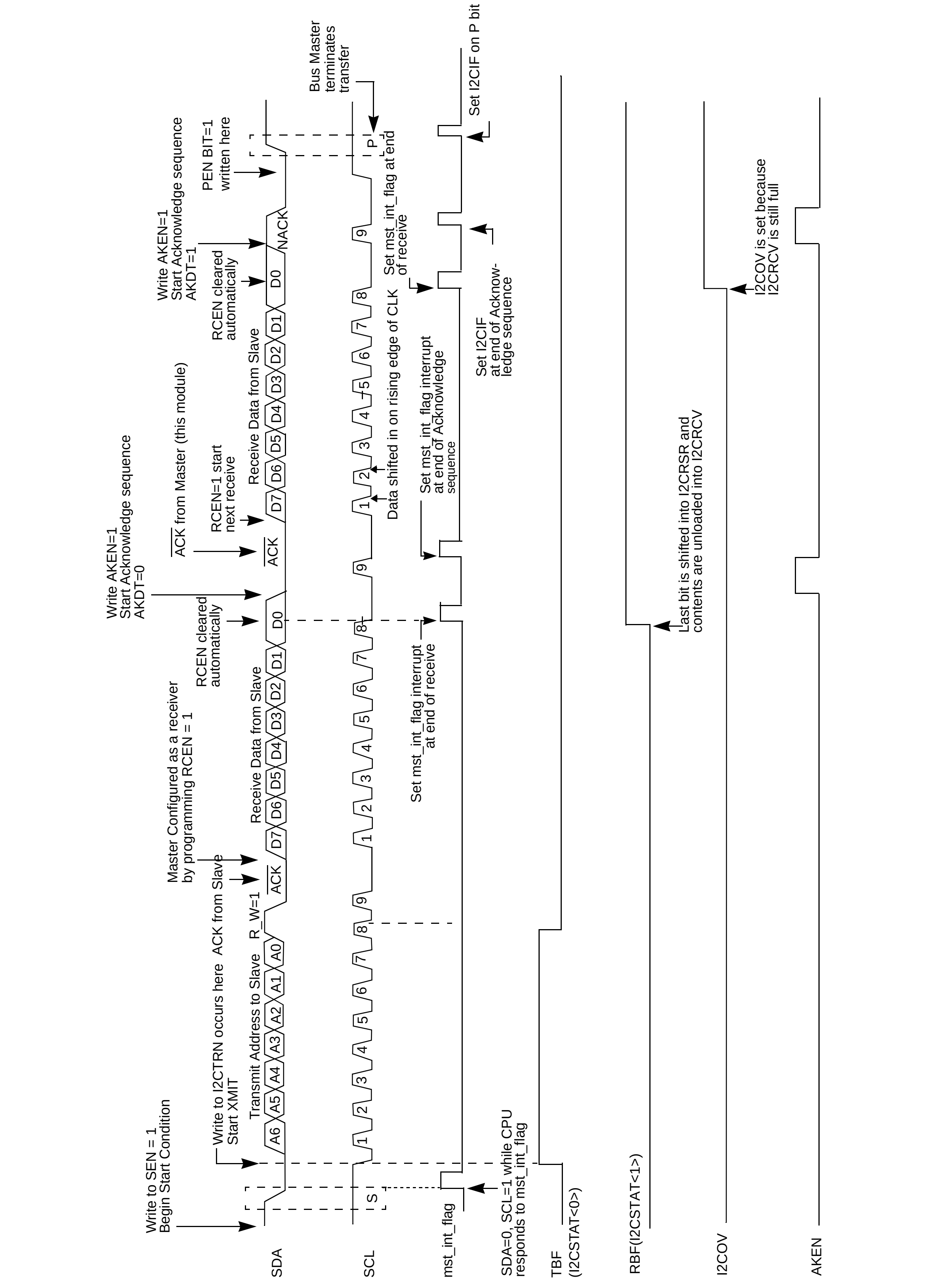
In receive operation, I2COV is set when 8 bits are received into the I2CRSR, and the RBF flag is already set from a previous reception.

IWCOL Status Flag:

If the user writes the I2CTRN when a receive is already in progress, then IWCOL is set and the contents of the buffer are unchanged (the write does not occur).

The D_A and R_W status bits are not applicable in Master mode.

FIGURE 18-11: I^{2}C MASTER MODE TIMING (RECEPTION, 7 BIT ADDRESS)


SDA
SCL
mst_int_flag
SDA=0, SCL=1 while CPU responds to mst_int_flag
TBF (I2CSTAT<0>)
RBF(I2CSTAT<1>)
I2COV
AKEN
Write to SEN = 1 Begin Start Condition
S
Write to I2CTRN occurs here Start XMIT
Transmit Address to Slave
A6
A5
A4
A3
A2
A1
A0
R_W=1
ACK from Slave
ACK
Master Configured as a receiver by programming RCEN = 1
Receive Data from Slave
D7
D6
D5
D4
D3
D2
D1
D0
RCEN cleared automatically
Write AKEN=1 Start Acknowledge sequence AKDT=0
ACK from Master (this module)
RCEN=1 start next receive
ACK
Receive Data from Slave
D7
D6
D5
D4
D3
D2
D1
D0
RCEN cleared automatically
Write AKEN=1 Start Acknowledge sequence AKDT=1
NACK
PEN BIT=1 written here
P
Bus Master terminates transfer
1
2
3
4
5
6
7
8
9
Set mst_int_flag interrupt at end of receive
Data shifted in on rising edge of CLK
Set mst_int_flag interrupt at end of Acknowledge sequence
Set mst_int_flag at end of receive
Set mst_int_flag at end
Set I2CIF at end of Acknow-ledge sequence
Set I2CIF on P bit
Last bit is shifted into I2CRSR and contents are unloaded into I2CRCV
I2COV is set because I2CRCV is still full

18.12.3 BAUD RATE GENERATOR

In I^{2}C 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. In I^{2}C Master mode, the BRG is not reloaded automatically. If clock arbitration is taking place, for instance, the BRG is reloaded when the SCL pin is sampled high.

EQUATION 18-1: SERIAL CLOCK RATE

$$FSCK = (F_{OSC}/2) / (I2CBRG * 2)$$

18.12.4 CLOCK ARBITRATION

Clock arbitration occurs when the master de-asserts 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 guarantees 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.

18.12.5 MULTI-MASTER COMMUNICATION, BUS COLLISION, AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a 1 on SDA, by letting SDA float high and 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 Bus Collision Interrupt (BCLIF) pulse and reset the master portion of the I^{2}C 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 de-asserted, and the I2CTRN can now be written to. When the user services the bus collision Interrupt Service Routine, and if the I^{2}C 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 de-asserted, 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 I^{2}C 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 I2CIF 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 Multi-Master mode, the interrupt generation on the detection of START and STOP conditions allows the determination of when the bus is free. Control of the I^{2}C bus can be taken when the P bit is set in the I2CSTAT register, or the bus is IDLE and the S and P bits are cleared.

18.12.5.1 Bus Collision During a START Condition

During a START condition, a bus collision occurs if:

1. SDA or SCL are sampled low at the beginning of the START condition.
2. SCL is sampled low before SDA is asserted low.

18.12.5.2 Bus Collision During a RESTART Condition

During a RESTART condition, bus collision occurs if:

1. A 0 is sampled on SDA, when SCL goes from 0 to 1.
2. SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data 1.

18.12.5.3 Bus Collision During a STOP Condition

Bus collision occurs during a STOP condition if:

1. After the SDA pin has been de-asserted and allowed to float high, SDA is sampled low after the BRG has timed out.
2. After the SCL pin is de-asserted, SCL is sampled low before SDA goes high.

18.13 I^{2}C Module Operation During CPU SLEEP and IDLE Modes

18.13.1 I^{2}C OPERATION DURING CPU SLEEP MODE

When the device enters SLEEP mode, all Q 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.

18.13.2 I^{2}C OPERATION DURING CPU IDLE MODE

For the I^{2}C, 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 18-1: 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	—	—	—	—	—	—	—	—	Receive Register								uuuu uuuu 0000 0000
I2CTRN	0202	—	—	—	—	—	—	—	—	Transmit Register								uuuu uuuu 0000 0000
I2CBRG	0204	—	—	—	—	—	—	—	Baud Rate Generator									uuuu uuu0 0000 0000
I2CCON	0206	I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN	GCEN	STREN	AKDT	AKEN	RCEN	PEN	RSEN	SEN	0u00 0000 0000 0000
I2CSTAT	0208	AKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10	IWCOL	I2COV	D_A	P	S	R_W	RBF	TBF	00uu u000 0000 0000
I2CADD	020A	—	—	—	—	—	—	Address Register										uuuu uu00 0000 0000

REGISTER 18-1: I2CSTAT: I^{2}C STATUS REGISTER

Upper Half:

R-0 HS, HC	R-0 HS, HC	U-0	U-0	U-0	U-0	R-0 HS, HC	R/W-0
AKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10
bit 15							bit 8

Lower Half:

R/W-0 HS	R/W-0 HS	R-0 HS, HC	R-0 HS, HC	R-0 HS, HC	R-0 HS, HC	R-0 HS, HC	R-0 HS, HC
IWCOL	I2COV	D_A	P	S	R_W	RBF	TBF
bit 7							bit 0

bit 15 **AKSTAT:** Acknowledge Status bit (In I^{2}C master mode only. Applicable to Master Transmit mode.)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave.

bit 14 **TRSTAT:** Transmit Status bit (In I^{2}C master mode only. Applicable to Master Transmit mode.)
1 = Master transmit is in progress (8 bits + ACK)
0 = Master transmit is not in progress.
Or'ing this bit with SEN, RCEN, PEN and AKEN will indicate if the I^{2}C master circuitry is active.

bit 13-11 **Unimplemented:** Read as '0'

bit 10 **BCL:** Master Bus Collision Detect bit (cleared when the I^{2}C module is disabled, I2CEN = 0)
1 = A bus collision has been detected during a master operation
0 = No collision

bit 9 **GCSTAT:** General Call Status bit
1 = General call address was received
0 = General call address was not received

bit 8 **ADD10:** 10-bit Address Status bit
1 = 10-bit address was matched
0 = 10-bit address was not matched

bit 7 **IWCOL:** Write Collision Detect bit
1 = An attempt to write the I2CTRN register failed because the I^{2}C module is busy (must be cleared in software)
0 = No collision

bit 6 **I2COV:** Receive Overflow Flag bit
1 = A byte is received while the I2CRCV register is still holding the previous byte. I2COV is a "don't care" in Transmit mode. I2COV must be cleared in software in either mode.
0 = No overflow

bit 5 **D_A:** Data/Address bit (valid only for Slave mode operation)
1 = Indicates that the last byte received was data
0 = Indicates that the last byte received was address

bit 4 **P:** STOP bit (This bit is updated when START, RESTART or STOP detected. Cleared when the I^{2}C module is disabled, I2CEN = 0.)
1 = Indicates that a STOP bit has been detected last
0 = STOP bit was not detected last

bit 3 **S:** START bit (This bit is updated when START, RESTART or STOP detected. Cleared when the I^{2}C module is disabled, I2CEN = 0.)
1 = Indicates that a START (or RESTART) bit has been detected last
0 = START bit was not detected last

bit 2 **R_W:** Read/Write bit Information (valid only for Slave mode operation)
1 = Read - indicates data transfer is output from slave
0 = Write - indicates data transfer is input to slave

REGISTER 18-1: I2CSTAT: I^{2}C STATUS REGISTER (Continued)

bit 1 **RBF:** Receive Buffer Full Status bit (cleared by reading I2CRCV)
`1` = Receive complete, I2CRCV is full
`0` = Receive not complete, I2CRCV is empty

bit 0 **TBF:** Transmit Buffer Full Status bit
`1` = Transmit in progress, I2CTRN is full (8 bits)
`0` = Transmit complete, I2CTRN is empty

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
HC = Cleared by Hardware	HS = Set by Hardware	-n = Value at POR	
1 = bit is set	0 = bit is cleared	x = bit is unknown	x = bit is unknown

REGISTER 18-2: I2CCON: I^{2}C CONTROL REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0 HC	R/W-0	U-0	R/W-0	R/W-0
I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0 HC	R/W-0 HC	R/W-0 HC	R/W-0 HC	R/W-0 HC
GCEN	STREN	AKDT	AKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

bit 15 **I2CEN:** I^{2}C Enable bit (only writable from software)
1 = Enables the I^{2}C module and configures the SDA and SCL pins as serial port pins
0 = Disables I^{2}C module. All I^{2}C pins are controlled by PORT functions. Power consumption is minimal.

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **I2CSIDL:** Stop in IDLE Mode bit
1 = Discontinue module operation when device enters a IDLE mode
0 = Continue module operation in IDLE mode

bit 12 **SCLREL:** SCL Release Control bit (in I^{2}C Slave mode only)

If STREN = 0:
1 = Release clock
0 = Force clock low (clock stretch). Automatically cleared to 0 at beginning of slave transmission.

If STREN = 1:
1 = Release clock
0 = Holds clock low (clock stretch). (User may program this bit to 0, will clock stretch at next SCL low.)
Automatically cleared to 0 at beginning of slave transmission.
Automatically cleared to 0 at end of slave reception.

bit 11 **IPMIEN:** Intelligent Peripheral Management Interface (IPMI) Enable bit
1 = Enable IPMI Support mode. All addresses Acknowledged.
0 = IPMI mode not enabled

bit 10 **A10M:** Indicates a 10-bit Slave Address bit
1 = I2CADD is a 10-bit slave address
0 = I2CADD is a 7-bit slave address

bit 9 **DISSLW:** Disable Slew Rate Control bit
1 = Slew rate control disabled for Standard Speed mode (100 kHZ). (Also disabled for 1 MHz mode.)
0 = Slew rate control enabled for High Speed mode (400 kHz)

bit 8 **SMEN:** SM bus Input Levels bit
1 = Enable input logic so that thresholds are compliant with SM bus specification
0 = Disable SM bus specific inputs

bit 7 **GCEN**: General Call Enable bit (in I^{2}C Slave mode only)
1 = Enable interrupt when a general call address is received in the I2CSR. (Module is enabled for reception.)
0 = General call address disabled.

bit 6 **STREN:** SCL Clock Stretch Enable bit (in I^{2}C slave mode only, used in conjunction with SCLREL bit)
1 = Enable clock stretching
0 = Disable clock stretching.

bit 5 **AKDT:** Acknowledge Data bit (In I^{2}C Master mode only. Applicable during master receive.)
Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

bit 4 **AKEN:** Acknowledge Sequence Enable bit (In I^{2}C Master mode only. Applicable during master receive.)
1 = Initiate Acknowledge sequence on SDA and SCL pins, and transmit AKDT data bit.
Automatically cleared by hardware at end of master Acknowledge sequence.
0 = Acknowledge sequence IDLE

REGISTER 18-2: I2CCON: I^{2}C CONTROL REGISTER (Continued)

bit 3 **RCEN:** Receive Enable bit (in I^{2}C Master mode only)
1 = Enables Receive mode for I^{2}C. Automatically cleared by hardware at end eighth bit of receive data byte.
0 = Receive sequence not in progress

bit 2 **PEN:** STOP Condition Enable bit (in I^{2}C Master mode only)
1 = Initiate STOP condition on SDA and SCL pins. Automatically cleared by hardware at end of master STOP sequence.
0 = STOP condition IDLE

bit 1 **RSEN:** RESTART Condition Enabled bit (in I^{2}C Master mode only)
1 = Initiate RESTART condition on SDA and SCL pins. Automatically cleared by hardware at end of master RESTART sequence.
0 = RESTART condition IDLE

bit 0 **SEN:** START Condition Enabled bit (in I^{2}C Master mode only)
1 = Initiate START condition on SDA and SCL pins. Automatically cleared by hardware at end of master START sequence.
0 = START condition IDLE

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
HC = Cleared by Hardware	HS = Set by Hardware	-n = Value at POR
1 = bit is set	0 = bit is cleared	x = bit is unknown

NOTES:

19.0 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART) MODULE

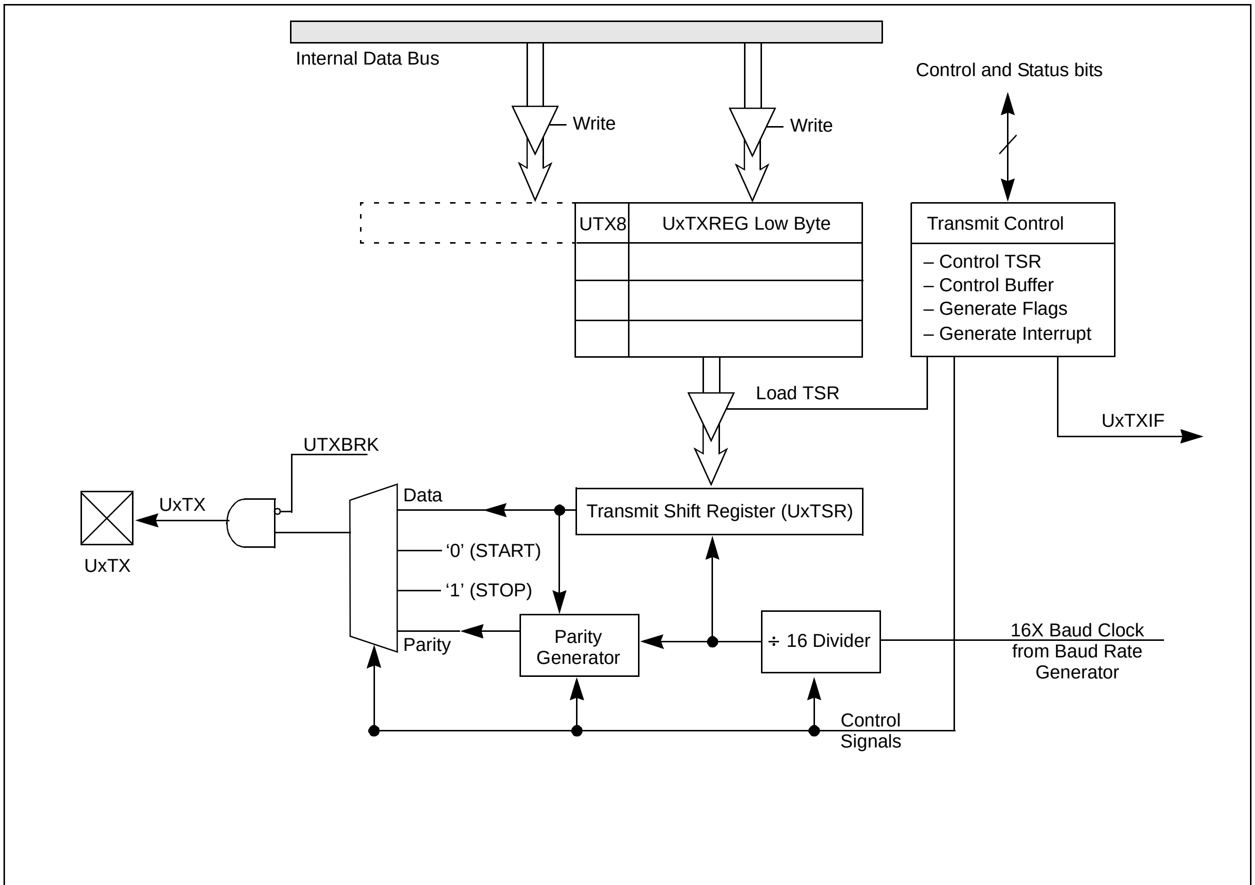
This section describes the Universal Asynchronous Receiver/Transmitter Communications module.

19.1 UART Module Overview

The key features of the UART module are:

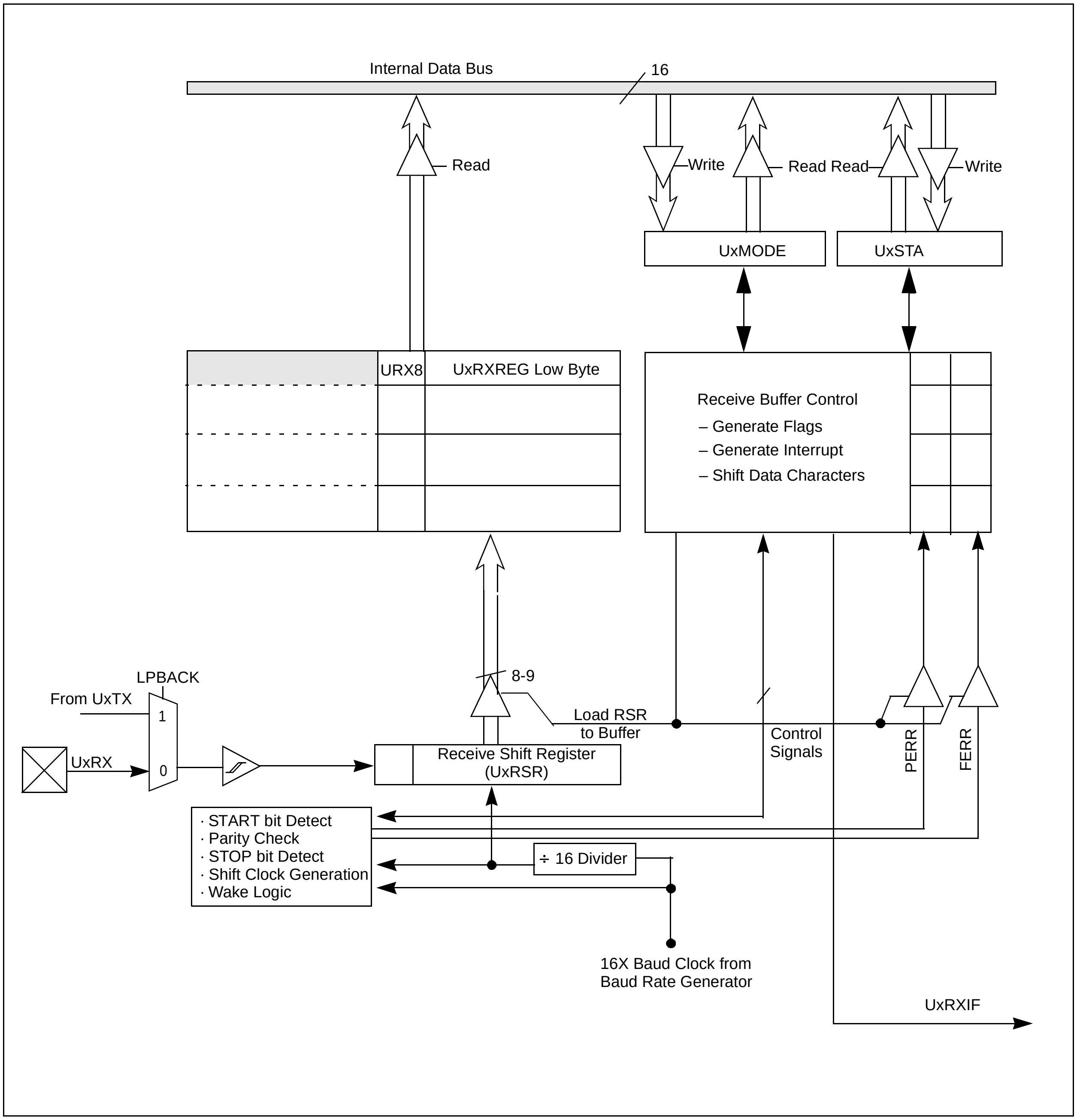
- Full-duplex, 8 or 9-bit data transmission
- Even, Odd or No Parity options (for 8-bit data)
- One or two STOP bits
- Fully integrated Baud Rate Generator with 16-bit prescaler
- Baud rates range from 38 bps to 1.875 Mbps at 30 MIPS
- 4-deep transmit data buffer
- 4-deep receive data buffer
- Parity, 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 19-1: UART TRANSMITTER BLOCK DIAGRAM



Note: x = 1 or 2.

FIGURE 19-2: UART RECEIVER BLOCK DIAGRAM



19.2 Enabling and Setting Up UART

19.2.1 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 pin. The UxTX pin is at logic ‘1’ when no transmission is taking place.

19.2.2 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.

19.2.3 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.

19.2.4 SETTING UP DATA, PARITY AND STOP BIT SELECTIONS

Control bits PDSEL<1:0> in the UxSTA 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, 1 STOP bit (typically represented as 8, N, 1).

19.3 Transmitting Data

19.3.1 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, and then, the Transmit and Receive Interrupt enable and priority bits are setup 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 data byte to be transmitted, to the lower byte of UxTXREG. The value will be transferred to Transmit Shift register (UxTSR) immediately and the serial bit stream will start shifting out during the next rising edge of the baud clock.
5. A Transmit interrupt will be generated depending on the value of the interrupt control bit UTXISEL (UxSTA<15>).

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.

19.3.2 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.

19.3.3 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 a Power Saving mode, or wakes up from a Power Saving mode.

19.3.4 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 of 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 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 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.

19.3.5 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.

19.4 Receiving Data

19.4.1 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 19.3.1).
2. Enable the UART (see Section 19.3.1).
3. A receive interrupt will be generated when one or more data bytes 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.

19.4.2 RECEIVE BUFFER (UxRXB)

The receive buffer is 4-deep. Including the Receive Shift register (UxRSR), the user effectively has a 5-deep FIFO (First In First Out) 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 a power-saving mode or wakes up from a power-saving mode.

19.4.3 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 and as a result, the Receive Buffer contains 3 or 4 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 and as a result, the Receive Buffer contains 4 characters (i.e., becomes full).

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

19.5 Reception Error Handling

19.5.1 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 remain valid.

19.5.2 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.

19.5.3 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.

19.5.4 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.

19.5.5 RECEIVE BREAK

The receiver will count and expect a certain number of bit times based on the values programmed in the PDSEL (UxMODE<2:1>) and STSEL (UxMODE<0>) bits.

If the break is much 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.

19.6 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 transmission. The URXISEL control bit does not have any impact on interrupt generation in this mode, since an interrupt (if enabled) will be generated if the received word has the 9th bit set.

19.7 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:

a) Configure UART for desired mode of operation.

a) Set LPBACK = 1 to enable Loopback mode.

b) Enable transmission as defined in Section 19.3.

19.8 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)

f_{CY} = Instruction Clock Rate (1/TCY)

Then Baud Rate = f_{CY} / (16* (BRG + 1))

Therefore, maximum baud rate possible is = f_{CY} /16
(If BRG = 0),

and minimum baud rate possible is = f_{CY} / (16* 65536).

With a full 16-bit baud rate generator, at 30 MIPS operation, the minimum baud rate achievable is 28.5 bps.

19.9 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. To enable this mode, the user must program the input capture module to detect the falling and rising edges of the START bit.

19.10 UART Operation During CPU SLEEP and IDLE Modes

19.10.1 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, transmit and receive registers and buffers, and the UxBRG register are not affected by SLEEP mode.

If the WAKE bit (UxSTA<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.

19.10.2 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 19-1: 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	—	—	ALTIO	—	—	WAKE	LPBACK	ABAUD	—	—	PDSEL1	PDSEL0	STSEL	0u0u u0uu 000u u000
U1STA	020E	UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0uuu 0001 0001 0000
U1TXREG	0210	—	—	—	—	—	—	—	UTX8	Transmit Register								uuuu uuu0 0000 0000
U1RXREG	0212	—	—	—	—	—	—	—	URX8	Receive Register								uuuu uuu0 0000 0000
U1BRG	0214	Baud Rate Generator Prescaler																0000 0000 0000 0000

TABLE 19-2: UART2 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
U2MODE	0216	UARTEN	—	USIDL	—	—	ALTIO	—	—	WAKE	LPBACK	ABAUD	—	—	PDSEL1	PDSEL0	STSEL	0u0u u0uu 000u u000
U2STA	0218	UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0uuu 0001 0001 0000
U2TXREG	021A	—	—	—	—	—	—	—	UTX8	Transmit Register								uuuu uuu0 0000 0000
U2RXREG	021C	—	—	—	—	—	—	—	URX8	Receive Register								uuuu uuu0 0000 0000
U2BRG	021E	Baud Rate Generator Prescaler																0000 0000 0000 0000

REGISTER 19-1: UxMODE: UARTx MODE REGISTER

Upper Half:

R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	U-0	U-0
UARTEN	—	USIDL	—	—	ALTIO	—	—
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
WAKE	LPBACK	ABAUD	—	—	PDSEL1	PDSEL0	STSEL
bit 7							bit 0

bit 15 **UARTEN:** UART Enable bit
1 = UART is enabled, all UART pins are controlled by UART
0 = UART is disabled, all UART pins are controlled by PORT latches. UART power consumption is minimal.

bit 14 **Unimplemented:** Read as '0'

bit 13 **USIDL:** Stop in IDLE Mode bit
1 = Discontinue module operation when device enters an IDLE mode
0 = Continue module operation in IDLE mode

bit 12-11 **Unimplemented:** Read as '0'

bit 10 **ALTIO:** UART Alternate I/O Selection bit (bit is U-0 when UALTIO configuration signal = 0)
1 = UART communicates on UxATX and UxARX signals
0 = UART communicates on UxTX and UxRX signals

bit 9-8 **Unimplemented:** Read as '0'

bit 7 **WAKE:** Enable Wake-up on START bit Detect During SLEEP Mode bit
1 = Enable wake-up
0 = No wake-up enabled

bit 6 **LPBACK:** UART Loopback Mode Select bit
1 = Enable Loopback mode
0 = Loopback mode is disabled

bit 5 **ABAUD:** Auto Baud Enable bit
1 = Input to Capture module from UxRX Pin
0 = Input to Capture module from ICx Pin

bit 4-3 **Unimplemented:** Read as '0'

bit 2-1 **PDSEL<1:0>:** Parity and Data bits Selection bits
11 = 9-bit data, no parity
10 = 8-bit data, odd parity
01 = 8-bit data, even parity
00 = 8-bit data, no parity

bit 0 **STSEL:** STOP bit Selection bit
1 = 2 STOP bits
0 = 1 STOP bit

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

REGISTER 19-2: UxSTA: UARTx STATUS AND CONTROL REGISTER

Upper Half:							
R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R-0	R-1
UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA
bit 7							bit 0

bit 15 **UTXISEL:** Transmission Interrupt Mode Selection bit
1 = Interrupt when a character is transferred to the Transmit Shift register and as a result, the transmit buffer becomes empty
0 = Interrupt when a character is transferred to the Transmit Shift register (this implies that there is at least one character open in the Transmit buffer)

bit 14-12 **Unimplemented:** Read as '0'

bit 11 **UTXBRK :** Transmit Break bit
1 = UxTX pin taken low, regardless of state of transmitter
0 = UxTX pin in its normal condition

bit 10 **UTXEN:** Transmit Enable bit
1 = Transmit enabled, UxTX pin controlled by UART
0 = Transmit disabled, any pending transmission is aborted and buffer is reset. UxTX pin controlled by PORT.

bit 9 **UTXBF:** Transmit Buffer Full Status bit (Read Only)
1 = Transmit buffer is full
0 = Transmit buffer is not full, at least one more character can be written

bit 8 **TRMT:** Transmit Shift Register is Empty bit (Read Only)
1 = Transmit shift register is empty and transmit buffer is empty (the last transmission has completed)
0 = Transmit shift register is not empty, a transmission is in progress or queued

bit 7-6 **URXISEL<1:0>:** Receive Interrupt Mode Selection bits
11 = Interrupt is set when Receive buffer is full, i.e., has 4 data characters
10 = Interrupt is set when Receive buffer is 3/4 full, i.e., has 3 data characters
0x = Interrupt is set when a character is received and transferred from the RSR to the Receive Buffer. Receive buffer has one or more characters.

bit 5 **ADDEN:** Address Character Detect bit (bit 8 of received data = 1)
1 = Address Detect mode enabled. If 9-bit mode is not selected, this does not take effect.
0 = Address Detect mode disabled

bit 4 **RIDLE:** Receiver IDLE bit (Read Only)
1 = Receiver is IDLE
0 = Receiver is active

bit 3 **PERR:** Parity Error Status bit (Read Only)
1 = Parity Error has been detected for the current character (the character at top of the receive FIFO)
0 = Parity Error has not been detected

bit 2 **FERR:** Framing Error Status bit (Read Only)
1 = Framing Error has been detected for the current character (the character at top of the receive FIFO)
0 = Framing Error has not been detected

bit 1 **OERR:** Receive Buffer Overrun Error Status bit (Read/Clear Only)
1 = Receive buffer has overflowed
0 = Receive buffer has not overflowed
(Clearing a previously set OERR bit (1 -> 0 transition) will reset the receiver buffer and the RSR to empty state.)

REGISTER 19-2: UxSTA: UARTx STATUS AND CONTROL REGISTER (Continued)

bit 0 **URXDA:** Receive Buffer Data Available bit (Read Only)
1 = Receive buffer has data, at least one more character can be read
0 = Receive buffer is empty

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

REGISTER 19-3: UxRXREG: UARTx RECEIVE REGISTER

Upper Half:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	R-0
—	—	—	—	—	—	—	URX8
bit 15							bit 8

Lower Half:							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
URX7	URX6	URX5	URX4	URX3	URX2	URX1	URX0
bit 7							bit 0

bit 15-9 **Unimplemented:** Read as '0'

bit 8 **URX8:** Data bit 8 of the Received Character (in 9-bit mode)

bit 7-0 **URX<7:0>:** Data bits 7-0 of the Received Character

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

REGISTER 19-4: UxTXREG: UARTx TRANSMIT REGISTER (NORMALLY WRITE ONLY)

Upper Half:							
U-x	U-x	U-x	U-x	U-x	U-x	U-x	W-x
—	—	—	—	—	—	—	UTX8
bit 15							bit 8

Lower Half:							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
UTX7	UTX6	UTX5	UTX4	UTX3	UTX2	UTX1	UTX0
bit 7							bit 0

bit 15-9 **Unimplemented:** Read as '0'

bit 8 **UTX8:** Data bit 8 of the Transmitted Character (in 9-bit mode)

bit 7-0 **UTX<7:0>:** Data bits 7-0 of the Transmitted Character

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

REGISTER 19-5: UxBRG: UARTx BAUD RATE REGISTER

Upper Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG09	BRG08
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BRG07	BRG06	BRG05	BRG04	BRG03	BRG02	BRG01	BRG00
bit 7							bit 0

bit 15-0 **BRG<15:0>:** Baud Rate Divisor bits

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

20.0 CAN MODULE

20.1 Overview

The Controller Area Network (CAN) module is a serial interface, useful for communicating with other peripherals or microcontroller devices. This interface/protocol was designed to allow communications within noisy environments.

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 CAN1.2, CAN2.0A and CAN2.0B
- 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
- 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
- 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 timer module for time-stamping and network synchronization
- Low power SLEEP mode

20.1.1 OVERVIEW OF THE MODULE

The CAN bus module consists of a Protocol Engine and message buffering and 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 2 receive registers.

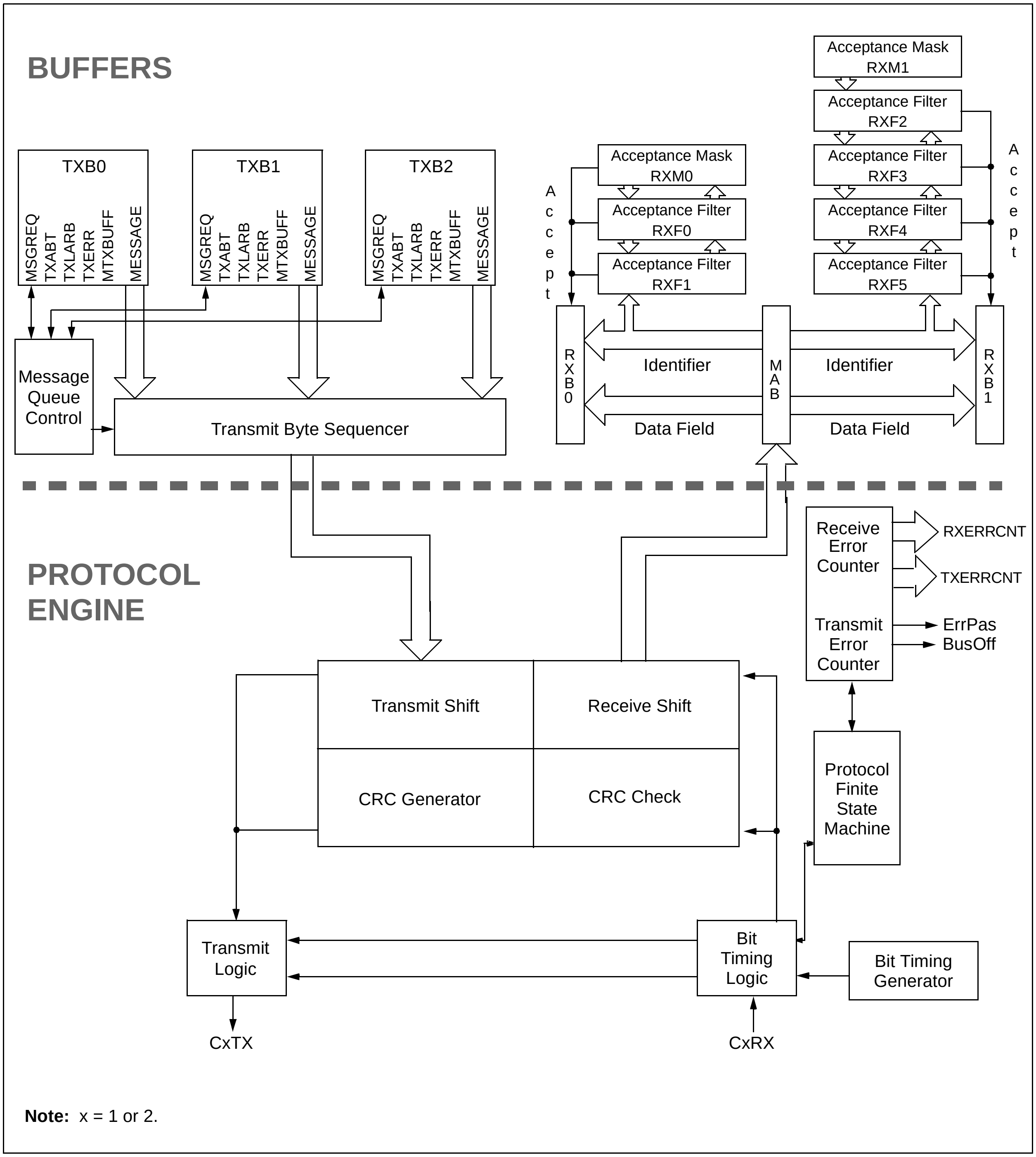
The CAN Module supports the following Frame types:

- Standard Data Frame
- Extended Data Frame
- Remote Frame
- Error Frame
- Overload Frame Reception
- Interframe Space

20.1.2 TRANSMIT/RECEIVE BUFFERS

The dsPIC30F has three transmit and two receive buffers, two acceptance masks (one for each receive buffer), and a total of six acceptance filters. Figure 20-1 is a block diagram of these buffers and their connection to the protocol engine.

FIGURE 20-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM



Note: x = 1 or 2.

21.0 DATA CONVERTER INTERFACE (DCI) MODULE

21.1 Module Introduction

The dsPIC30F Data Converter Interface (DCI) module allows simple interfacing of devices such as audio coder/decoders (Codecs), A/D converters, and D/A converters. The following interfaces are supported:

- Framed Synchronous Serial Transfer (Single or Multi-Channel)
- Inter-IC Sound (I^2S) Interface
- AC-Link compliant mode

The DCI module provides the following general features:

- Programmable word size up to 16-bits
- Support for up to 16 time slots, for a maximum frame size of 256 bits
- Data buffering for up to 4 samples without CPU overhead

21.2 Module I/O Pins

There are four I/O pins associated with the module. When enabled, the module controls the data direction of each of the four pins.

21.2.1 CSCK PIN

The CSCK pin provides the serial clock for the DCI module. The CSCK pin may be configured as an input or output, using the CSCKD control bit in the DCICON2 SFR (see Register 21-2). When configured as an output, the serial clock is provided by the dsPIC30F. When configured as an input, the serial clock must be provided by an external device.

21.2.2 CSDO PIN

The serial data output (CSDO) pin is configured as an output only pin when the module is enabled. The CSDO pin drives the serial bus whenever data is to be transmitted. The CSDO pin is tri-stated or driven to ‘0’ during CSCK periods when data is not transmitted, depending on the state of the CSDOM control bit. This allows other devices to place data on the serial bus during transmission periods not used by the DCI module.

21.2.3 CSDI PIN

The serial data input (CSDI) pin is configured as an input only pin when the module is enabled.

21.2.3.1 COFS PIN

The Codec frame synchronization (COFS) pin is used to synchronize data transfers that occur on the SDO and CSDI pins. The COFS pin may be configured as an input or an output. The data direction for the COFS pin is determined by the COFSD control bit in the DCICON1 Register (see Register 21-1).

The DCI module accesses the shadow registers while the CPU is in the process of accessing the memory mapped buffer registers.

21.2.4 BUFFER DATA ALIGNMENT

Data values are always stored left justified in the buffers, since most Codec data is represented as a signed 2’s complement fractional number. If the received word length is less than 16-bits, the unused LS bits in the receive buffer registers are set to 0 by the module. If the transmitted word length is less than 16 bits, the unused LS bits in the transmit buffer register are ignored by the module. The word length setup is described in subsequent sections of this document.

21.2.5 TRANSMIT/RECEIVE SHIFT REGISTER

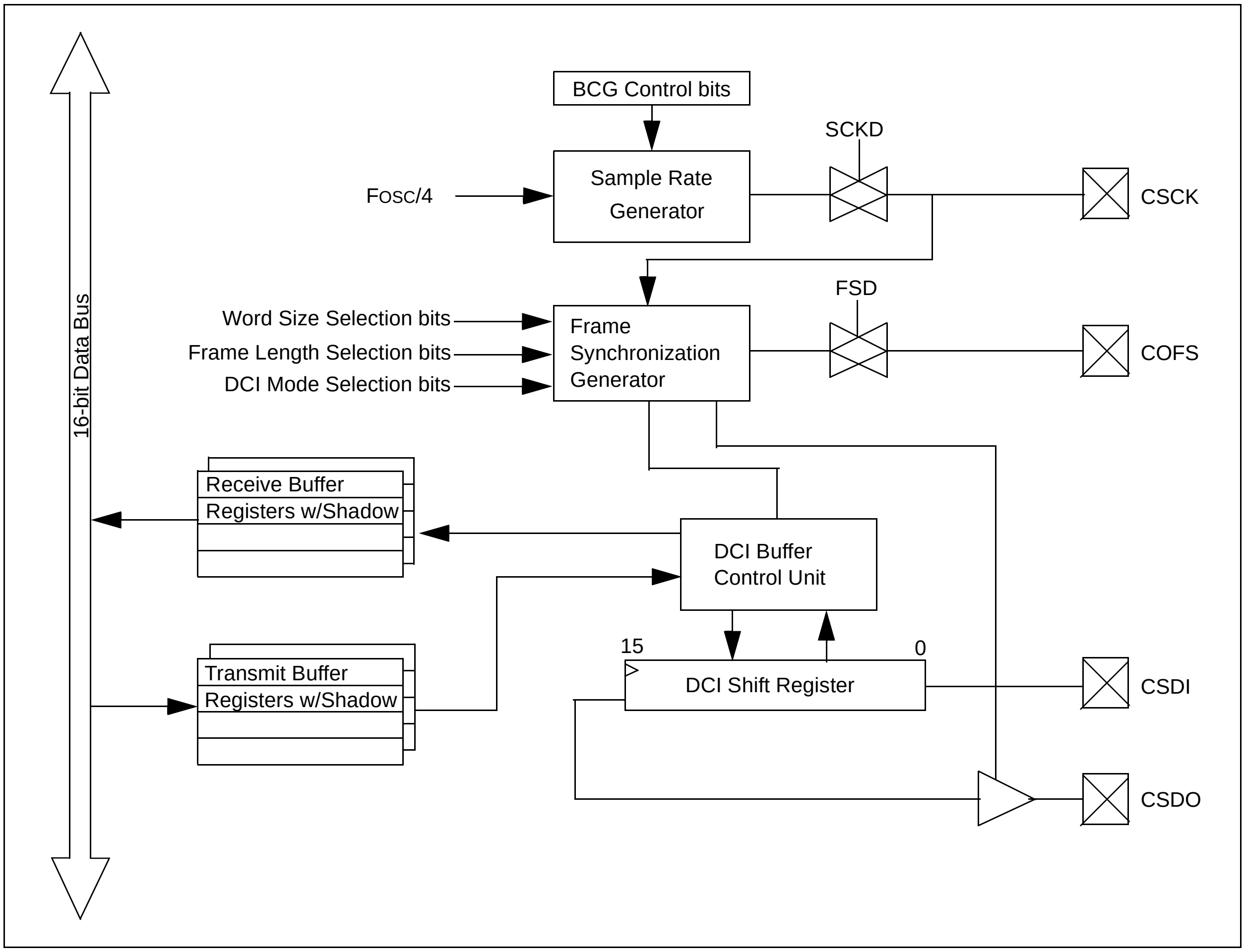
The DCI module has a 16-bit shift register for shifting serial data in and out of the module. Data is shifted in/out of the shift register MS bit first, since audio PCM data is transmitted in signed 2’s complement format.

21.2.6 DCI BUFFER CONTROL

The DCI module contains a buffer control unit for transferring data between the shadow buffer memory and the serial shift register. The buffer control unit is a simple 2-bit address counter that points to word locations in the shadow buffer memory. For the receive memory space (high address portion of DCI buffer memory), the address counter is concatenated with a ‘0’ in the MSb location to form a 3-bit address. For the transmit memory space (high portion of DCI buffer memory), the address counter is concatenated with a ‘1’ in the MSb location.

Note:	The DCI buffer control unit always accesses the same relative location in the transmit and receive buffers, so only one address counter is provided.

FIGURE 21-1: DCI MODULE BLOCK DIAGRAM



21.3 DCI Module Operation

21.3.1 MODULE ENABLE

The DCI module is enabled or disabled by setting/clearing the DCIEN control bit in the DCICON1 SFR. Clearing the DCIEN control bit has the effect of resetting the module. In particular, all counters associated with CSCK generation, frame sync, and the DCI buffer control unit are RESET.

The DCI clocks are shut-down when the DCIEN bit is cleared.

When enabled, the DCI controls the data direction for the four I/O pins associated with the module. The PORT, LAT and TRIS register values for these I/O pins are overridden by the DCI module when the DCIEN bit is set.

It is also possible to override the CSCK pin separately when the bit clock generator is enabled. This permits the bit clock generator to operate, without enabling the rest of the DCI module.

21.3.2 WORD SIZE SELECTION BITS

The WS<3:0> word size selection bits in the DCICON2 SFR (see Register 21-2) determine the number of bits in each DCI data word. Essentially, the WS<3:0> bits determine the counting period for a 4-bit counter clocked from the CSCK signal.

Any data length, up to 16-bits may be selected. The value loaded into the WS<3:0> bits is one less the desired word length. For example, a 16-bit data word size is selected when WS<3:0> = `1111`.

Note:	These WS<3:0> control bits are used only in the Multi-Channel and I^{2}S modes. These bits have no effect in AC-Link mode, since the data slot sizes are fixed by the protocol.

21.3.3 FRAME SYNC GENERATOR

The frame sync generator (COFSG) is a 4-bit counter that sets the frame length in data words. The frame sync generator is incremented each time the word size counter is reset (refer to Section 21.3.2). The period for the frame synchronization generator is set by writing the COFSG<3:0> control bits in the DCICON2 SFR. The COFSG period in clock cycles is determined by the following formula:

EQUATION 21-1: COFSG PERIOD

$$\text{FrameLength} = \text{WordLength} \bullet (\text{FSGvalue} + 1)$$

Frame lengths, up to 16 data words may be selected. The frame length in CSCK periods can vary up to a maximum of 256 depending on the word size that is selected.

> **Note:** The COFSG control bits will have no effect in AC-Link mode, since the frame length is set to 256 CSCK periods by the protocol.

21.3.4 FRAME SYNC MODE CONTROL BITS

The type of frame sync signal is selected using the Frame Synchronization mode control bits (COFSM<1:0>) in the DCICON1 SFR (see Register 21-1). The following operating modes can be selected:

- Multi-Channel mode
- I^{2}S mode
- AC-Link mode (16-bit)
- AC-Link mode (20-bit)

The operation of the COFSM control bits depends on whether the DCI module generates the frame sync signal as a master device, or receives the frame sync signal as a slave device.

The master device in a DSP/Codec pair is the device that generates the frame sync signal. The frame sync signal initiates data transfers on the CSDI and CSDO pins and usually has the same frequency as the data sample rate (COFS).

The DCI module is a frame sync master, if the COFSD control bit is cleared and is a frame sync slave, if the COFSD control bit is set.

21.3.5 MASTER FRAME SYNC OPERATION

When the DCI module is operating as a frame sync master device (COFSD = 0), the COFSM mode bits determine the type of frame sync pulse that is generated by the frame sync generator logic.

A new COFS signal is generated when the frame sync generator resets to 0.

In the Multi-Channel mode, the frame sync pulse is driven high for the CSCK period to initiate a data transfer. The number of CSCK cycles between successive frame sync pulses will depend on the word size and frame sync generator control bits. A timing diagram for the frame sync signal in Multi-Channel mode is shown in Figure 21-2.

In the AC-Link mode of operation, the frame sync signal has a fixed period and duty cycle. The AC-Link frame sync signal is high for 16 CSCK cycles and is low for 240 CSCK cycles. A timing diagram with the timing details at the start of an AC-Link frame is shown in Figure 21-3.

In the I^{2}S mode, a frame sync signal having a 50% duty cycle is generated. The period of the I^{2}S frame sync signal in CSCK cycles is determined by the word size and frame sync generator control bits. A new I^{2}S data transfer boundary is marked by a high-to-low or a low-to-high transition edge on the COFS pin.

21.3.6 SLAVE FRAME SYNC OPERATION

When the DCI module is operating as a frame sync slave (COFSD = 1), data transfers are controlled by the Codec device attached to the DCI module. The COFSM control bits control how the DCI module responds to incoming COFS signals.

In the Multi-Channel mode, a new data frame transfer will begin one CSCK cycle after the COFS pin is sampled high (see Figure 21-2). The pulse on the COFS pin resets the frame sync generator logic.

In the I^{2}S mode, a new data word will be transferred one CSCK cycle after a low-to-high or a high-to-low transition is sampled on the COFS pin. A rising or falling edge on the COFS pin resets the frame sync generator logic.

In the AC-Link mode, the tag slot and subsequent data slots for the next frame will be transferred one CSCK cycle after the COFS pin is sampled high.

The COFSG and WS bits must be configured to provide the proper frame length when the module is operating in the Slave mode. Once a valid frame sync pulse has been sampled by the module on the COFS pin, an entire data frame transfer will take place. The module will not respond to further frame sync pulses until the data frame transfer has completed.

FIGURE 21-2: FRAME SYNC TIMING, MULTI-CHANNEL MODE

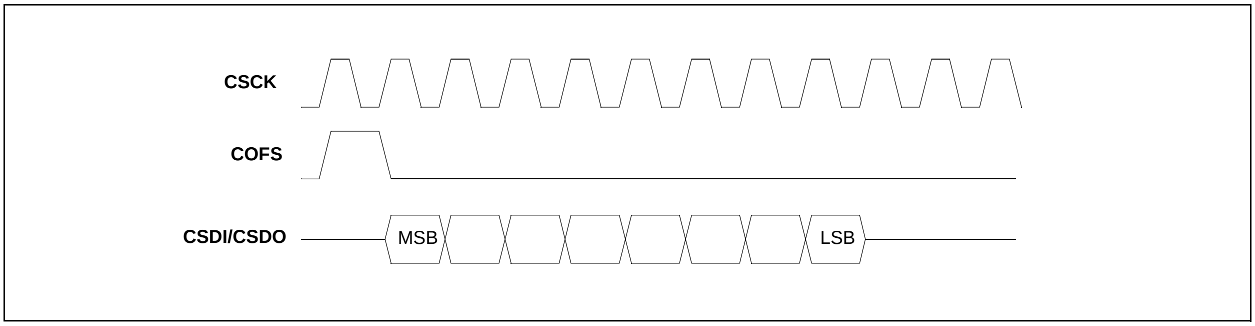


FIGURE 21-3: FRAME SYNC TIMING, AC-LINK START OF FRAME

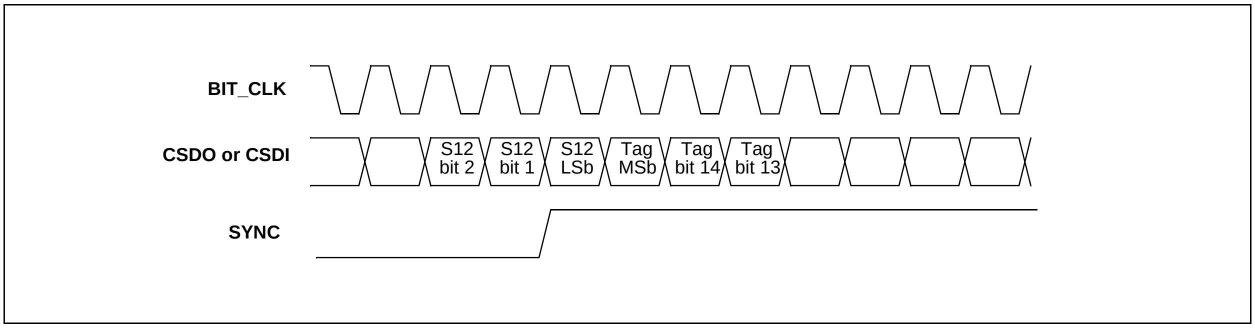
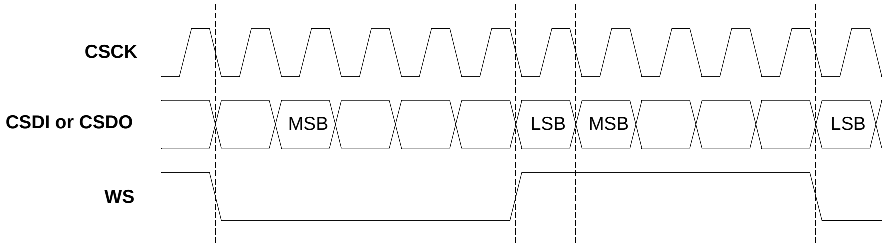


FIGURE 21-4: I²S INTERFACE FRAME SYNC TIMING



Note: A 5-bit transfer is shown here for illustration purposes. The I^{2}S protocol does not specify word length — this will be system dependent.

21.3.7 BIT CLOCK GENERATOR

The DCI module has a dedicated 12-bit time-base that produces the bit clock. The bit clock rate (period) is set by writing a non-zero 12-bit value to the BCG<11:0> control bits in the DCICON1 SFR.

When the BCG<11:0> bits are set to zero, the bit clock will be disabled. If the BCG<11:0> bits are set to a non-zero value, the bit clock generator is enabled. These bits should be set to '0' and the CSCKD bit set to 1, if the serial clock for the DCI is received from an external device.

The formula for the bit clock frequency is given in Equation 21-2.

EQUATION 21-2: BIT CLOCK FREQUENCY

$$f_{BCK} = \frac{f_{CYC}}{2 \bullet (BCG + 1)}$$

The required bit clock frequency will be determined by the system sampling rate and frame size. Typical bit clock frequencies range from 16x to 512x the converter sample rate, depending on the data converter and the communication protocol that is used.

To achieve bit clock frequencies associated with common audio sampling rates, the user will need to select a crystal frequency that has an 'even' binary value. Examples of such crystal frequencies are listed in Table 21-1.

TABLE 21-1: DEVICE FREQUENCIES FOR COMMON CODEC CSCK FREQUENCIES

FOSC	PLL	FCYC
2.048 MHz	16x	32.768 MIPS
4.096 MHz	8x	32.768 MIPS
4.800 MHz	8x	38.4 MIPS
9.600 MHz	4x	38.4 MIPS

Note 1: When the CSCK signal is applied externally (CSCKD = 1), the BCG<9:0> bits have no effect on the operation of the DCI module.

2: When the CSCK signal is applied externally (CSCKD = 1), the external clock high and low times must meet the device timing requirements.

21.3.8 SAMPLE CLOCK EDGE CONTROL BIT

The sample clock edge (CSCKE) control bit determines the sampling edge for the CSCK signal. If the CSCK bit is cleared (default), data will be sampled on the falling edge of the CSCK signal. The AC-Link protocols and most Multi-Channel formats require that data be sampled on the falling edge of the CSCK signal. If the CSCK bit is set, data will be sampled on the rising edge of CSCK. The I^{2}S protocol requires that data be sampled on the rising edge of the CSCK signal.

21.3.9 DATA JUSTIFICATION CONTROL BIT

In most applications, the data transfer begins one CSCK cycle after the COFS signal is sampled active. This is the default configuration of the DCI module. An alternate data alignment can be selected by setting the DJST control bit in the DCICON2 SFR. When DJST = 1, data transfers will begin during the same CSCK cycle when the COFS signal is sampled active.

21.3.10 TRANSMIT SLOT ENABLE BITS

The TSCON SFR (see Register 21-5) has control bits that are used to enable up to 16 time-slots for transmission. These control bits are the TSE<15:0> bits. The size of each time-slot is determined by the WS<3:0> word size selection bits and can vary up to 16-bits.

If a transmit time-slot is enabled via one of the TSE bits (TSEx = 1), the contents of the current transmit shadow buffer location will be loaded into the CSDO shift register and the DCI buffer control unit is incremented to point to the next location.

During an unused transmit time-slot, the CSDO pin will drive 0's or will be tri-stated during all disabled time-slots, depending on the state of the CSDOM bit in the DCICON1 SFR.

The data frame size in bits is determined by the chosen data word size and the number of data word elements in the frame. If the chosen frame size has less than 16 elements, the additional slot enable bits will have no effect.

Each transmit data word is written to the 16-bit transmit buffer as left justified data. If the selected word size is less than 16-bits, then the LS bits of the transmit buffer memory will have no effect on the transmitted data. The user should write 0's to the unused LS bits of each transmit buffer location.

21.3.11 RECEIVE SLOT ENABLE BITS

The RSCON SFR (see Register 21-6) contains control bits that are used to enable up to 16 time-slots for reception. These control bits are the RSE<15:0> bits. The size of each receive time-slot is determined by the WS<3:0> word size selection bits and can vary from 1 to 16 bits.

If a receive time-slot is enabled via one of the RSE bits (RSEx = 1), the shift register contents will be written to the current DCI receive shadow buffer location and the buffer control unit will be incremented to point to the next buffer location.

Data is not packed in the receive memory buffer locations if the selected word size is less than 16 bits. Each received slot data word is stored in a separate 16-bit buffer location. Data is always stored in a left justified format in the receive memory buffer.

21.3.12 SLOT ENABLE BITS OPERATION WITH FRAME SYNC

The TSE and RSE control bits operate in concert with the DCI frame sync generator. In the Master mode, a COFS signal is generated whenever the frame sync generator is reset. In the Slave mode, the frame sync generator is reset whenever a COFS pulse is received.

The TSE and RSE control bits allow up to 16 consecutive time-slots to be enabled for transmit or receive. After the last enabled time-slot has been transmitted/ received, the DCI will stop buffering data until the next occurring COFS pulse.

21.3.13 SYNCHRONOUS DATA TRANSFERS

The DCI buffer control unit will be incremented by one word location, whenever a given time-slot has been enabled for transmission or reception. In most cases, data input and output transfers will be synchronized, which means that a data sample is received for a given channel at the same time a data sample is transmitted. Therefore, the transmit and receive buffers will be filled with equal amounts of data when a DCI interrupt is generated.

In some cases, the amount of data transmitted and received during a data frame may not be equal. As an example, assume a two-word data frame is used. Furthermore, assume that data is only received during slot #0, but is transmitted during slot #0 and slot #1. In this case, the buffer control unit counter would be incremented twice during a data frame, but only one receive register location would be filled with data.

21.3.14 BUFFER LENGTH CONTROL

The amount of data that is buffered between interrupts is determined by the buffer length (BLEN<1:0>) control bits in the DCISTAT SFR (see Register 21-4). The size of the transmit and receive buffers may be varied from 1 to 4 data words, using the BLEN control bits. The BLEN control bits are compared to the current value of the DCI buffer control unit address counter. When the 2 LS bits of the DCI address counter match the BLEN<1:0> value, the buffer control unit will be reset to 0. In addition, the contents of the receive shadow registers are transferred to the receive buffer registers and the contents of the transmit buffer registers are transferred to the transmit shadow registers.

21.3.15 BUFFER ALIGNMENT WITH DATA FRAMES

There is no direct coupling between the position of the AGU address pointer and the data frame boundaries. This means that there will be an implied assignment of each transmit and receive buffer that is a function of the BLEN control bits and the number of enabled data slots via the TSE and RSE control bits.

As an example, assume that a 4-word data frame is chosen and that we want to transmit on all four time-slots in the frame. This configuration would be established by setting the TSE0, TSE1, TSE2, and TSE3 control bits in the TSCON SFR. With this module setup, the TXBUF0 register would be naturally assigned to slot #0, the TXBUF1 register would be naturally assigned to slot #1, and so on.

Note:	When more than four time-slots are active within a data frame, the user code must keep track of which time-slots are to be read/written at each interrupt. In some cases, the alignment between transmit/ receive buffers and their respective slot assignments could be lost. Examples of such cases include an emulation breakpoint or a hardware trap. In these situations, the user should poll the SLOT status bits to determine what data should be loaded into the buffer registers to resynchronize the software with the DCI module.

21.3.16 TRANSMIT STATUS BITS

There are two transmit status bits in the DCISTAT SFR.

The TMPTY bit is set when the contents of the transmit buffer registers are transferred to the transmit shadow registers. The TMPTY bit may be polled in software to determine when the transmit buffer registers may be written. The TMPTY bit is cleared automatically by the hardware when a write to one of the four transmit buffers occurs.

The TUNF bit is read only and indicates that a transmit underflow has occurred for at least one of the transmit buffer registers that is in use. The TUNF bit is set at the time the transmit buffer registers are transferred to the transmit shadow registers. The TUNF status bit is cleared automatically when the buffer register that underflowed is written by the CPU.

Note: The transmit status bits only indicate status for buffer locations that are used by the module. If the buffer length is set to less than four words, for example, the unused buffer locations will not affect the transmit status bits.

21.3.17 RECEIVE STATUS BITS

There are two receive status bits in the DCISTAT SFR.

The RFUL status bit is read only and indicates that new data is available in the receive buffers. The RFUL bit is cleared automatically when all receive buffers in use have been read by the CPU.

The ROV status bit is read only and indicates that a receive overflow has occurred for at least one of the receive buffer locations. A receive overflow occurs when the buffer location is not read by the CPU before new data is transferred from the shadow registers. The ROV status bit is cleared automatically when the buffer register that caused the overflow is read by the CPU.

When a receive overflow occurs for a specific buffer location, the old contents of the buffer are overwritten.

Note: The receive status bits only indicate status for buffer locations that are used by the module. If the buffer length is set to less than four-words, for example, the unused buffer locations will not affect the transmit status bits.

21.3.18 SLOT STATUS BITS

The SLOT<3:0> status bits in the DCISTAT SFR indicate the current active time-slot. These bits will correspond to the value of the frame sync generator counter. The user may poll these status bits in software when a DCI interrupt occurs, to determine what time-slot data was last received and which time-slot data should be loaded into the TXBUF registers.

21.3.19 CSDO MODE BIT

The CSDOM control bit controls the behavior of the CSDO pin during unused transmit slots. A given transmit time-slot is unused if its corresponding TSEx bit in the TSCON SFR is cleared.

If the CSDOM bit is cleared (default), the CSDO pin will drive 0’s onto the CSDO pin during unused time-slot periods. This mode will be used when there are only two devices attached to the serial bus.

If the CSDOM bit is set, the CSDO pin will be tri-stated during unused time-slot periods. This mode allows multiple devices to share the same CSDO line in a Multi-Channel application. Each device on the CSDO line is configured so that it will only transmit data during specific time-slots. No two devices will transmit data during the same time-slot.

21.3.20 DIGITAL LOOPBACK MODE

Digital Loopback mode is enabled by setting the DLOOP control bit in the DCISTAT SFR. When the DLOOP bit is set, the module internally connects the SDO signal to CSDI. The actual data input on the CSDI I/O pin will be ignored in Digital Loopback mode.

21.3.21 UNDERFLOW MODE CONTROL BIT

When an underflow occurs, one of two actions may occur, depending on the state of the Underflow mode (UNFM) control bit in the DCICON2 SFR. If the UNFM bit is cleared (default), the module will transmit 0’s on the SDO pin during the active time-slot for the buffer location. In this operating mode, the Codec device attached to the DCI module will simply be fed digital ‘silence’. If the UNFM control bit is set, the module will transmit the last data written to the buffer location. This operating mode permits the user to send continuous data to the Codec device without consuming CPU overhead.

21.4 DCI Module interrupts

The frequency of DCI module interrupts is dependent on the BLEN<1:0> control bits in the DCICON2 SFR. An interrupt to the CPU is generated each time the set buffer length has been reached and a shadow register transfer takes place. A shadow register transfer is defined as the time when the previously written TXBUF values are transferred to the transmit shadow registers and new received values in the receive shadow registers are transferred into the RXBUF registers.

21.5 DCI Module Operation During CPU SLEEP and IDLE Modes

21.5.1 DCI MODULE OPERATION DURING CPU SLEEP MODE

The DCI module has the ability to operate while in SLEEP mode and wake the CPU when the CSCK signal is supplied by an external device (CSCKD = 1). The DCI module will generate an asynchronous interrupt when a DCI buffer transfer has completed and the CPU is in SLEEP mode.

21.5.2 DCI MODULE OPERATION DURING CPU IDLE MODE

If the DCISIDL control bit is cleared (default), the module will continue to operate normally even in IDLE mode. If the DCISIDL bit is set, the module will halt when IDLE mode is asserted.

21.6 Multi-Channel Mode Operation

The module is enabled for multi-channel operation by writing a value of 0 to the COFSM<1:0> control bits in the DCICON1 SFR. The Multi-Channel mode is used for both single and multi-channel transfers. The setup values for Multi-Channel mode are summarized in Table 21-2.

TABLE 21-2: MULTI-CHANNEL SETUP VALUES

COFSM<1:0>	`00`
BCG<9:0>	Select desired bit clock frequency (CSCKD = `0` only)
UNFM	Select behavior for a transmit underflow
CMOD	Select single or continuous transfer operation (COFSD = `0` only)
COFSD	`0` for master COFS `1` for slave COFS
CSCKE	`0`
CSCKD	`0` for master CSCK `1` for slave CSCK
DJST	`0`
WS<3:0>	Set for desired word length
COFSG<3:0>	Set for desired frame sync interval
BLEN<1:0>	Set for # data words to buffer between interrupts
TSE<15:0>	Select active transmit slots
RSE<15:0>	Select active receive slots

21.6.1 SETUP DETAILS FOR SINGLE CHANNEL CODEC

This section addresses the setup requirements for a single channel Codec. The setup example that follows assumes a single channel 16-bit Codec that requires a 256 x f_S CSCK frequency.

Most Codecs require a CSCK signal that is some multiple of the sampling frequency. In many cases, the CSCK signal has a frequency that is 256 x f_S. Therefore, a frame sync pulse must be generated every 256 CSCK cycles to start a data transfer.

In some cases, the CSCK signal is derived from a crystal oscillator on the Codec. In this case, the CSCKD control bit would be set to allow the DCI module to operate from external clock sources. In other applications, the CSCK signal can be provided to the Codec by the DCI module. In this case, a PLL may be present on the Codec to generate internal clocks from the CSCK input. The CSCKD bit would be cleared and the BCG control bits would be loaded with a value to produce the desired CSCK frequency.

To generate a 256-bit frame interval, the WS<3:0> control bits are set to a value of '`1111`' to provide a 16-bit data word size. Next, the COFSG<3:0> control bits are set to a value of '`1111`' to provide 16 data words per frame. The 256-bit frame could be set up to transfer sixteen, 16-bit data words between frame sync pulses. Because a single channel Codec is used, data is only transferred during the first 16 CSCK periods of the frame. Therefore, the RSE0 and TSE0 control bits are set to enable reception and transmission during the first data slot of the frame.

The amount of data to be buffered between interrupts must be decided. The BLEN<1:0> control bits must be set accordingly. One sample of data is received from the codec ADC for every sample that is transmitted to the codec DAC. For an application that requires minimal processing delay, the DCI module may be setup to provide an interrupt after each sample is transmitted/received. Optionally, up to four samples may be buffered before an interrupt is generated. The data samples can be stored and processed together to generate four new output samples. This process is known as 'block processing' and helps to reduce interrupt overhead.

The value written to the COFSD control bit depends on which device will initiate data transfers. The DCI module is the master device if the COFSD bit is cleared, and is the slave device if the bit is set. The DCI module may be a master or slave device depending on user preference and Codec functionality.

If the DCI module is the master device, data transfers do not have to be continuous. If the CMOD bit is cleared, data transfers will only take place during frame periods when the transmit buffers are written.

21.6.2 MULTI-CHANNEL CODEC CONFIGURATION

The transfer of control data and multiple data channels over the serial connection between the dsPIC device and the Codec is enabled by Time Division Multiplexing (TDM) of the data. The number of TDM time-slots will depend on the actual Codec that is selected.

A hypothetical example of serial time-slot assignment for a Codec is given below:

<u>Transmit Time-slots</u>

- Slot #0: Control Register Address
- Slot #1: Control Register Write Data
- Slot #2: DAC Channel 1 Data
- Slot #3: DAC Channel 2 Data

<u>Receive Time-slots</u>

- Slot #0: Control Register Address Echo
- Slot #1: Control Register Read Data
- Slot #2: ADC Channel 1 Data
- Slot #3: ADC Channel 2 Data

The DCI module is configured similar to the single channel setup described in Section 21.6.1, except that multiple slots are enabled for transmission and reception. In this example, TSE0, TSE1, TSE2, and TSE3 are set to transmit data on the first four slots of the frame. Similarly, RSE0, RSE1, RSE2, and RSE3 are set to enable reception during the first four time-slots.

The WS<3:0> control bits should be set to the native word size of the Codec and the COFSG<3:0> control bits should be set to provide the correct number of CSCK cycles per frame.

21.7 AC-Link Mode Operation

21.7.1 AC-LINK WORD SIZE ISSUES

The AC-Link protocol is a 256-bit frame with one 16-bit data slot, followed by twelve 20-bit data slots. The DCI module has two operating modes for the AC-Link protocol. These operating modes are selected by the COFSM<1:0> control bits in the DCICON1 SFR. The first AC-Link mode is called ‘16-bit AC-Link mode’ and is selected by setting COFSM<1:0> = `10`. The second AC-Link mode is called ‘20-bit AC-Link mode’ and is selected by setting COFSM<1:0> = `11`.

21.7.2 16-BIT AC-LINK MODE

In the 16-bit AC-Link mode, data word lengths are restricted to 16-bits. Note that this restriction only affects the 20-bit data time-slots of the AC-Link protocol. For received time-slots, the incoming data is simply truncated to 16-bits. For outgoing time-slots, the 4 LS bits of the data word are set to 0 by the module. This truncation of the time-slots limits the A/D and DAC data to 16-bits, but permits proper data alignment in the TXBUF and RXBUF registers. Each RXBUF and TXBUF register will contain one data time-slot value.

21.7.3 20-BIT AC-LINK MODE

The 20-bit AC-Link mode allows all bits in the data time-slots to be transmitted and received, but does not maintain data alignment in the TXBUF and RXBUF registers.

The 20-bit AC-Link mode functions similar to the Multi-Channel mode of the DCI module, except for the duty cycle of the frame synchronization signal. The AC-Link frame synchronization signal should remain high for 16 CSCK cycles and should be low for the following 240 cycles.

The 20-bit mode treats each 256-bit AC-Link frame as sixteen, 16-bit time-slots. In the 20-bit AC-Link mode, the module operates as if COFSG<3:0> = `1111` and WS<3:0> = `1111`. The data alignment for 20-bit data slots is ignored. For example, an entire AC-link data frame can be transmitted and received in a packed fashion by setting all bits in the TSCON and RSCON SFRs. Since the total available buffer length is 64-bits, it would take 4 consecutive interrupts to transfer the AC-Link frame. The application software must keep track of the current AC-Link frame segment.

21.7.4 AC-LINK SETUP

The module is enabled for AC-Link mode by writing `10` or `11` to the COFSM<1:0> control bits in the DCICON1 SFR. The word size selection bits (WS<3:0>) and the frame synchronization generator bits (COFS<4:0>) have no effect in the AC-Link modes, since the frame and word sizes are set by the protocol.

The CSCKD control bit is set in software. The COFSD control bit is cleared because the DCI will generate the COFS signal from the incoming CSCK signal. The CSCKE bit is cleared so that data is sampled on the rising edge.

The user must decide which time-slots in the AC-Link data frame are to be buffered, and set the TSE and RSE control bits accordingly. At a minimum, it will be necessary to buffer the transmit and receive Tag slots, which implies that the TSE0 and RSE0 bits should be set in software.

Note:	Only the TSE<12:0> and RSE12:0> bits will have an effect in the 16-bit AC-Link mode since an AC-Link frame has 13 time-slots.

21.8 I^{2}S Mode Operation

The DCI module is configured for I^{2}S mode by writing a value of `01` to the COFSM<1:0> control bits in the DCICON1 SFR. When operating in the I^{2}S mode, the DCI module will generate frame synchronization signals with a 50% duty cycle. Each edge of the frame synchronization signal marks the boundary of a new data word transfer.

The user must also select the frame length and data word size using the COFSG and WS control bits in the DCICON2 SFR.

21.8.1 I^{2}S FRAME AND DATA WORD LENGTH SELECTION

The WS and COFSG control bits are set to produce the period for one half of an I^{2}S data frame. That is, the frame length is the total number of CSCK cycles required for a left or a right data word transfer.

Although only one data word is transferred per I^{2}S subframe, It may be necessary to set the COFSG value to a higher value to produce the appropriate number of CSCK cycles per frame. For example, many I^{2}S Codecs use a 64x serial clock. In other words, there are 64 CSCK cycles per frame and 32 cycles per left/right subframe. Assuming that a 16-bit Codec is connected to the DCI module, the user should configure the COFSG bits for 2 words per frame (COFSG<3:0> = `0001`) and the WS bits are configured for a 16-bit data word (WS<3:0> = `1111`).

A 4 data word frame has been defined for this example, but only the first time-slot is of interest. Therefore, the TSE0 and RSE0 control bits are set to make the module actively transmit and receive data during these time-slots.

The BLEN bits must be set for the desired buffer length. Setting BLEN<1:0> = `01` will produce a CPU interrupt, once per I^{2}S frame.

21.8.2 I^{2}S DATA JUSTIFICATION

As per the I^{2}S specification, a data word transfer will, by default, begin one CSCK cycle after a transition of the WS signal. A ‘MS bit left justified’ option can be selected using the DJST control bit in the DCICON2 SFR.

If DJST = 1, the I^{2}S data transfers will be MS bit left justified. The MS bit of the data word will be presented on the SDO pin during the same CSCK cycle as the rising or falling edge of the COFS signal. The SDO pin is tristated after the data word has been sent.

The left justified data option allows two stereo Codecs to be connected to the same serial bus. The word size selection bits are set to twice the Codec word length and data is read/written to the DCI memory in a packed format.

Timing diagrams for I^{2}S mode are shown in Figure 21-5. For reference, these diagrams assume an 8-bit word size (WS<3:0> = `0111`). Two elements would be required to achieve a 16-bit sub-frame (COFSG<3:0> = `0001`). The third timing diagram in Figure 21-5 uses packed data to read/write from two Codecs. For this example, the DCI module is configured for a 16-bit data word (WS<3:0> = `1111`). Two packed 8-bit words are written to each 16-bit location in the DCI memory buffer.

FIGURE 21-5: I²S TIMING DIAGRAMS

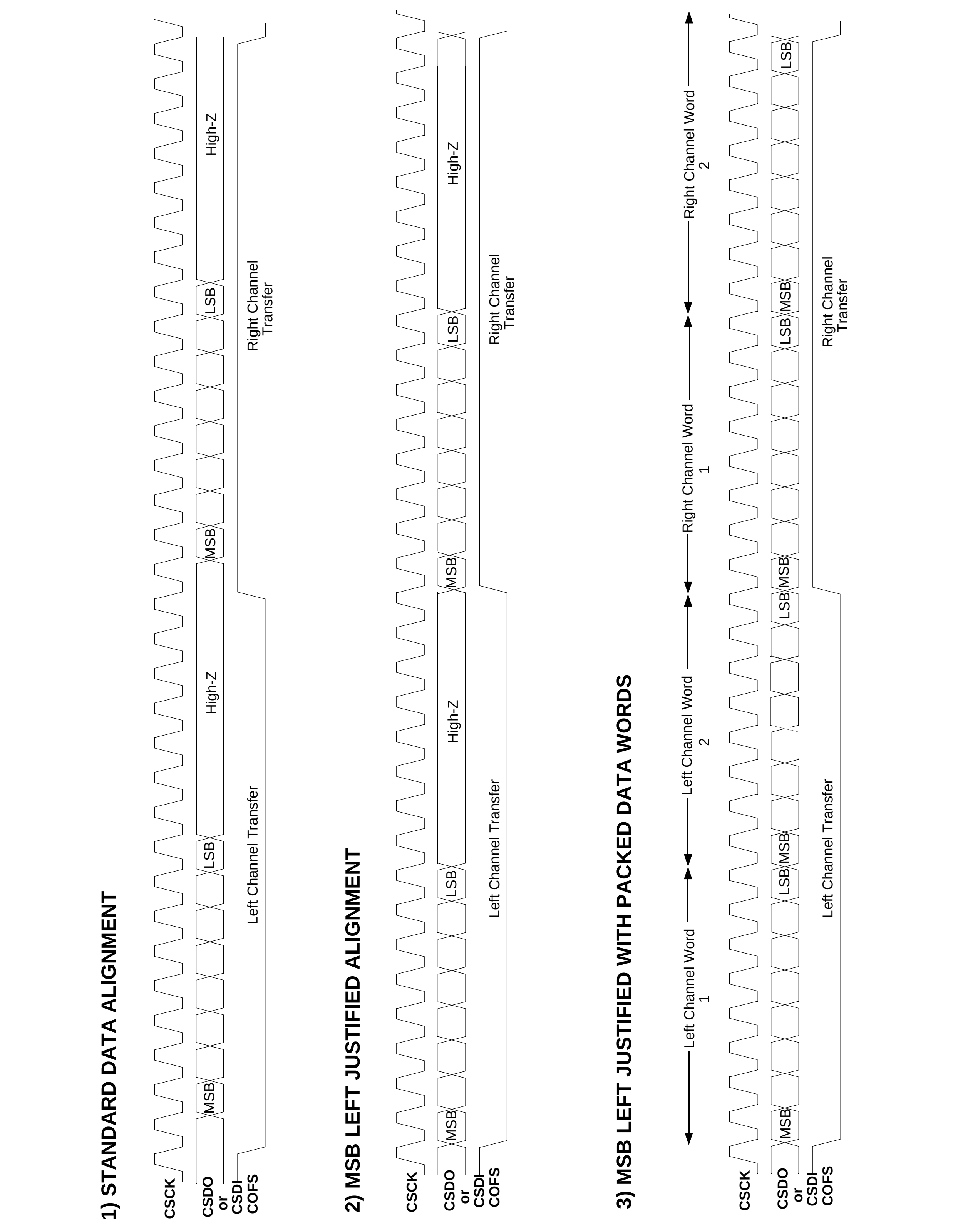


TABLE 21-3: DCI REGISTER MAP

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
DCICON1	0240	DCIEN	—	DCISIDL	—	DLOOP	CSCKD	CSCKE	COFSD	UNFM	CSDOM	DJST	—	—	—	COFSM1	COFSM0	0u0u 0000 000u uu00
DCICON2	0242	—	—	—	—	BLEN1	BLEN0	—	COFSG<3:0>				—	WS<3:0>				uuuu 00u0 000u 0000
DCICON3	0244	—	—	—	—	DCI Bit Clock Generator Control Register												uuuu 0000 0000 0000
DCISTAT	0246	—	—	—	—	SLOT3	SLOT2	SLOT1	SLOT0	—	—	—	—	ROV	RFUL	TUNF	TMPTY	uuuu 0000 uuuu 0000
TSCON	0248	TSE15	TSE14	TSE13	TSE12	TSE11	TSE10	TSE9	TSE8	TSE7	TSE6	TSE5	TSE4	TSE3	TSE2	TSE1	TSE0	0000 0000 0000 0000
RSCON	024C	RSE15	RSE14	RSE13	RSE12	RSE11	RSE10	RSE9	RSE8	RSE7	RSE6	RSE5	RSE4	RSE3	RSE2	RSE1	RSE0	0000 0000 0000 0000
RXBUF0	0250	Receive Buffer #0 Data Register																0000 0000 0000 0000
RXBUF1	0252	Receive Buffer #1 Data Register																0000 0000 0000 0000
RXBUF2	0254	Receive Buffer #2 Data Register																0000 0000 0000 0000
RXBUF3	0256	Receive Buffer #3 Data Register																0000 0000 0000 0000
TXBUF0	0258	Transmit Buffer #0 Data Register																0000 0000 0000 0000
TXBUF1	025A	Transmit Buffer #1 Data Register																0000 0000 0000 0000
TXBUF2	025C	Transmit Buffer #2 Data Register																0000 0000 0000 0000
TXBUF3	025E	Transmit Buffer #3 Data Register																0000 0000 0000 0000

REGISTER 21-1: DCICON1: DCI MODULE CONTROL REGISTER1

Upper Half:

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
DCIEN	—	DCISIDL	—	DLOOP	CSCKD	CSCKE	COFSD
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
UNFM	DJST	CSDOM	—	—	—	COFSM1	COFSM0
bit 7							bit 0

bit 15 **DCIEN:** DCI Module Enable bit
1 = Module is enabled
0 = Module is disabled

bit 14 **Unimplemented:** Read as ‘0’

bit 13 **DCISIDL:** DCI Stop in IDLE Control bit
1 = Module will halt in CPU IDLE mode
0 = Module will continue to operate in CPU IDLE mode

bit 12 **Unimplemented:** Read as ‘0’

bit 11 **DLOOP:** Digital Loopback Mode Control bit
1 = Digital Loopback mode is enabled
0 = Digital Loopback mode is disabled

bit 10 **CSCKD:** Sample Clock Direction Control bit
1 = CSCK pin is an input when DCI module is enabled
0 = CSCK pin is an output when DCI module is enabled

bit 9 **CSCKE:** Sample Clock Edge Control bit
1 = Data changes on CSCK falling edge, sampled on CSCK rising edge
0 = Data changes on CSCK rising edge, sampled on CSCK falling edge

bit 8 **COFSD:** Output Frame Synchronization Direction Control bit
1 = COFS pin is an input when DCI module is enabled
0 = COFS pin is an output when DCI module is enabled

bit 7 **UNFM:** Underflow Mode bit
1 = Transmit last value written to the transmit buffers on a transmit underflow
0 = Transmit 0’s on a transmit underflow

bit 6 **CSDOM:** Serial Data Output Mode bit
1 = SDO pin will be tri-stated during disabled transmit time-slots
0 = SDO pin drives 0’s during disabled transmit time-slots

bit 5 **DJST:** DCI Data Justification Control bit
1 = Data transmission/reception is begun during the same CSCK cycle as the frame synchronization pulse
0 = Data transmission/reception is begun one CSCK cycle after frame synchronization pulse

bit 4-2 **Unimplemented:** Read as ‘0’

bit 1-0 **COFSM<1:0>:** Frame Sync Mode bits
11 = 20-bit AC-Link mode
10 = 16-bit AC-Link mode
01 = I^{2}S Frame Sync mode
00 = Multi-Channel Frame Sync mode

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

REGISTER 21-2: DCICON2: DCI MODULE CONTROL REGISTER2

Upper Half:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	U-0	R/W-0
—	—	—	—	BLEN1	BLEN0	—	COFSG3
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
COFSG2	COFSG1	COFSG0	—	WS2	WS1	WS0	WS0
bit 7							bit 0

bit 15-12 **Unimplemented:** Read as ‘0’

bit 11-10 **BLEN<1:0>:** Buffer Length Control bits
`11` = Four data words will be buffered between interrupts
`10` = Three data words will be buffered between interrupts
`01` = Two data words will be buffered between interrupts
`00` = One data word will be buffered between interrupts

bit 9 **Unimplemented:** Read as ‘0’

bit 8-5 **COFSG<3:0>:** Frame Sync Generator Control bits
`1111` = Data frame has 16 words
`||`
`0010` = Data frame has 3 words
`0001` = Data frame has 2 words
`0000` = Data frame has 1 word

bit 4 **Unimplemented:** Read as ‘0’

bit 3-0 **WS<3:0>:** DCI Data Word Size bits
`1111` = Data word size is 16-bits
`||`
`0010` = Data word size is 3-bits
`0001` = Data word size is 2-bits
`0000` = Data word size is 1-bit

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

REGISTER 21-3: DCICON3: DCI MODULE CONTROL REGISTER3

Upper Half:							
U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	BCG11	BCG10	BCG9	BCG8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BCG7	BCG6	BCG5	BCG4	BCG3	BCG2	BCG1	BCG0
bit 7							bit 0

bit 15-12 **Unimplemented:** Read as ‘0’

bit 11-0 **BCG<11:0>:** DCI bit Clock Generator Control bits

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

REGISTER 21-4: DCISTAT: DCI STATUS REGISTER

Upper Half:							
U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	SLOT3	SLOT2	SLOT1	SLOT0
bit 15							bit 8

Lower Half:							
U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	ROV	RFUL	TUNF	TMPTY
bit 7							bit 0

bit 15-12 **Unimplemented:** Read as ‘0’

bit 11-8 **SLOT<3:0>:** DCI Slot Status bits
`1111` = Slot #15 is currently active
| |
`0010` = Slot #2 is currently active
`0001` = Slot #1 is currently active
`0000` = Slot #0 is currently active

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3 **ROV:** Receive Overflow Status bit
`1` = A receive overflow has occurred for at least one buffer location
`0` = A receive overflow has not occurred

bit 2 **RFUL:** Receive Buffer Full Status bit
`1` = New data is available in the receive buffers
`0` = The receive buffer is empty

bit 1 **TUNF:** Transmit Buffer Underflow Status bit
`1` = A transmit underflow has occurred for at least one transmit buffer
`0` = A transmit underflow has not occurred

bit 0 **TMPTY:** Transmit Buffer Empty Status bit
`1` = The transmit buffer is empty
`0` = The transmit buffer is not empty

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

REGISTER 21-5: TSCON: TRANSMIT SLOT CONTROL REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TSE15	TSE14	TSE13	TSE12	TSE11	TSE10	TSE9	TSE8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TSE7	TSE6	TSE5	TSE4	TSE3	TSE2	TSE1	TSE0
bit 7							bit 0

bit 11 **TSE<15:0>:** Transmit Slot Enable Control bits
1 = Transmit buffer contents are sent during the time-slot
0 = SDO pin is tri-stated during the time-slot

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

REGISTER 21-6: RSCON: RECEIVE SLOT CONTROL REGISTER

Upper Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RSE15	RSE14	RSE13	RSE12	RSE11	RSE10	RSE9	RSE8
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RSE7	RSE6	RSE5	RSE4	RSE3	RSE2	RSE1	RSE0
bit 7							bit 0

bit 11 **RSE<15:0> :** Receive Slot Enable bits
1 = CSDI data is received during time-slot n
0 = CSDI data is ignored during time-slot n

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

NOTES:

22.0 12-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The 12-bit Analog-to-Digital Converter (A/D) allows conversion of an analog input signal to a corresponding 12-bit digital number. This module is based on a Successive Approximation Register (SAR) architecture, and provides a maximum sampling rate of 500 ksps. The A/D module has up to 16 analog inputs, which are multiplexed into a sample and hold amplifier. The output of the sample and hold is the input into the converter, which generates the result. The analog reference voltage is software selectable to either the device supply voltage (AVDD/AVSS), or the voltage level on the (VREF+/VREF-) pin. The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode, with RC oscillator selection.

The A/D module has six 16-bit registers.

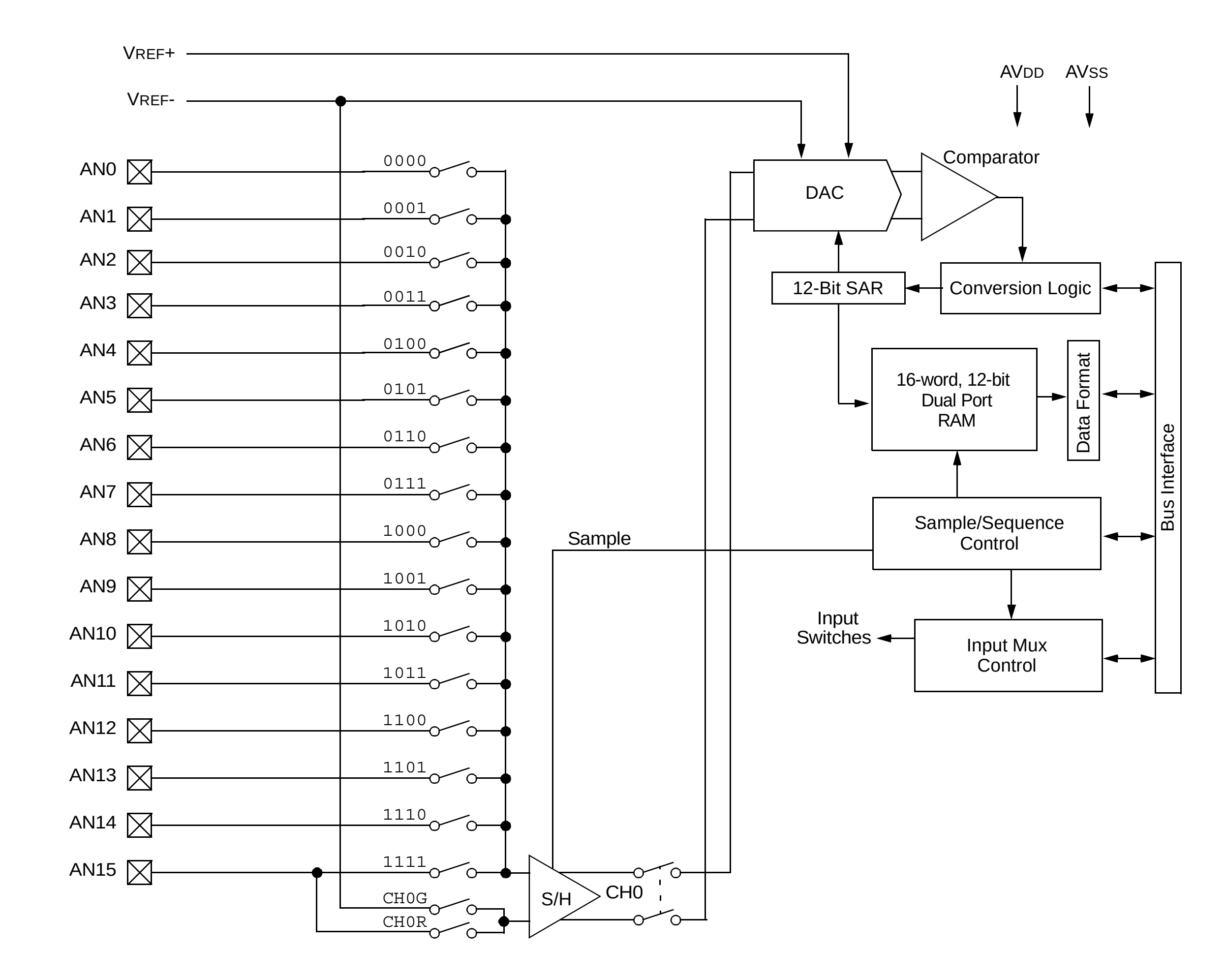
- A/D Control Register1 (ADCON1)
- A/D Control Register2 (ADCON2)
- A/D Control Register3 (ADCON3)
- A/D Input Select Register (ADCHS)
- A/D Port Configuration Register (ADPCFG)
- A/D Input Scan Selection Register (ADCSSL)

The ADCON1, ADCON2 and ADCON3 registers control the operation of the A/D 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, 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 12-bit A/D module is shown in Figure 22-1.

FIGURE 22-1: 12-BIT A/D FUNCTIONAL BLOCK DIAGRAM



22.1 A/D Result Buffer

The module contains a 16-word dual port RAM, called ADRES<15:0>, to buffer the A/D results. The RAM is 12-bits wide, but the data obtained is represented in one of four different 16-bit data formats.

Only word writes are allowed to the result buffer. If byte writes are attempted, the results are indeterminate.

22.2 Conversion Operation

After the A/D module has been configured as desired, the sampling is started by setting the SAMP bit. Various sources, such as a programmable bit, timer time-outs and external events, will terminate sampling and start a conversion. When the A/D conversion is completed, the result is loaded into ADRES<15:0>, the CONV bit is cleared, and the A/D interrupt flag, ADIF, is set. The ADC module can be configured for different interrupt rates, as described in Section 22.3.

The following steps should be followed for doing an A/D conversion:

1. Configure the A/D 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 A/D module
2. Configure A/D interrupt (if required)
   - Clear ADIF bit
   - Select A/D interrupt priority
3. Start sampling
4. Wait the required sampling time
5. Trigger sample end, start conversion
   - Module sets CONV bit
6. Wait for A/D conversion to complete, by either:
   - Polling for the CONV bit to be cleared
   - Waiting for the A/D interrupt
7. Read A/D result buffer, clear ADIF if required.

22.3 Selecting the Conversion Sequence

Several groups of control bits select the sequence that the A/D connects inputs to the sample/hold channel, converts a channel, writes the buffer memory and generates interrupts.

The sequence is controlled by the sampling clocks.

The SMPI bits will select how many sample clocks occur before an interrupt occurs. This can vary from 1 sample per interrupt, to 16 samples per interrupt.

The BUFM bit will split the 16-word results buffer into (2) 8-word groups. Writing to the 8-word buffers will be alternated on each interrupt event.

Use of the BUFM bit will depend on how much time is available for the moving of the buffers after the interrupt.

If the processor can quickly unload a full buffer within the time it takes to sample and convert one channel, the BUFM bit can be 0 and up to 16 conversions (corresponding to the 16 input channels) may be done per interrupt. The processor will have one sample and conversion time to move the sixteen conversions.

If the processor cannot unload the buffer within the sample and conversion time, the BUFM bit should be 1. For example, if SMPI<3:0> (ADCON2<5:2>) = `0111`, then eight conversions will be loaded into 1/2 of the buffer, following which, an interrupt occurs. The next eight conversions will be loaded into the other 1/2 of the buffer. The processor will have 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. If the ALTS bit is 0, only the SAMPLE A inputs are selected for sampling. If the ALTS bit is 1 and SMPI<3:0> = `0000`, on the first sample/convert sequence, the SAMPLE A inputs are selected, and on the next sample/convert sequence, the SAMPLE B inputs are selected.

The CSCNA bit (ADCON2<10>) will allow the CH0 channel inputs to be scanned across a selected number of analog inputs during SAMPLE A samples. 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 occurs. If the number of inputs selected is greater than the number of samples taken per interrupt, the higher numbered inputs are unused.

22.4 Programming the Sample Trigger

The sample trigger will terminate sampling and start the requested conversions.

The SSRC<2:0> bits select the source of the sample trigger.

When SSRC<2:0> = `000`, the sample trigger is under software control. Clearing the SAMP bit will cause the sample trigger.

When SSRC<2:0> = `111`, the sample trigger is under A/D clock control. The SAMC bits select the number of A/D clocks between the start of sampling 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 or external interrupts.

The SSRC bits provide for up to 6 alternate sources of sample trigger.

22.5 Aborting a Conversion

Clearing the CONV bit during a conversion will abort the current conversion and stop the sampling sequencing until the next sampling trigger. The ADRES will not be updated with the partially completed A/D conversion sample. That is, the ADRES will continue to contain the value of the last completed conversion (or the last value written to the ADRES register).

If the clearing of the CONV bit coincides with an auto start, the clearing has a higher priority, and a new conversion will not start.

After the A/D conversion is aborted, a 2TAD wait is required before the next sampling may be started by setting the SAMP bit.

22.6 Selecting the A/D Conversion Clock

The A/D conversion requires 13 TAD. The source of the A/D conversion clock is software selected, using a six-bit counter. There are 64 possible options for TAD.

TAD = TCY * (0.5*(ADCS<5:0> +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 350 nsec. Table 22-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

TABLE 22-1: TYPICAL TAD VS. DEVICE OPERATING FREQUENCIES

AD Clock Interval (TAD)/TCONV							
AD Clock Source Select			**Device TCY/Device MIPS/TQ/FOSC**				
Clock	**ADRC**	**ADCS<5:0>**	**33.33 nsec/ 30 MIPS/ 8.33 nsec/ 120 MHz**	**40 nsec/ 25 MIPS/ 10 nsec/ 100 MHz**	**80 nsec/ 12.5 MIPS/ 20 nsec/ 50 MHz**	**160 nsec/ 6.25 MIPS/ 40 nsec/ 25 MHz**	**1000 nsec/ 1 MIPS/ 250 nsec/ 4 MHz**
2 TQ	0	000000	16.67 ns[2]/ 0.22 µs	20 ns[2]/ 0.26 µs	40 ns[2]/ 0.52 µs	80 ns[2]/ 1.04 µs	500 ns/ 6.5 µs
4 TQ	0	000001	33.33 ns[2]/ 0.44 µs	40 ns[2]/ 0.52 µs	80 ns[2]/ 1.04 µs	160 ns/ 2.08 µs	1.0 µs/ 13 µs
8 TQ	0	000011	66.66 ns[2]/ 0.88 µs	80 ns[2]/ 1.04 µs	160 ns/ 2.08 µs	320 ns/ 4.16 µs	2.0 µs[3]/ 26 µs
16 TQ	0	000111	133.32 ns[2]/ 1.76 µs	160 ns/ 2.08 µs	320 ns/ 4.16 µs	640 ns[3]/ 8.32 µs	4.0 µs[3]/ 52 µs
32 TQ	0	001111	266.64 ns/ 3.52 µs	320 ns/ 4.16 µs	640 ns[3]/ 8.32 µs	1.28 µs[3]	8.0 µs[3]/ 104 µs
64 TQ	0	011111	533.28 ns[3]/ 7.04 µs	640 ns[3]/ 8.32 µs	1.28 µs[3]	2.56 µs[3]	16.0 µs[3]/ 208 µs
RC	1	xxxxxx	200 - 400 ns/ 3.9 µs[1,4]	200 - 400 ns/ 3.9 µs[1,4]	200 - 400 ns / 3.9 µs[1,4]	200 - 400 ns / 3.9 µs[1,4]	200 - 400 ns/ 3.9 µs[1]

Note 1: The RC source has a typical TAD time of 300 ns for VDD > 3.0V.
2: These values violate the minimum required TAD time.
3: For faster conversion times, the selection of another clock source is recommended.
4: A/D cannot meet full accuracy with RC clock source and FOSC > 20 MHz.

22.7 A/D Sampling Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 22-2. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), see Figure 22-2. The impedance for analog sources must be small enough to meet accuracy requirements at the given speed. After the analog input channel is selected (changed), this sampling must be done before the conversion can be started.

To calculate the minimum sampling time, Equation 22-1 may be used. This equation assumes that the input is stepped some multiple (n) of the LSB step size and the output must be captured to within 1/2 LSb error (8192 steps for 12-bit A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

The CHOLD is assumed to be 18 pF for the A/D.

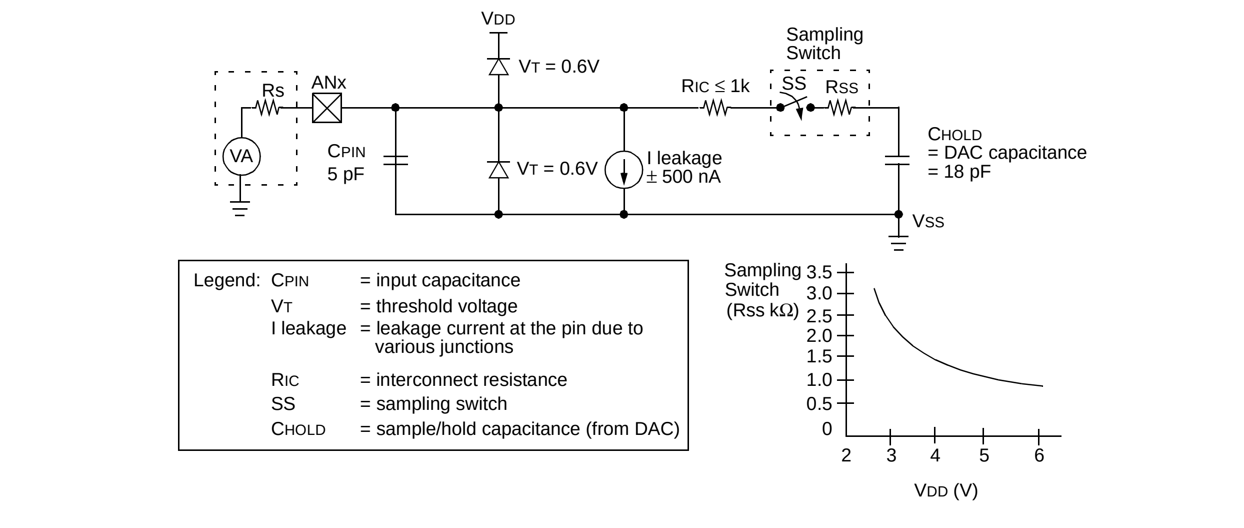
EQUATION 22-1: A/D SAMPLING TIME EQUATIONS

$$
\begin{aligned}
\Delta V_O &= \Delta V_I \bullet (1 - e^{(-T_C/C_{HOLD}\,(R_{IC}+R_{SS}+R_S))}) \\
1 - (\Delta V_O / \Delta V_I) &= e^{(-T_C/C_{HOLD}\,(R_{IC}+R_{SS}+R_S))} \\
\Delta V_I &= n \bullet LSB \\
\Delta V_O &= n \bullet LSB - 1/2\ LSB \\
\Delta V_O / \Delta V_I &= (n \bullet LSB - 1/2\ LSB) / n \bullet LSB \\
1 - (\Delta V_O / \Delta V_I) &= 1 / 2n \\
1 / 2n &= e^{(-T_C/C_{HOLD}\,(R_{IC}+R_{SS}+R_S))} \\
T_C &= C_{HOLD} \bullet (R_{IC}+R_{SS}+R_S) \bullet -\text{In}(1/2 \bullet n) \\
TSMP &= \text{Amplifier Settling Time} \\
&\quad + \text{Holding Capacitor Charging Time } (T_C) \\
&\quad + \text{Temperature Coefficient} \quad \dagger
\end{aligned}
$$

† The temperature coefficient is only required for temperatures > 25°C.

$$
\begin{aligned}
TSMP &= 0.5\ \text{ms} \\
&\quad + C_{HOLD} \bullet (R_{IC}+R_{SS}+R_S) \bullet -\text{In}(1/2 \bullet n) \\
&\quad + [(\text{Temp} - 25°\text{C})(0.05\ \mu\text{s}/°\text{C})]
\end{aligned}
$$

FIGURE 22-2: ANALOG INPUT MODEL



22.8 Module Power-down Modes

The module has 3 internal Power modes.

When the ADON bit is 1, the module is in Active mode and is fully powered and functional.

When ADON is 0 and ADSTBY is 1, the module is in Standby mode. In Standby mode, the digital portions of the module are active; however, the analog portions are powered down, including the bias generators.

When ADON is 0 and ADSTBY 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 Standby or Off mode, the user must wait for the bias generators to stabilize.

22.9 A/D Operation During CPU SLEEP and IDLE Modes

22.9.1 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 exiting from SLEEP mode.

Register contents are not affected by the device entering or leaving SLEEP mode.

The A/D 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 A/D module waits for 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 completed, the CONV bit will be cleared and the result loaded into the ADRES register.

If the A/D interrupt is enabled, the device will wake up from SLEEP. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set.

22.9.2 A/D OPERATION DURING CPU IDLE MODE

For the A/D, the ADSIDL bit selects if the module will stop on IDLE or continue on IDLE. If ADSIDL = 0, the module will continue operation on assertion of IDLE mode. If ADSIDL = 1, the module will stop on IDLE.

22.10 Effects of a RESET

A device RESET forces all registers to their RESET state. This forces the A/D module to be turned off, and any conversion and sampling sequence is aborted. The values that are in the ADRES registers are not modified. The A/D result register will contain unknown data after a Power-on Reset.

22.11 Output Formats

The A/D result is 12-bits wide. The data buffer RAM is also 12-bits wide. The 12-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 right justified (integer) format.

FIGURE 22-3: A/D OUTPUT DATA FORMATS

RAM Contents:					d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
Read to Bus:																
Signed Fractional	$\overline{d11}$	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0
Fractional	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0
Signed Integer	$\overline{d11}$	$\overline{d11}$	$\overline{d11}$	$\overline{d11}$	$\overline{d11}$	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
Integer	0	0	0	0	d11	d10	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

22.12 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 (V_{OH} or V_{OL}) will be converted.

The A/D operation is independent of the state of the CHS<3:0> bits and the TRIS bits.

When reading the PORT register, all pins configured as analog input channel will read as cleared (a low level).

Pins configured as digital inputs, will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.

Analog levels on any pin that is defined as a digital input (including the AN pins), may cause the input buffer to consume current that exceeds the device specifications.

22.13 Connection Considerations

Since the analog inputs employ ESD protection, they have diodes to V_{DD} and V_{SS}. This requires that the analog input must 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 22-2: 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												uuuu 0000 0000 0000
ADCBUF1	0282	—	—	—	—	ADC Data Buffer 1												uuuu 0000 0000 0000
ADCBUF2	0284	—	—	—	—	ADC Data Buffer 2												uuuu 0000 0000 0000
ADCBUF3	0286	—	—	—	—	ADC Data Buffer 3												uuuu 0000 0000 0000
ADCBUF4	0288	—	—	—	—	ADC Data Buffer 4												uuuu 0000 0000 0000
ADCBUF5	028A	—	—	—	—	ADC Data Buffer 5												uuuu 0000 0000 0000
ADCBUF6	028C	—	—	—	—	ADC Data Buffer 6												uuuu 0000 0000 0000
ADCBUF7	028E	—	—	—	—	ADC Data Buffer 7												uuuu 0000 0000 0000
ADCBUF8	0290	—	—	—	—	ADC Data Buffer 8												uuuu 0000 0000 0000
ADCBUF9	0292	—	—	—	—	ADC Data Buffer 9												uuuu 0000 0000 0000
ADCBUFA	0294	—	—	—	—	ADC Data Buffer 10												uuuu 0000 0000 0000
ADCBUFB	0296	—	—	—	—	ADC Data Buffer 11												uuuu 0000 0000 0000
ADCBUFC	0298	—	—	—	—	ADC Data Buffer 12												uuuu 0000 0000 0000
ADCBUFD	029A	—	—	—	—	ADC Data Buffer 13												uuuu 0000 0000 0000
ADCBUFE	029C	—	—	—	—	ADC Data Buffer 14												uuuu 0000 0000 0000
ADCBUFF	029E	—	—	—	—	ADC Data Buffer 15												uuuu 0000 0000 0000
ADCON1	02A0	ADON	—	ADSIDL	ADSTBY	—	—	FORM<1:0>		SSRC<2:0>			—	—	ASAM	SAMP	CONV	0u00 uu00 000u u000
ADCON2	02A2	VCFG<2:0>			OFFCAL	—	CSCNA	—	—	BUFS	—	SMPI<3:0>				BUFM	ALTS	0000 u0uu 0u00 0000
ADCON3	02A4	—	—	—	SAMC<4:0>					ADRC	—	ADCS<5:0>						uuu0 0000 0u00 0000
ADCHS	02A6	—	—	—	CH0NB	CH0SB<3:0>				—	—	—	CH0NA	CH0SA<3:0>				uuu0 0000 uuu0 0000
ADPCFG	02A8	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000
ADCSSL	02AA	CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8	CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0	0000 0000 0000 0000

REGISTER 22-1: ADCON1: A/D CONTROL REGISTER1

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
ADON	—	ADSIDL	ADSTBY	—	—	FORM1	FORM0
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0 HC, HS	R/C-0 HC, HS
SSRC2	SSRC1	SSRC0	—	—	ASAM	SAMP	CONV
bit 7							bit 0

bit 15 **ADON:** A/D Operating Mode bit
`1` = A/D converter module is operating
`0` = A/D converter is in Standby mode or off

bit 14 **Unimplemented:** Read as '0'

bit 13 **ADSIDL:** Stop in IDLE Mode bit
`1` = Discontinue module operation when device enters IDLE mode
`0` = Continue module operation in IDLE mode

bit 12 **ADSTBY:** A/D Standby Mode bit
`1` = If ADON = 0, A/D converter module in Standby mode. Analog circuits powered down.
`0` = If ADON = 0, A/D converter is shut-off and consumes no operating current

bit 11-10 **Unimplemented:** Read as '0'

bit 9-8 **FORM<1:0>:** Data Output Format bits
`11` = Signed Fractional (DOUT = `sddd dddd dddd 0000` where `s` = .NOT.d<11>)
`10` = Fractional (`DOUT = dddd dddd dddd 0000`)
`01` = Signed Integer (DOUT = `ssss sddd dddd dddd` where `s` = .NOT.d<11>)
`00` = Integer (DOUT = `0000 dddd dddd dddd`)

bit 7-5 **SSRC<2:0>:** Sample Clock Source Select bits
`111` = Internal counter ends sampling and starts conversion (auto convert)
`110` = Reserved
`101` = Reserved
`100` = Reserved
`011` = MPWM interval ends sampling and starts conversion
`010` = GP Timer compare ends sampling and starts conversion
`001` = Active transition on INT pin ends sampling and starts conversion
`000` = Clearing sample bit ends sampling and starts conversion

bit 4-3 **Unimplemented:** Read as '0'

bit 2 **ASAM:** A/D Sample Auto Start bit
`1` = Sampling begins immediately after last conversion. SAMP bit is auto set.
`0` = Sampling begins when SAMP bit set

bit 1 **SAMP:** A/D Sample Enable bit
`1` = A/D sample/hold amplifiers are sampling
`0` = A/D sample/hold amplifiers are holding

bit 0 **CONV:** A/D Conversion Status bit
`1` = A/D conversion cycle in progress. Cleared by hardware when A/D conversion complete.
`0` = A/D conversion completed/not in progress

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
C = Clearable bit	S = Settable bit	-n = Value at POR	HC = Hardware Clear
1 = bit is set	0 = bit is cleared	x = bit is unknown	HS = Hardware Set

REGISTER 22-2: ADCON2: A/D CONTROL REGISTER2

Upper Byte:

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0	U-0
VCFG2	VCFG1	VCFG0	OFFCAL	—	CSCNA	—	—
bit 15							bit 8

Lower Byte:

R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI3	SMPI2	SMPI1	SMPI0	BUFM	ALTS

bit 15-13 **VCFG<2:0>:** Voltage Reference Configuration bits

	A/D VREF+	**A/D VREF-**
000	AVDD	AVSS
001	External VREF+	AVSS
010	AVDD	External VREF-
011	External VREF+	External VREF-
100	Internal VRH	Internal VRL
101	Internal VRL	AVSS
101	Internal VRH	AVSS
111	AVDD	Internal VRL
Note:	Shaded regions in the table are unused. If these options are selected, the A/D references will default to AVDD and AVSS.	

bit 12 **OFFCAL:** Selects Offset Calibration Mode bit
1 = + and - inputs of channel sample/hold shorted together
0 = + and - inputs of channel sample/hold normal

bit 11 **Unimplemented:** Read as '0'

bit 10 **CSCNA:** Scan Input Selections for CH0+ bit during SAMPLE A
1 = Scan inputs
0 = Do not scan inputs

bit 9-8 **Unimplemented:** Read as '0'

bit 7 **BUFS:** Buffer Fill Status bit (only valid when BUFM = 1)
1 = A/D is currently filling buffer 0x8 - 0xF, user should access data in 0x0 - 0x7
0 = A/D is currently filling buffer 0x0 - 0x7, user should access data in 0x8 - 0xF

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

bit 5-2 **SMPI<3:0>:** Selects Number of Samples per Interrupt bits
1111 = Interrupts at the completion of conversion for each 16th sample
1110 = Interrupts at the completion of conversion for each 15th sample
. . . . . .
0001 = Interrupts at the completion of conversion for each 2nd sample
0000 = Interrupts at the completion of conversion for each sample

bit 1 **BUFM:** Buffer Fill Mode Select bit
1 = Starts buffer filling at address 0x0 on first interrupt and 0x8 on next interrupt
0 = Always starts filling buffer at address 0x0

bit 0 **ALTS:** Alternate Input Sample Mode Select bit
1 = Uses channel input selects for SAMPLE A on first sample and SAMPLE B on next samples
0 = Always uses channel input selects for SAMPLE A

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

REGISTER 22-3: ADCON3: A/D CONTROL REGISTER3

Upper Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0
bit 15							bit 8

Lower Byte:							
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	—	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as '0'

bit 12-8 **SAMC<4:0>:** Auto Sample Time bits
`11111` = 31 T_{AD}
· · · · ·
`00001` = 1 T_{AD}
`00000` = 0 T_{AD}

bit 7 **ADRC:** A/D Conversion Clock Source bit
`1` = A/D internal RC clock
`0` = Clock derived from system clock

bit 6 **Unimplemented:** Read as ’0’

bit 5-0 **ADCS<5:0>:** A/D Conversion Clock Select bits
`111111` = $T_Q \cdot 2 \cdot$ (ADCS<5:0> +1) = $128 \cdot T_Q$
· · · · · ·
`000001` = $T_Q \cdot 2 \cdot$ (ADCS<5:0> +1) = $4 \cdot T_Q$
`000000` = $T_Q \cdot 2 \cdot$ (ADCS<5:0> +1) = $2 \cdot T_Q$

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

REGISTER 22-4: ADCHS: A/D INPUT SELECT REGISTER

Upper Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CH0NB	CH0SB3	CH0SB2	CH0SB1	CH0SB0
bit 15							bit 8

Lower Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	CH0NA	CH0SA3	CH0SA2	CH0SA1	CH0SA0
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as '0'

bit 12 **CH0NB:** Channel 0 Negative Input Select bit for SAMPLE B
Same definition as bit <4>

bit 11-8 **CH0SB<3:0>:** Channel 0 Positive Input Select bit for SAMPLE B
Same definition as bits <3:0>

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **CH0NA:** Channel 0 Negative Input Select bit for SAMPLE A
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VREF-

bit 3-0 **CH0SA<3:0>:** Channel 0 Positive Input Select bit for SAMPLE A
1111 = Channel 0 positive input is AN15
1110 = Channel 0 positive input is AN14
1101 = Channel 0 positive input is AN13
1100 = Channel 0 positive input is AN12
1011 = Channel 0 positive input is AN11
1010 = Channel 0 positive input is AN10
1001 = Channel 0 positive input is AN9
1000 = Channel 0 positive input is AN8
0111 = Channel 0 positive input is AN7
0110 = Channel 0 positive input is AN6
0101 = Channel 0 positive input is AN5
0100 = Channel 0 positive input is AN4
0011 = Channel 0 positive input is AN3
0010 = Channel 0 positive input is AN2
0001 = Channel 0 positive input is AN1
0000 = Channel 0 positive input is AN0

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

REGISTER 22-5: ADPCFG: A/D PORT CONFIGURATION REGISTER

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 15-0 **PCFG<15:0>:** A/D Port Configuration Control bits
1 = Port pin in Digital mode, port read input enabled, A/D input multiplexor connected to AV_{SS}
0 = Port pin in Analog mode, port read input disabled, A/D samples pin voltage

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

REGISTER 22-6: ADCSSL: A/D INPUT SCAN SELECT REGISTER

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
bit 7							bit 0

bit 15-0 **CSSL<15:0>:** A/D Input Scan Selection bits
1 = Select ANx for input scan
0 = Skip ANx for input scan

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

23.0 10-BIT HIGH SPEED ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The10-bit high-speed analog-to-digital converter (A/D) allows conversion of an analog input signal to a corresponding 10-bit digital number. This module is based on a Successive Approximation Register (SAR) architecture, and provides a maximum sampling rate of 500 ksps. The A/D module has up to 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 voltage is software selectable to either the device supply voltage (AVDD/AVSS) or the voltage level on the (VREF+/VREF-) pin. The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode.

The A/D module has six 16-bit registers.

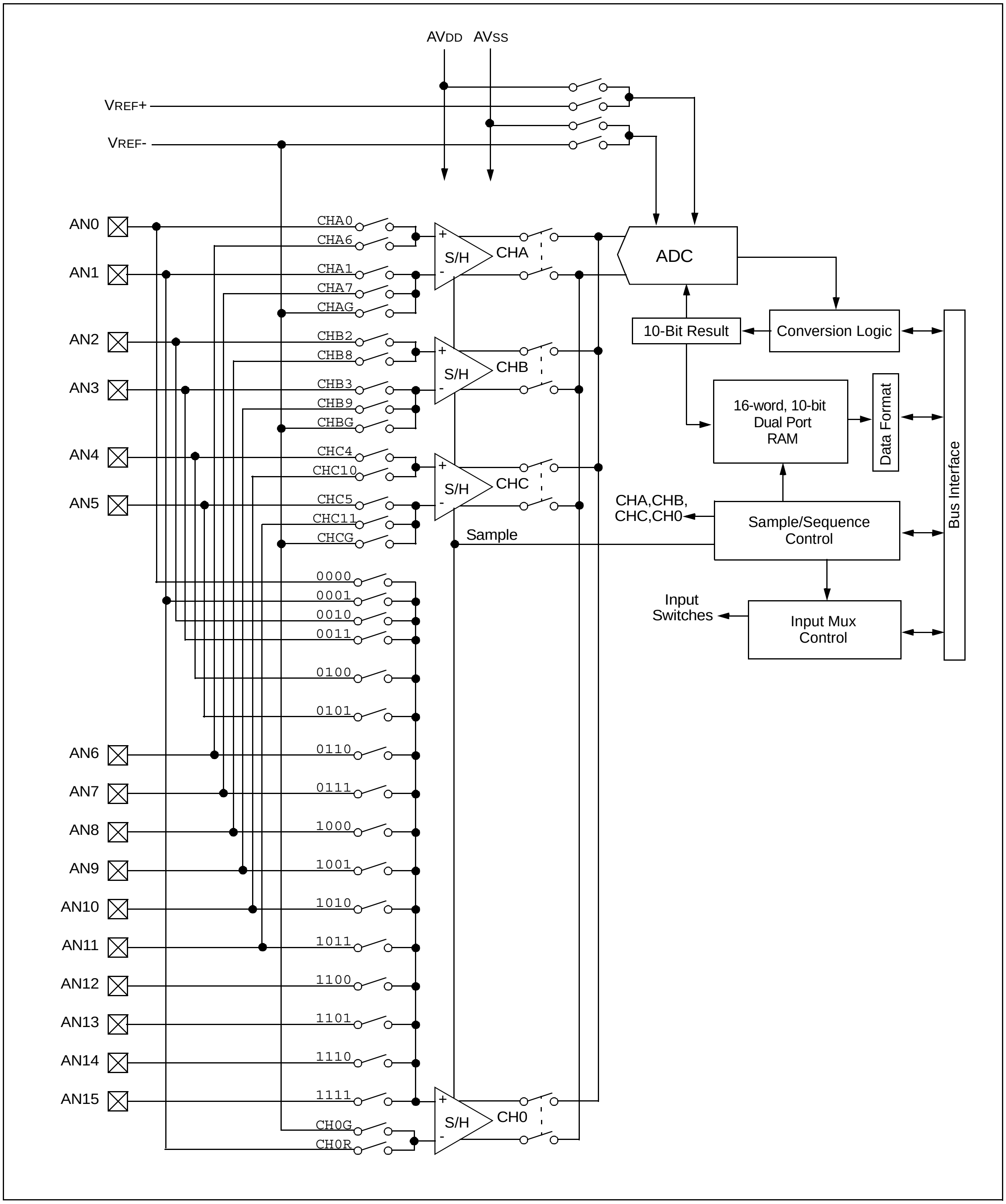
- A/D Control Register1 (ADCON1)
- A/D Control Register2 (ADCON2)
- A/D Control Register3 (ADCON3)
- A/D Input Select Register (ADCHS)
- A/D Port Configuration Register (ADPCFG)
- A/D Input Scan Selection Register (ADCSSL)

The ADCON1, ADCON2 and ADCON3 registers control the operation of the A/D 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, 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 A/D module is shown in Figure 23-1.

FIGURE 23-1: 10-BIT HIGH SPEED A/D FUNCTIONAL BLOCK DIAGRAM



23.1 A/D Result Buffer

The module contains a 16-word dual port RAM, called ADRES<15:0>, to buffer the A/D results. The RAM is 10-bits wide, but is read into different format 16-bit words.

Only word writes are allowed to the result buffer. If byte writes are attempted, the results are indeterminate.

23.2 Conversion Operation

After the A/D module has been configured as desired, the sampling is started by setting the SAMP bit. Various sources, such as a programmable bit, timer time-outs and external events, will terminate sampling and start a conversion. When the A/D conversion is completed, the result is loaded into ADRES<15:0>, the CONV bit is cleared, and if, at the correct number of samples as specified by the SMPI bit, the A/D interrupt flag ADIF is set.

The following steps should be followed for doing an A/D conversion:

1. Configure the A/D 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 A/D module
2. Configure A/D interrupt (if required):
    - Clear ADIF bit
    - Select A/D interrupt priority
3. Start sampling.
4. Wait the required sampling time.
5. Trigger sample end, start conversion:
    - Module sets CONV bit
6. Wait for A/D conversion to complete, by either:
    - Polling for the CONV bit to be cleared
    - Waiting for the A/D interrupt
7. Read A/D result buffer, clear ADIF if required.

23.3 Selecting the Conversion Sequence

Several groups of control bits select the sequence that 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 sample/convert sequence for multiple channels. If the SIMSAM bit is 0, the two or four selected channels are sampled and converted sequentially, with two or four sample clocks. If the SIMSAM bit is 1, two or four selected channels are sampled 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 bits selects how many channels are sampled. This can vary from 1, 2 or 4 channels. If CHPS selects 1 channel, the CH0 channel will be sampled and at the sample clock, CH0 will be converted. The result is stored in the buffer. If CHPS selects 2 channels, the CH0 and CHA channels will be sampled and converted. If CHPS selects 4 channels, the CH0, CHA, CHB and CHC channels will be sampled and converted.

The SMPI bits will select how many sample clocks occur 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, will split the 16-word results buffer (ADRES) into two 8-word groups. Writing to the 8-word buffers will be alternated on each interrupt event. Use of the BUFM bit will depend 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 sample and convert one channel, the BUFM bit can be 0 and up to 16 conversions may be done per interrupt. The processor will have one sample and conversion time to move the sixteen conversions.

If the processor cannot unload the buffer within the sample and conversion time, the BUFM bit should be 1. For example, if SMPI<3:0> (ADCON2<5:2>) = `0111`, then eight conversions will be loaded into 1/2 of the buffer, following which, an interrupt occurs. The next eight conversions will be loaded into the other 1/2 of the buffer. The processor will have 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. If the ALTS bit is 0, only the SAMPLE A inputs are selected for sampling. If the ALTS bit is 1 and SMPI<3:0> = `0000`, on the first sample/convert sequence, the SAMPLE A inputs are selected, and on the next sample/convert sequence, the SAMPLE B inputs are selected.

The CSCNA bit (ADCON2<10>) will allow the CH0 channel inputs to be scanned across a selected number of analog inputs during SAMPLE A samples. 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 occurs. If the number of inputs selected is greater than the number of samples taken per interrupt, the higher numbered inputs are unused.

23.4 Programming the Start-of-Conversion Trigger

The sample trigger will terminate sampling and start the requested conversions.

The SSRC<2:0> bits select the source of the sample trigger.

When SSRC<2:0> = `000`, the sample trigger is under software control. Clearing the SAMP bit will cause the sample trigger.

When SSRC<2:0> = `111`, the sample trigger is under A/D clock control. The SAMC bits select the number of A/D clocks between the start of sampling 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 or external interrupts.

The SSRC bits provide for up to 6 alternate sources of sample trigger.

23.5 Aborting a Conversion

Clearing the CONV bit during a conversion will abort the current conversion and stop the sampling sequencing until the next sampling trigger. The ADRES will not be updated with the partially completed A/D conversion sample. That is, the ADRES will continue to contain the value of the last completed conversion (or the last value written to the ADRES register).

If the clearing of the CONV bit coincides with an auto start, the clearing has a higher priority.

After the A/D conversion is aborted, a $2T_{AD}$ wait is required before the next sampling may be started by setting the SAMP bit.

If sequential sampling is specified, the A/D will continue at the next sample pulse which corresponds with the next channel converted. If simultaneous sampling is specified, the A/D will continue with the next multichannel group conversion sequence.

23.6 Selecting the A/D Conversion Clock

The A/D conversion requires 11 T_{AD}. The source of the A/D conversion clock is software selected using a six bit counter. There are 64 possible options for T_{AD}.

$T_{AD} = T_{CY} * (0.5*(ADCS<5:0> +1))$

The internal RC oscillator is selected by setting the ADRC bit.

For correct A/D conversions, the A/D conversion clock (T_{AD}) must be selected to ensure a minimum T_{AD} time of 150 nsec. Table 23-1 shows the resultant T_{AD} times derived from the device operating frequencies and the A/D clock source selected.

TABLE 23-1: TYPICAL TAD VS. DEVICE OPERATING FREQUENCIES

AD Clock Interval (TAD /TCONV)							
AD Clock Source Select			**Device TCY/Device MIPS/TQ/FOSC**				
Clock	**ADRC**	**ADCS<5:0>**	**33.33 nsec/ 30 MIPS/ 8.33 nsec/ 120 MHz**	**40 nsec/ 25 MIPS/ 10 nsec/ 100 MHz**	**80 nsec/ 12.5 MIPS/ 20 nsec/ 50 MHz**	**160 nsec/ 6.25 MIPS/ 40 nsec/ 25 MHz**	**1000 nsec/ 1 MIPS/ 250 nsec/ 4 MHz**
2 TQ	0	000000	16.67 ns[2]/ 0.22 µs	20 ns[2]/ 0.26 µs	40 ns[2]/ 0.52 µs	80 ns[2]/ 1.04 µs	500 ns/ 6.5 µs
4 TQ	0	000001	33.33 ns[2]/ 0.44 µs	40 ns[2]/ 0.52 µs	80 ns[2]/ 1.04 µs	160 ns/ 2.08 µs	1.0 µs/ 13 µs
8 TQ	0	000011	66.66 ns[2]/ 0.88 µs	80 ns[2]/ 1.04 µs	160 ns/ 2.08 µs	320 ns/ 4.16 µs	2.0 µs[3]/ 26 µs
16 TQ	0	000111	133.32 ns[2]/ 1.76 µs	160 ns/ 2.08 µs	320 ns/ 4.16 µs	640 ns[3]/ 8.32 µs	4.0 µs[3]/ 52 µs
32 TQ	0	001111	266.64 ns/ 3.52 µs	320 ns/ 4.16 µs	640 ns[3]/ 8.32 µs	1.28 µs[3]	8.0 µs[3]/ 104 µs
64 TQ	0	011111	533.28 ns[3]/ 7.04 µs	640 ns[3]/ 8.32 µs	1.28 µs[3]	2.56 µs[3]	16.0 µs[3]/ 208 µs
128 TQ	0	111111	1066.56 ns[3]/ 14.08 µs	1280 ns[3]/ 16.64 µs	2.56 µs[3]	5.12 µs[3]	32.0 µs[3]/ 416 µs
RC	1	xxxxxx	200 - 400 ns/ 3.9 µs[1,4]	200 - 400 ns/ 3.9 µs[1,4]	200 - 400 ns/ 3.9 µs[1,4]	200 - 400 ns/ 3.9 µs[1,4]	200 - 400 ns/ 3.9 µs[1]

Note 1: The RC source has a typical TAD time of 300 ns for VDD > 3.0V.

2: These values violate the minimum required TAD time.

3: For faster conversion times, the selection of another clock source is recommended.

4: A/D cannot meet full accuracy with RC clock source and FOSC > 20 MHz

23.7 A/D Sampling Requirements

For the A/D converter to meet its specified accuracy, the charge holding capacitor (C_{HOLD}) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 23-2. The source impedance (R_S) and the internal sampling switch (R_{SS}) impedance directly affect the time required to charge the capacitor C_{HOLD}. The sampling switch (R_{SS}) impedance varies over the device voltage (V_{DD}), see Figure 23-2. The impedance for analog sources must be small enough to meet accuracy requirements at the given speed. After the analog input channel is selected (changed), this sampling must be done before the conversion can be started.

To calculate the minimum sampling time, Equation 23-1 may be used. This equation assumes that the input is stepped some multiple (n) of the LSB step size and the output must be captured to within 1/2 LSb error (2096 steps for 10-bit A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

The C_{HOLD} is assumed to be 5 pF for the A/D.

EQUATION 23-1: A/D SAMPLING TIME EQUATIONS

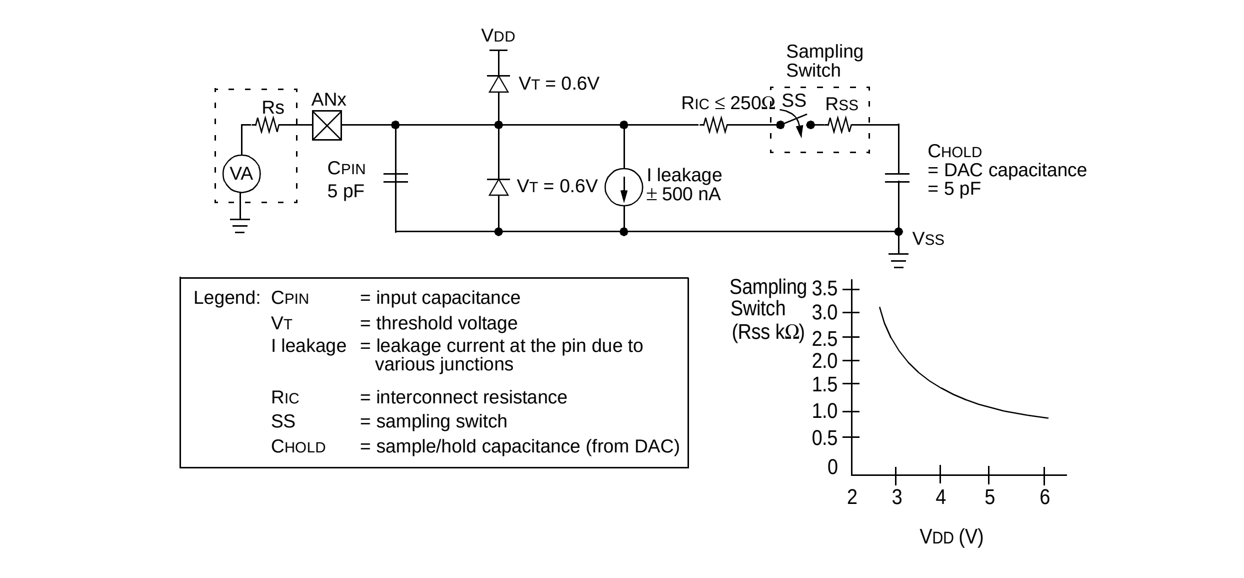
$$\Delta V_O = \Delta V_I \bullet (1 - e^{(-T_C/C_{HOLD}\ (R_{IC}+R_{SS}+R_S))})$$
$$1 - (\Delta V_O / \Delta V_I) = e^{(-T_C/C_{HOLD}\ (R_{IC}+R_{SS}+R_S))}$$
$$\Delta V_I = n \bullet LSB$$
$$\Delta V_O = n \bullet LSB - 1/2\ LSB$$
$$\Delta V_O / \Delta V_I = (n \bullet LSB - 1/2\ LSB) / n \bullet LSB$$
$$1 - (\Delta V_O / \Delta V_I) = 1 / 2n$$
$$1 / 2n = e^{(-T_C/C_{HOLD}\ (R_{IC}+R_{SS}+R_S))}$$
$$T_C = C_{HOLD} \bullet (R_{IC}+R_{SS}+R_S) \bullet -In(1/2 \bullet n)$$

TSMP = Amplifier Settling Time
+ Holding Capacitor Charging Time (T_C)
+Temperature Coefficient †

† The temperature coefficient is only required for temperatures > 25°C.

$$TSMP = 0.5\text{ ms} + C_{HOLD} \bullet (R_{IC}+R_{SS}+R_S) \bullet -In(1/2 \bullet n) + [(Temp - 25°C)(0.05\text{ ms}/°C)]$$

FIGURE 23-2: ANALOG INPUT MODEL



Legend:		
	CPIN	= input capacitance
	VT	= threshold voltage
	I leakage	= leakage current at the pin due to various junctions
	RIC	= interconnect resistance
	SS	= sampling switch
	CHOLD	= sample/hold capacitance (from DAC)

23.8 Module Power-down Modes

The module has 3 internal Power modes.

When the ADON bit is 1, the module is in Active mode and is fully powered and functional.

When ADON is 0 and ADSTBY is 1, the module is in Standby mode. In Standby mode, the digital portions of the module are active; however, the analog portions are powered down, including the bias generators.

When ADON is 0 and ADSTBY 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 Standby or Off mode, the user must wait for the bias generators to stabilize.

23.9 A/D Operation During CPU SLEEP and IDLE Modes

23.9.1 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 exiting from SLEEP mode.

Register contents are not affected by the device entering or leaving SLEEP mode.

The A/D 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 A/D module waits for 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 completed, the CONV bit will be cleared and the result loaded into the ADRES register.

If the A/D interrupt is enabled, the device will wake-up from SLEEP. If the A/D interrupt is not enabled, the A/D module will then be turned off, although the ADON bit will remain set.

23.9.2 A/D OPERATION DURING CPU IDLE MODE

For the A/D, the ADSIDL bit selects if the module will stop on IDLE or continue on IDLE. If ADSIDL = 0, the module will continue operation on assertion of IDLE mode. If ADSIDL = 1, the module will stop on IDLE.

23.10 Effects of a RESET

A device RESET forces all registers to their RESET state. This forces the A/D module to be turned off, and any conversion and sampling sequence are aborted. The values that are in the ADRES registers are not modified. The A/D result register will contain unknown data after a Power-on Reset.

23.11 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 right justified (integer) format.

FIGURE 23-3: A/D OUTPUT DATA FORMATS

RAM contents:							d09	d08	d07	d06	d05	d04	d03	d02	d01	d00
Read to Bus:																
Signed Fractional (1.15)	$\overline{d09}$	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0
Fractional (1.15)	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00	0	0	0	0	0	0
Signed Integer	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	$\overline{d09}$	d08	d07	d06	d05	d04	d03	d02	d01	d00
Integer	0	0	0	0	0	0	d09	d08	d07	d06	d05	d04	d03	d02	d01	d00

23.12 Configuring Analog Port Pins

The use of the ADCON1 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 (V_{OH} or V_{OL}) will be converted.

The A/D operation is independent of the state of the CHS<3:0> bits and the TRIS bits.

When reading the port register, all pins configured as analog input channel will read as cleared (a low level).

Pins configured as digital inputs, will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.

Analog levels on any pin that is defined as a digital input (including the AN pins), may cause the input buffer to consume current that exceeds the device specifications.

23.13 Connection Considerations

Since the analog inputs employ ESD protection, they have diodes to V_{DD} and V_{SS}. This requires that the analog input must 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 23-2: 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										uuuu uu00 0000 0000
ADCBUF1	0282	—	—	—	—	—	—	ADC Data Buffer 1										uuuu uu00 0000 0000
ADCBUF2	0284	—	—	—	—	—	—	ADC Data Buffer 2										uuuu uu00 0000 0000
ADCBUF3	0286	—	—	—	—	—	—	ADC Data Buffer 3										uuuu uu00 0000 0000
ADCBUF4	0288	—	—	—	—	—	—	ADC Data Buffer 4										uuuu uu00 0000 0000
ADCBUF5	028A	—	—	—	—	—	—	ADC Data Buffer 5										uuuu uu00 0000 0000
ADCBUF6	028C	—	—	—	—	—	—	ADC Data Buffer 6										uuuu uu00 0000 0000
ADCBUF7	028E	—	—	—	—	—	—	ADC Data Buffer 7										uuuu uu00 0000 0000
ADCBUF8	0290	—	—	—	—	—	—	ADC Data Buffer 8										uuuu uu00 0000 0000
ADCBUF9	0292	—	—	—	—	—	—	ADC Data Buffer 9										uuuu uu00 0000 0000
ADCBUFA	0294	—	—	—	—	—	—	ADC Data Buffer 10										uuuu uu00 0000 0000
ADCBUFB	0296	—	—	—	—	—	—	ADC Data Buffer 11										uuuu uu00 0000 0000
ADCBUFC	0298	—	—	—	—	—	—	ADC Data Buffer 12										uuuu uu00 0000 0000
ADCBUFD	029A	—	—	—	—	—	—	ADC Data Buffer 13										uuuu uu00 0000 0000
ADCBUFE	029C	—	—	—	—	—	—	ADC Data Buffer 14										uuuu uu00 0000 0000
ADCBUFF	029E	—	—	—	—	—	—	ADC Data Buffer 15										uuuu uu00 0000 0000
ADCON1	02A0	ADON	—	ADSIDL	ADSTBY	—	—	FORM<1:0>		SSRC<2:0>			—	SIMSAM	ASAM	SAMP	CONV	0u00 uu00 000u 0000
ADCON2	02A2	VCFG<2:0>			OFFCAL	—	CSCNA	CHPS<1:0>		BUFS	—	SMPI<3:0>				BUFM	ALTS	0000 u000 0u00 0000
ADCON3	02A4	—	—	—	SAMC<4:0>					ADRC	—	ADCS<5:0>						uuu0 0000 0u00 0000
ADCHS	02A6	CHXNB<1:0>		CHXSB	CH0NB	CH0SB<3:0>				CHXNA<1:0>		CHXSA	CH0NA	CH0SA<3:0>				0000 0000 0000 0000
ADPCFG	02A8	PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8	PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000 0000 0000 0000
ADCSSL	02AA	CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8	CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0	0000 0000 0000 0000

REGISTER 23-1: ADCON1: A/D CONTROL REGISTER1

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
ADON	—	ADSIDL	ADSTBY	—	—	FORM1	FORM0
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0 HC, HS	R/C-0 HC, HS
SSRC2	SSRC1	SSRC0	—	SIMSAM	ASAM	SAMP	CONV
bit 7							bit 0

bit 15 **ADON:** A/D Operating Mode bit
`1` = A/D converter module is operating
`0` = A/D converter is in Standby mode or off

bit 14 **Unimplemented:** Read as '0'

bit 13 **ADSIDL:** Stop in IDLE Mode bit
`1` = Discontinue module operation when device enters IDLE mode
`0` = Continue module operation in IDLE mode

bit 12 **ADSTBY:** A/D Standby Mode bit
`1` = If ADON = 0, A/D converter module in Standby mode. Analog circuits powered down.
`0` = If ADON = 0, A/D converter is shut-off and consumes no operating current

bit 11-10 **Unimplemented:** Read as '0'

bit 9-8 **FORM<1:0>:** Data Output Format bits
`11` = Signed Fractional (DOUT = `sddd dddd dd00 0000` where `s` = .NOT.d<9>)
`10` = Fractional (DOUT = `dddd dddd dd00 0000`)
`01` = Signed Integer (DOUT = `ssss sssd dddd dddd` where `s` = .NOT.d<9>)
`00` = Integer (DOUT = `0000 00dd dddd dddd`)

bit 7-5 **SSRC<2:0>:** Sample Clock Source Select bits
`111` = Internal counter ends sampling and starts conversion (auto convert)
`110` = Reserved
`101` = Reserved
`100` = Reserved
`011` = Motor Control PWM interval ends sampling and starts conversion
`010` = GP Timer compare ends sampling and starts conversion
`001` = Active transition on INT pin ends sampling and starts conversion
`000` = Clearing sample bit ends sampling and starts conversion

bit 4 **Unimplemented:** Read as '0'

bit 3 **SIMSAM:** Simultaneous Sample Select bit (only applicable when CHPS = `01` or `1x`)
`1` = Samples CH0, CHA, CHB, CHC simultaneously (when CHPS = `1x`)
- or -
Samples CH0 and CHA simultaneously (when CHPS = `01`)
`0` = Samples multiple channels individually in sequence

bit 2 **ASAM:** A/D Sample Auto Start bit
`1` = Sampling begins immediately after last conversion. SAMP bit is auto set.
`0` = Sampling begins when SAMP bit set

bit 1 **SAMP:** A/D Sample Enable bit
`1` = A/D sample/hold amplifiers are sampling
`0` = A/D sample/hold amplifiers are holding

REGISTER 23-1: ADCON1: A/D CONTROL REGISTER1 (Continued)

bit 0 **CONV:** A/D Conversion Status bit
`1` = A/D conversion cycle in progress. Cleared by hardware when A/D conversion complete.
`0` = A/D conversion completed/not in progress.

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

REGISTER 23-2: ADCON2: A/D CONTROL REGISTER2

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
VCFG2	VCFG1	VCFG0	OFFCAL	—	CSCNA	CHPS1	CHPS0
bit 15							bit 8

Lower Byte:							
R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI3	SMPI2	SMPI1	SMPI0	BUFM	ALTS
bit 7							bit 0

bit 15-13 **VCFG<2:0>:** Voltage Reference Configuration bits

	A/D VREF+	A/D VREF-
000	AVDD	AVSS
001	External VREF+	AVSS
010	AVDD	External VREF-
011	External VREF+	External VREF-
100	Internal VRH	Internal VRL
101	Internal VRL	AVSS
101	Internal VRH	AVSS
111	AVDD	Internal VRL
Note: Shaded regions in the table are unused. If these options are selected, the A/D references will default to AVDD and AVSS.		

bit 12 **OFFCAL:** Selects Offset Calibration Mode bit
1 = + and - inputs of channel sample/hold shorted together
0 = + and - inputs of channel sample/hold normal

bit 11 **Unimplemented:** Read as '0'

bit 10 **CSCNA:** Scan Input Selections for CH0+ during SAMPLE A bit
1 = Scan inputs
0 = Do not scan inputs

bit 9-8 **CHPS<1:0>:** Selects Channels Utilized bits
1x = Converts CH0, CHA, CHB and CHC
01 = Converts CH0 and CHA
00 = Converts CH0

bit 7 **BUFS:** Buffer Fill Status bit
Only valid when BUFM =1
1 = A/D is currently filling buffer 0x8 - 0xF, user should access data in 0x0 - 0x7
0 = A/D is currently filling buffer 0x0 - 0x7, user should access data in 0x8 - 0xF

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

bit 5-2 **SMPI<3:0>:** Selects Number of Samples per Interrupt bits
1111 = Interrupts at the completion of conversion for each 16th sample
1110 = Interrupts at the completion of conversion for each 15th sample
. . . . .
0001 = Interrupts at the completion of conversion for each 2nd sample
0000 = Interrupts at the completion of conversion for each sample

bit 1 **BUFM:** Buffer Fill Mode Select bit
1 = Starts buffer filling at address 0x0 on first interrupt and 0x8 on next interrupt
0 = Always starts filling buffer at address 0x0

REGISTER 23-2: ADCON2: A/D CONTROL REGISTER2 (Continued)

bit 0 **ALTS:** Alternate Input Sample Mode Select bit
`1` = Uses channel input selects for SAMPLE A on first sample and SAMPLE B on next sample
`0` = Always uses channel input selects for SAMPLE A

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

REGISTER 23-3: ADCON3: A/D CONTROL REGISTER3

Upper Byte:							
U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0
bit 15							bit 8

Lower Byte:							
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	—	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 15-13 **Unimplemented:** Read as '0'

bit 12-8 **SAMC<4:0>:** Auto Sample Time bits
11111 = 31 T_{AD}
· · · · ·
00001 = 1 T_{AD}
00000 = 0 T_{AD}

bit 7 **ADRC:** A/D Conversion Clock Source bit
1 = A/D internal RC clock
0 = Clock derived from system clock

bit 6 **Unimplemented:** Read as ’0’

bit 5-0 **ADCS<5:0>:** A/D Conversion Clock Select bits
111111 = $T_Q \cdot 2 \cdot$ (ADCS<5:0> +1) = $128 \cdot T_Q$
· · · · · ·
000001 = $T_Q \cdot 2 \cdot$ (ADCS<5:0> +1) = $4 \cdot T_Q$
000000 = $T_Q \cdot 2 \cdot$ (ADCS<5:0> +1) = $2 \cdot T_Q$

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

REGISTER 23-4: ADCHS: A/D INPUT SELECT REGISTER

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CHXNB1	CHXNB0	CHXSB	CH0NB	CH0SB3	CH0SB2	CH0SB1	CH0SB0
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CHXNA1	CHXNA0	CHXSA	CH0NA	CH0SA3	CH0SA2	CH0SA1	CH0SA0
bit 7							bit 0

bit 15-14 **CHXNB<1:0>:** Channel A, B, C Negative Input Select for SAMPLE B bits
Same definition as bits <6:7>

bit 13 **CHXSB:** Channel A, B, C Positive Input Select for SAMPLE B bit
Same definition as bit <5>

bit 12 **CH0NB:** Channel 0 Negative Input Select for SAMPLE B bit
Same definition as bit <4>

bit 11-8 **CH0SB<3:0>:** Channel 0 Positive Input Select for SAMPLE B bits
Same definition as bits <3:0>

bit 6-7 **CHXNA<1:0>:** Channel A, B, C Negative Input Select for SAMPLE A bits
11 = CHA negative input is AN9, CHB negative input is AN10, CHC negative input is AN11
10 = CHA negative input is AN6, CHB negative input is AN7, CHC negative input is AN8
0x = CHA, CHB, CHC negative input is VREF-

bit 5 **CHXSA:** Channel A, B, C Positive Input Select for SAMPLE A bit
1 = CHA positive input is AN3, CHB positive input is AN4, CHC positive input is AN5
0 = CHA positive input is AN0, CHB positive input is AN1, CHC positive input is AN2

bit 4 **CH0NA:** Channel 0 Negative Input Select for SAMPLE A bit
1 = Channel 0 negative input is AN1
0 = Channel 0 negative input is VREF-

bit 3-0 **CH0SA<3:0>:** Channel 0 Positive Input Select for SAMPLE A bit

1111	= Channel 0 positive input is AN15
1110	= Channel 0 positive input is AN14
1101	= Channel 0 positive input is AN13
1100	= Channel 0 positive input is AN12
1011	= Channel 0 positive input is AN11
1010	= Channel 0 positive input is AN10
1001	= Channel 0 positive input is AN9
1000	= Channel 0 positive input is AN8
0111	= Channel 0 positive input is AN7
0110	= Channel 0 positive input is AN6
0101	= Channel 0 positive input is AN5
0100	= Channel 0 positive input is AN4
0011	= Channel 0 positive input is AN3
0010	= Channel 0 positive input is AN2
0001	= Channel 0 positive input is AN1
0000	= Channel 0 positive input is AN0

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

REGISTER 23-5: ADPCFG: A/D PORT CONFIGURATION REGISTER

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 15-0 **PCFG<15:0>:** A/D Port Configuration Control bits
1 = Port pin in Digital mode, port read input enabled, A/D input multiplexor connected to AVSS
0 = Port pin in Analog mode, port read input disabled, A/D samples pin voltage

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

REGISTER 23-6: ADCSSL: A/D INPUT SCAN SELECT REGISTER

Upper Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8
bit 15							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
bit 7							bit 0

bit 15-0 **CSSL<15:0>:** A/D Input Scan Selection bits
1 = Select ANx for input scan
0 = Skip ANx for input scan

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

24.0 SYSTEM INTEGRATION

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 it 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 the application. 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. A set of configuration bits are used to select various options

24.1 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 post-scaler 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 24-1 provides a summary of the dsPIC30F Oscillator Operating modes.

TABLE 24-1: 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 SOSC1:SOSC2(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.
LPRC	32 kHz internal RC Oscillator.

Note 1: dsPIC30F maximum operating frequency of 120 MHz must be met.

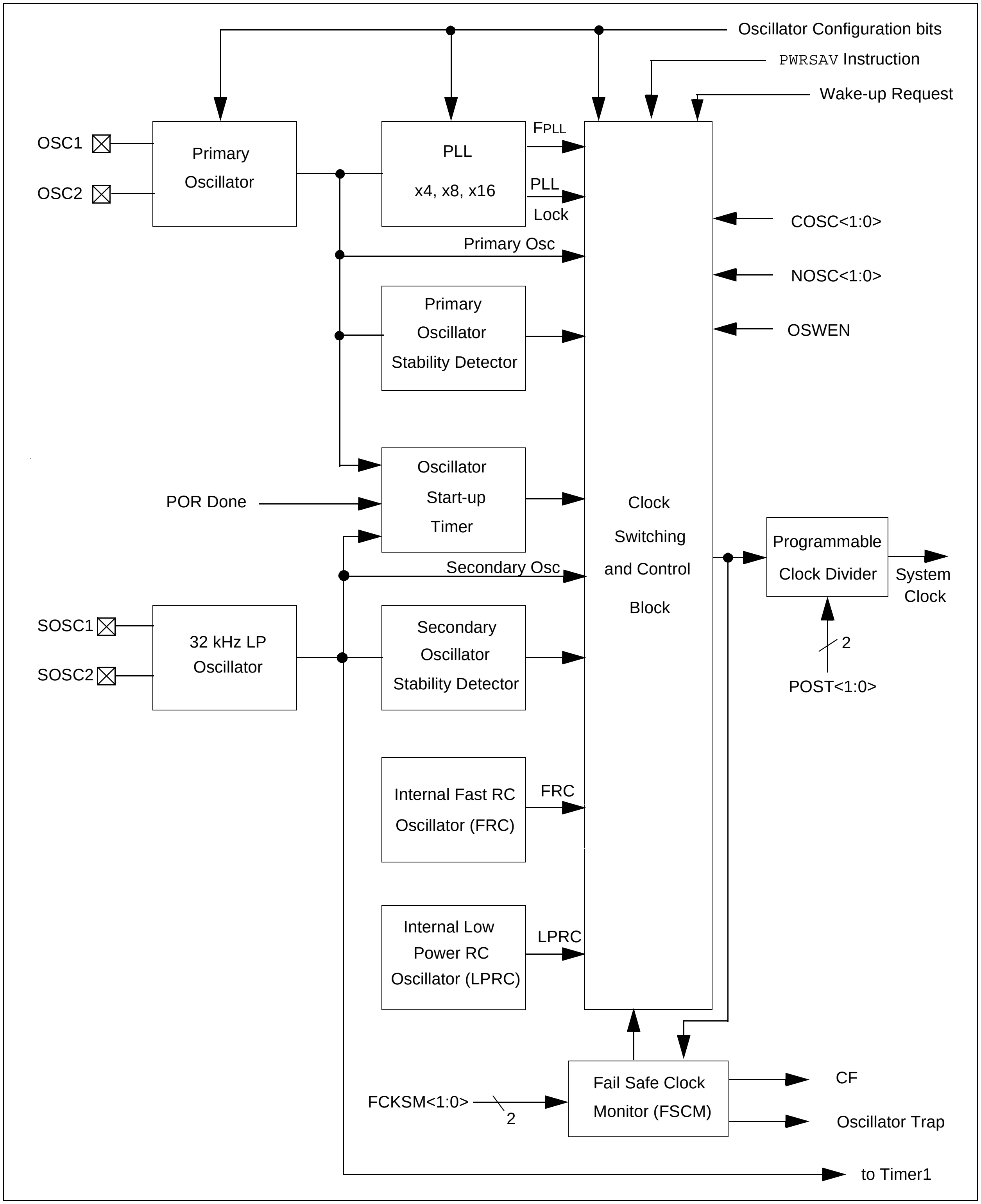
2: LP oscillator can be conveniently shared as system clock as well as real-time clock for Timer1.

3: Requires external R and C. Frequency operation up to 4 MHz.

A simplified diagram of the oscillator system is shown in Figure 24-1.

Configuration bits determine the clock source upon Power-on Reset (POR) and Brown-out Reset (BOR). Thereafter, clock source can be changed between permissible clock sources. The OSCCON register controls the clock switching and reflects system clock related status bits.

FIGURE 24-1: OSCILLATOR SYSTEM BLOCK DIAGRAM


Oscillator Configuration bits
PWRSAV Instruction
Wake-up Request
OSC1
OSC2
Primary Oscillator
PLL x4, x8, x16
FPLL
PLL Lock
COSC<1:0>
NOSC<1:0>
OSWEN
Primary Osc
Primary Oscillator Stability Detector
Oscillator Start-up Timer
POR Done
Clock Switching and Control Block
Programmable Clock Divider
System Clock
2
POST<1:0>
Secondary Osc
SOSC1
SOSC2
32 kHz LP Oscillator
Secondary Oscillator Stability Detector
Internal Fast RC Oscillator (FRC)
FRC
Internal Low Power RC Oscillator (LPRC)
LPRC
FCKSM<1:0>
2
Fail Safe Clock Monitor (FSCM)
CF
Oscillator Trap
to Timer1

24.2 Oscillator Configurations

24.2.1 INITIAL CLOCK SOURCE SELECTION

While coming out of Power-on Reset or Brown-out Reset, the device selects its clock source based on:

a) FOS<1:0> configuration bits that select one of four oscillator groups

b) AND FPR<3:0> configuration bits that select one of 13 oscillator choices within the primary group.

The selection is as shown in Table 24-2.

TABLE 24-2: CONFIGURATION BIT VALUES FOR CLOCK SELECTION

Oscillator Mode	Oscillator Source	FOS1	FOS0	FPR3	FPR2	FPR1	FPR0	OSC2 Function
EC	Primary	1	1	1	1	1	1	CLKOUT
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	CLKOUT
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	X	OSC2
HS	Primary	1	1	0	0	1	X	OSC2
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>).

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.

24.2.2 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 is included. It is a simple 10-bit counter that counts 1024 TOSC cycles before releasing the oscillator clock to the rest of the system. The time-out period is designated as TOST. The TOST 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 Oscillator, XT, XTL, and HS modes (upon wake-up from SLEEP, POR and BOR) for the Primary Oscillator.

24.2.3 LP OSCILLATOR CONTROL

Enabling the LP oscillator is controlled by two elements:

1. The current oscillator group bits COSC<1:0>.
2. The LPOSCEN bit (OSCON register).

In normal operating mode, the LP oscillator is ON if:

- COSC<1:0> = `00` (LP selected as main oscillator) or
- LPOSCEN = `1`

In SLEEP mode, the LP oscillator is ON if
LPOSCEN = `1`.

In IDLE mode, the LP oscillator is ON if:

- COSC<1:0> = `00` (LP selected as main oscillator) or
- 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 will still require a start-up time

24.2.4 PLL

The PLL multiplies the clock which is generated by the primary oscillator. The PLL is selectable to have either gains of x4, x8, and x16. Input and output frequency ranges are summarized in the table below.

TABLE 24-3: PLL FREQUENCY RANGE

FIN	PLL Multiplier	FOUT
4 MHz - 10 MHz	x4	16 MHz - 40 MHz
4 MHz - 10 MHz	x8	32 MHz - 80 MHz
4 MHz - 10 MHz	x16	64 MHz - 160 MHz[(1)]

Note 1: User must ensure **FIN** and selected PLL multiplier does not exceed 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 will be rescinded. The state of this signal is reflected in the read only LOCK bit in the OSCCON register.

24.2.5 FAST RC OSCILLATOR (FRC)

The FRC oscillator is a fast (8 MHz 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.

The dsPIC30F operates from the FRC Oscillator whenever the Current Oscillator Selection control bits in the OSCCON register (OSCCON<13:12>) are set to ‘01’.

24.2.6 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 will remain 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 will shut-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.

24.2.7 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 configuration bits (Clock Switch and Monitor Selection bits) in the FOSC device configuration register. If the FSCM function is enabled, the LPRC Internal oscillator will run at all times (except during SLEEP mode) and will not be subject to control by the SWDTEN bit.

In the event of an oscillator failure, the FSCM will generate a Clock Failure Trap event and will switch the system clock over to the FRC Oscillator. The user will then have the option to either attempt to restart the oscillator or execute a controlled shut-down. 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 will be activated and the FSCM will initiate a Clock Failure Trap, and the COSC<1:0> bits are loaded with FRC oscillator selection. This will effectively shut-off the original oscillator that was trying to start.

The user may detect this situation and restart the oscillator in the Clock Fail Trap ISR.

Upon a clock failure detection the FSCM module will initiate a clock switch to the FRC Oscillator as follows:

1. The COSC 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 will abort a clock transition in progress (used for hang-up situations).

If configuration bits FCKSM<1:0> = `1x`, then the clock switching function and the fail safe clock monitor function 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 will reflect the clock source selection.

24.2.8 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` "`46h`" `to OSCCON low`
- `Byte Write "57h" to OSCCON low`

Byte Write is allowed for one instruction cycle. Write 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` "`78h`" `to OSCCON high`
- `Byte Write` "`9Ah`" `to OSCCON high`

Byte Write is allowed for one instruction cycle. Write "Desired Value" or use bit manipulation instruction.

REGISTER 24-1: OSCCON: OSCILLATOR CONTROL REGISTER

Upper Half:							
U-0	U-0	R-y	R-y	U-0	U-0	R/W-y	R/W-y
—	—	COSC1	COSC0	—	—	NOSC1	NOSC0
bit 15							bit 8

Lower Half:							
R/W-0	R/W-0	R-0	U-0	R/W-0	U-0	R/W-0	R/W-0
POST1	POST0	LOCK	—	CF	—	LPOSCEN	OSWEN
bit 7							bit 0

bit 15-14 **Unimplemented:** Read as '0'

bit 13-12 **COSC<1:0>:** Current Oscillator Source Status bits
11 = Primary oscillator
10 = Internal LPRC oscillator
01 = Internal FRC oscillator
00 = LP crystal 32 kHz crystal oscillator (Timer1)

bit 11-10 **Unimplemented:** Read as '0'

bit 9-8 **NOSC<1:0>**: New Oscillator Group Selection bits
11 = Primary oscillator
10 = Internal LPRC oscillator
01 = Internal FRC oscillator
00 = Low power 32 kHz crystal oscillator (Timer1)

bit 7-6 **POST<1:0>:** Oscillator Postscaler Selection bits
11 = Oscillator postscaler divides clock by 64
10 = Oscillator postscaler divides clock by 16
01 = Oscillator postscaler divides clock by 4
00 = Oscillator postscaler does not alter clock

bit 5 **LOCK:** PLL Lock Status bit
1 = Indicates that PLL is in lock
0 = Indicates that PLL is out of lock (or disabled)

bit 4 **Unimplemented:** Read as '0'

bit 3 **CF:** Clock Fail Status bit
1 = FSCM has detected clock failure
0 = FSCM has NOT detected clock failure

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

bit 1 **LPOSCEN:** 32 kHz LP Oscillator Enable bit
1 = LP oscillator is enabled
0 = LP oscillator is disabled

bit 0 **OSWEN:** Oscillator Switch Enable bit
1 = Request oscillator switch to selection specified by NOSC<1:0> bits
0 = Oscillator switch is complete

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	1 = bit is set	0 = bit is cleared	x = bit is unknown
y = Value set from configuration bits on POR and BOR			

REGISTER 24-2: FOSC: OSCILLATOR SELECTION CONTROL REGISTER

Upper Third:

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

Middle Third:

R/P-1	R/P-1	U-0	U-0	U-0	U-0	R/P-1	R/P-1
FCKSM1	FCKSM0	—	—	—	—	FOS1	FOS0
bit 15							bit 8

Lower Third:

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	FPR3	FPR2	FPR1	FPR0
bit 7							bit 0

bit 23-16 **Unimplemented:** Read as ‘0’

bit 15-14 **FCKSM<1:0>:** Clock Switching and Monitor Selection bits
`1x` = Clock switching is disabled, fail safe clock monitor is disabled
`01` = Clock switching is enabled, fail safe clock monitor is disabled
`00` = Clock switching is enabled, fail safe clock monitor is enabled

bit 13-10 **Unimplemented:** Read as ‘0’

bit 9-8 **FOS<1:0>:** Oscillator Group Selection bits on POR and BOR
`11` = Primary group (13 choices)
`10` = Internal LPRC group (LPRC)
`01` = Internal FRC group (FRC)
`00` = Secondary group (LP external)

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-0 **FPR<3:0>:** Primary Oscillator Selection bits

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

24.3 RESET

The dsPIC30F differentiates between various kinds of RESET:

a) Power-on Reset (POR)

b) $\overline{\text{MCLR}}$ Reset during normal operation

c) $\overline{\text{MCLR}}$ Reset during SLEEP

d) Watchdog Timer (WDT) Reset (during normal operation)

e) Programmable Brown-out Reset (PBOR)

f) `RESET` Instruction

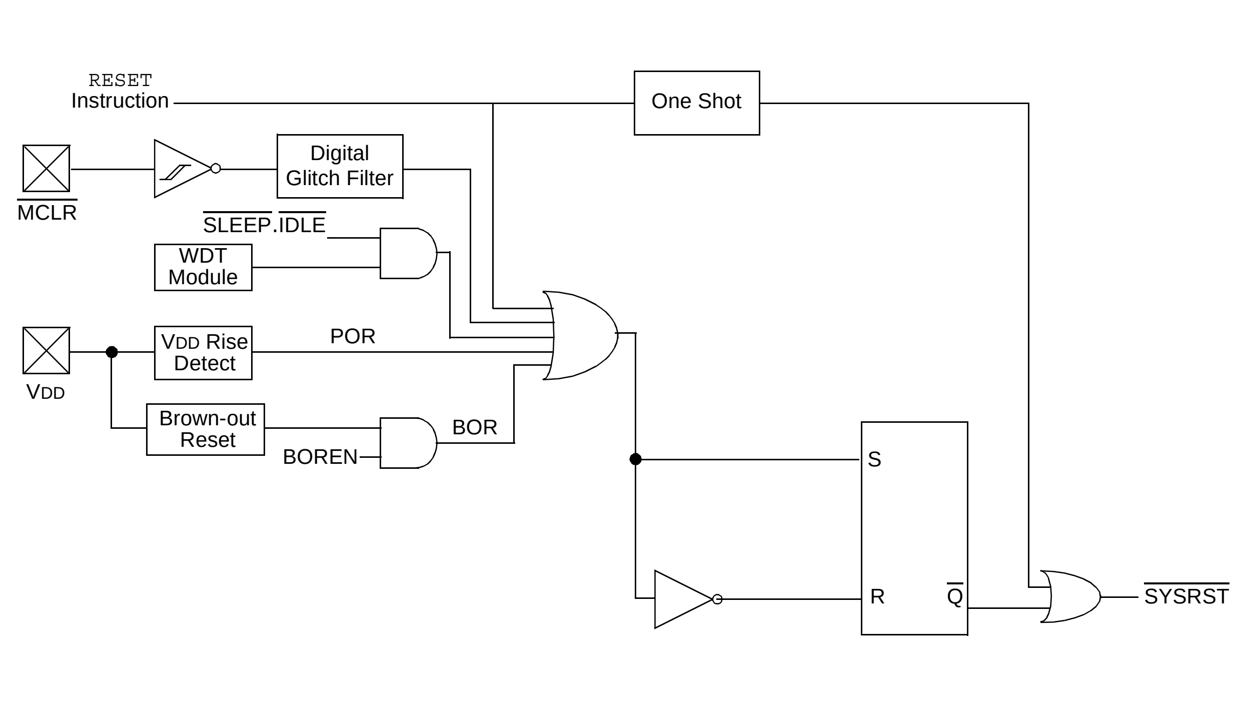
Different registers are affected in different ways by various RESET conditions. Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register are set or cleared differently in different RESET situations, as indicated in Table 24-4. 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 24-2.

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 24-2: RESET SYSTEM BLOCK DIAGRAM



24.3.1 POR: POWER-ON RESET

A power-on event will generate an internal POR Reset pulse when a V_{DD} rise is detected. The RESET pulse will occur at the POR circuit threshold voltage (V_{POR}), which is nominally 1.85V. The device supply voltage characteristics must meet specified starting voltage and rise rate requirements. For more information, please refer to the Electrical Specifications section (TBD) of the specific device data sheet. The POR pulse will reset a POR timer and place 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, T_{POR}, which is nominally 10µs and ensures that the device bias circuits are stable. Furthermore, a user selected power-up time-out (T_{PWRT}) is applied. The T_{PWRT} 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 $T_{POR} + T_{PWRT}$. When these delays have expired, $\overline{\text{SYSRST}}$ will be negated on the next leading edge of the Q1 clock, and the PC will jump to the RESET vector.

The timing for the $\overline{\text{SYSRST}}$ signal is shown in Figure 24-3 through Figure 24-5.

FIGURE 24-3: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ TIED TO VDD)

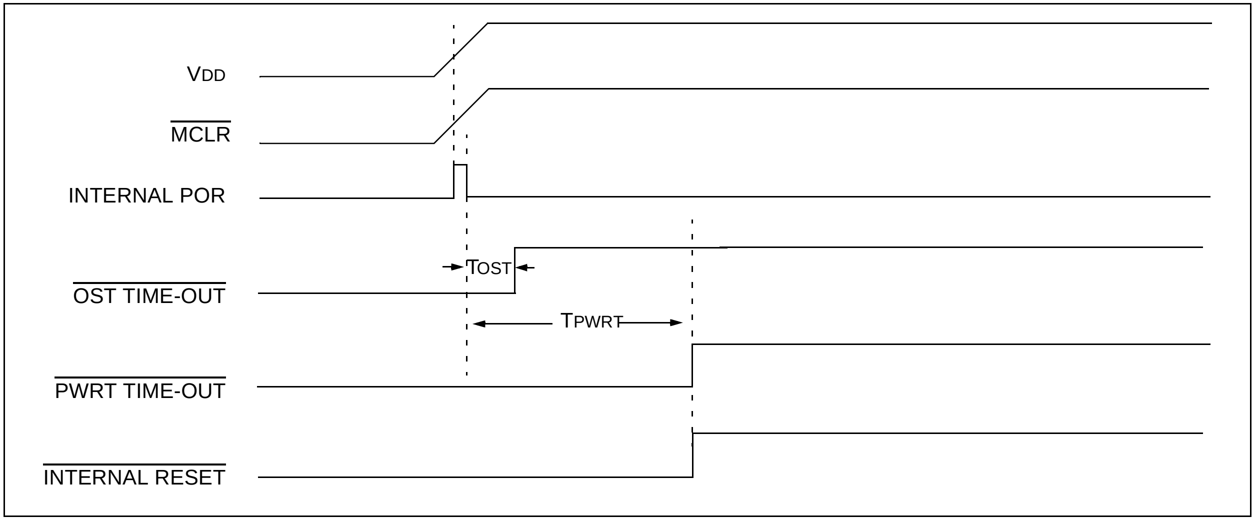


FIGURE 24-4: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO VDD): CASE 1

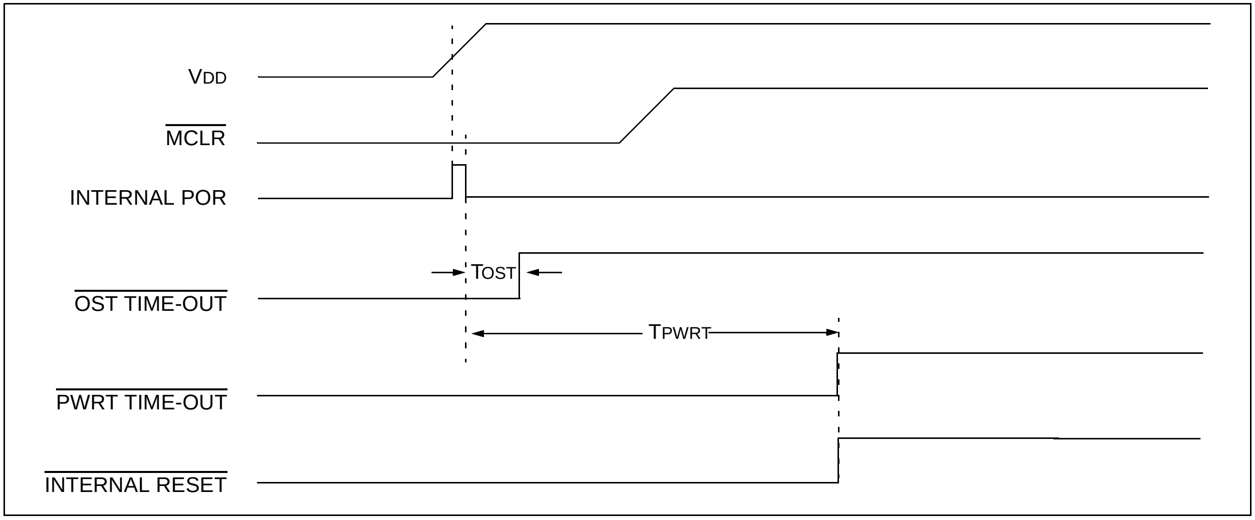
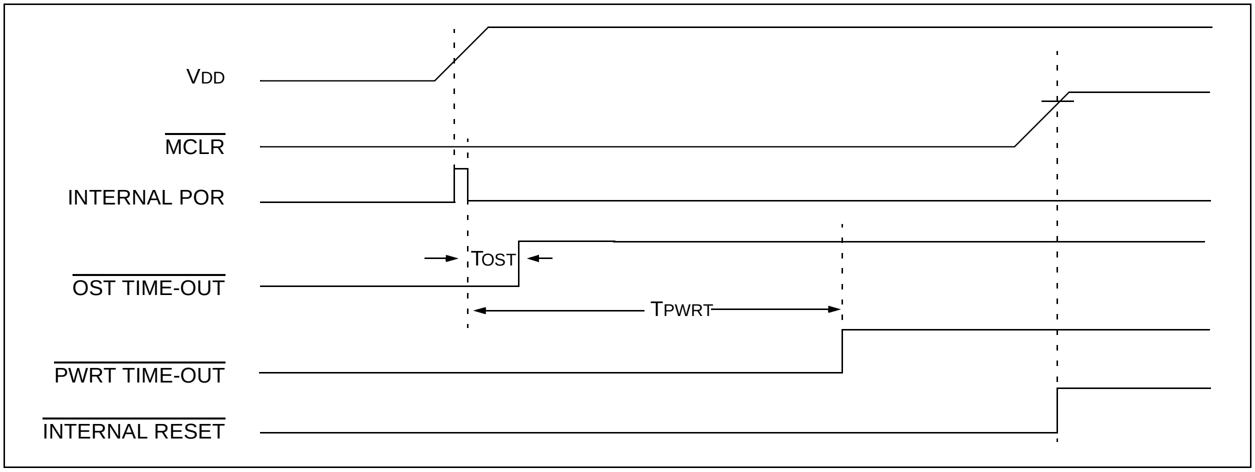


FIGURE 24-5: TIME-OUT SEQUENCE ON POWER-UP ($\overline{\text{MCLR}}$ NOT TIED TO VDD): CASE 2



24.3.1.1 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) will 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 will occur. The device will automatically switch to the FRC oscillator and the user can switch to the desired crystal oscillator in the TRAP ISR.

24.3.1.2 Operating without FSCM and PWRT

If the FSCM is disabled and the Power-up Timer (PWRT) is also disabled, then on a power-up, the device will exit from RESET rapidly. If the clock source is FRC, LPRC, EXTRC 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.

24.3.2 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 waveform portions of the AC cycles 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:

- 2.0V
- 2.7V
- 4.2V
- 4.5V

Note: The BOR voltage trip points indicated here are nominal values provided for design guidance only. Refer to the Electrical Specifications in the specific device data sheet for BOR voltage limit specifications.

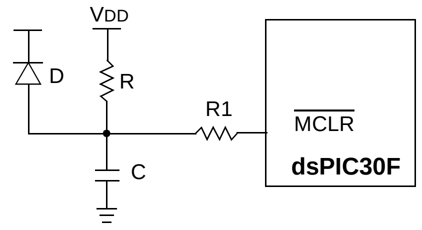
A BOR will generate a RESET pulse which will reset the device. The BOR Reset will select the clock source, based on the device configuration bit values (FOS<1:0> and FPR<3:0>). Furthermore, if an oscillator mode is selected, the BOR Reset will activate the Oscillator Start-up Timer (OST). The system clock is held until OST expires. If the PLL is used, then the clock will be held until the LOCK bit (OSCCON<5>) is "1".

Concurrently, the POR timeout (T_{POR}) and the PWRT time-out (T_{PWRT}) will be applied before the internal RESET is released.

The BOR status bit (RCON<1>) will be set to indicate that a BOR has occurred.

The BOR circuit, if enabled, will continue to operate while in SLEEP or IDLE modes and will reset the device should V_{DD} fall below the BOR threshold voltage.

FIGURE 24-6: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW V_{DD} POWER-UP)



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

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

3: R1 = TBDΩ to TBDkΩ will limit any current flowing into $\overline{\text{MCLR}}$ from external capacitor C, in the event of $\overline{\text{MCLR}}/V_{PP}$ pin breakdown due to Electrostatic Discharge (ESD), or Electrical Overstress (EOS).

Note: Dedicated supervisory devices such as the MCP1XX and MCP8XX may also be used as an external Power-on Reset circuit.

Table 24-4 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 24-4: INITIALIZATION CONDITION FOR RCON REGISTER CASE 1

Condition	Program Counter	EXTR	SWR	WDTO	IDLE	SLEEP	POR	BOR
Power-on Reset	0x0000	0	0	0	0	0	1	1
Brown-out Reset	0x0000	0	0	0	0	0	0	1
$\overline{\text{MCLR}}$ Reset during normal operation	0x0000	1	0	0	0	0	0	0
Software Reset during normal operation	0x0000	0	1	0	0	0	0	0
$\overline{\text{MCLR}}$ Reset during SLEEP	0x0000	1	0	0	0	1	0	0
$\overline{\text{MCLR}}$ Reset during IDLE	0x0000	1	0	0	1	0	0	0
WDT Time-out Reset	0x0000	0	0	1	0	0	0	0
WDT Wake-up	PC + 2	0	0	1	0	1	0	0
Interrupt Wake-up from SLEEP	PC + 2(1)	0	0	0	0	1	0	0
Clock Failure Trap	0x0004	0	0	0	0	0	0	0

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as ’0’

Note 1: When the wake-up is due to an enabled interrupt, the PC is loaded with the corresponding interrupt vector.

Table 24-5 shows a second example of the bit conditions for the RCON Register. In this case, it is not assumed the user has set/cleared specific bits prior to action specified in the condition column.

TABLE 24-5: INITIALIZATION CONDITION FOR RCON REGISTER CASE 2

Condition	Program Counter	EXTR	SWR	WDTO	IDLE	SLEEP	POR	BOR
Power-on Reset	0x0000	0	0	0	0	0	1	1
Brown-out Reset	0x0000	u	u	u	u	u	0	1
$\overline{\text{MCLR}}$ Reset during normal operation	0x0000	1	0	0	0	0	u	u
Software Reset during normal operation	0x0000	0	1	0	0	0	u	u
$\overline{\text{MCLR}}$ Reset during SLEEP	0x0000	1	u	0	0	1	u	u
$\overline{\text{MCLR}}$ Reset during IDLE	0x0000	1	u	0	1	0	u	u
WDT Time-out Reset	0x0000	0	0	1	0	0	u	u
WDT Wake-up	PC + 2	u	u	1	u	1	u	u
Interrupt Wake-up from SLEEP	PC + 2(1)	u	u	u	u	1	u	u
Clock Failure Trap	0x0004	u	u	u	u	u	u	u

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as ’0’

Note 1: When the wake-up is due to an enabled interrupt, the PC is loaded with the corresponding interrupt vector.

REGISTER 24-3: RCON: RESET AND SYSTEM CONTROL REGISTER

Upper Half:

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0
bit 15							bit 8

Lower Half:

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1
EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR
bit 7							bit 0

bit 15:13 **Unimplemented:** Read as ‘0’

bit 12 **LVDEN:** Low Voltage Detect Power Enable bit
1 = Enables LVD, powers up LVD circuit
0 = Disables LVD, powers down LVD circuit

bit 11:8 **LVDL<3:0>:** Low Voltage Detection Limit bits
1111 = External analog input is used (input comes from the LVDIN pin)
1110 = 4.5V min - 4.77V max
1101 = 4.2V min - 4.45V max
1100 = 4.0V min - 4.24V max
1011 = 3.8V min - 4.03V max
1010 = 3.6V min - 3.82V max
1001 = 3.5V min - 3.71V max
1000 = 3.3V min - 3.50V max
0111 = 3.0V min - 3.18V max
0110 = 2.8V min - 2.97V max
0101 = 2.7V min - 2.86V max
0100 = 2.5V min - 2.65V max
0011 = 2.4V min - 2.54V max
0010 = 2.2V min - 2.33V max
0001 = 2.0V min - 2.12V max
0000 = 1.8V min - 1.91V max

bit 7 **EXTR:** External Reset ($\overline{\text{MCLR}}$) Pin bit
1 = A Master Clear (pin) Reset has occurred
0 = A Master Clear (pin) Reset has not occurred

bit 6 **SWR:** Software RESET (Instruction) Flag bit
1 = A ‘RESET’ instruction has been executed
0 = A ‘RESET’ instruction has not been executed

bit 5 **SWDTEN:** Software Enable/Disable of WDT bit
1 = WDT is turned on
0 = WDT is turned off

Note: If FWDTEN configuration bit is = 1, the WDT is ALWAYS ENABLED, regardless of SWDTEN bit setting.

bit 4 **WDTO:** Watchdog Timer Time-out Flag bit
1 = WDT time-out has occurred
0 = WDT time-out has NOT occurred

bit 3 **SLEEP:** Wake from SLEEP Flag bit
1 = Device has been in SLEEP mode
0 = Device has NOT been in SLEEP mode

bit 2 **IDLE:** Wake-up from IDLE Flag bit
1 = Device was in IDLE mode
0 = Device was NOT in IDLE mode

REGISTER 24-3: RCON: RESET AND SYSTEM CONTROL REGISTER (Continued)

bit 1 **BOR:** Brown-out Reset Flag bit
1 = A Brown-out Reset has occurred. Note that BOR is also set after Power-on Reset.
0 = A Brown-out Reset has NOT occurred

bit 0 **POR:** Power-on Reset Flag bit
1 = A Power-up Reset has occurred
0 = A Power-up Reset has NOT occurred

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

24.4 Watchdog Timer (WDT)

24.4.1 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, which runs off an on-chip RC oscillator, requiring no external component. Therefore, the WDT timer will continue to operate even if the main processor clock (e.g., the crystal oscillator) fails.

24.4.2 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 to the desired state.

If enabled, the WDT will increment until it overflows or “times out”. A WDT time-out will force 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 will wake-up. The WDTO bit in the RCON register will be cleared to indicate a wake-up resulting from a WDT time-out.

Setting FWDTEN = 0 allows user software to enable/disable Watchdog Timer via the SWDTEN (RCON<5>) control bit.

REGISTER 24-4: FWDT: CONFIGURATION BITS FOR WDT

Upper Third:

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

Middle Third:

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

Lower Third:

R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-0	R/P-1
FWPSA1	FWPSA0	FWPSB3	FWPSB2	FWPSB1	FWPSB0	—	FWDTEN
bit 7							bit 0

bit 23-8: **Unimplemented:** Read as ‘0’

bit 7-6 **FWPSA<1:0>:** Prescale Value Selection bits for Prescaler A bits
00 = 1:1
01 = 1:8
10 = 1:64
11 = 1:512

bit 5-2 **FWPSB<3:0>:** Prescale Value Selection bits for Prescaler B bits
0000 = 1:1
0001 = 1:2
•
•
•
1110 = 1:15
1111 = 1:16

bit 1 **Unimplemented:** Read as 0

bit 0 **FWDTEN:** Watchdog Enable Configuration bit
1 = Watchdog enabled (LPRC oscillator cannot be disabled. Clearing the SWDTEN bit in the WDTCON register will have no effect.)
0 = Watchdog disabled (LPRC oscillator can be disabled by clearing the SWDTEN bit in the WDTCON register)

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

24.5 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.

24.5.1 SLEEP MODE

In SLEEP mode, the clock to the CPU and the 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, LPRC clock remains active if WDT is operational during SLEEP.

The Brown-out protection circuit and the Low Voltage Detect circuit, if enabled, will remain functional during SLEEP.

The processor wakes up from SLEEP if at least one of the following conditions is true:

- on occurrence of any interrupt that is individually enabled and meets the required priority level
- on any RESET (POR, BOR and $\overline{\text{MCLR}}$)
- on WDT time-out

On waking up from SLEEP mode, the processor will restart the same clock that was active prior to entry into SLEEP mode. When clock switching is enabled, bits COSC<1:0> will determine the oscillator source that will be used on wake-up. If clock switch is disabled, then there is only one system clock.

Note: If a POR or BOR Reset 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 will be 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 EXTRC oscillators are used, then the start-up delay will be a few cycles. A delay of TPOR (~ 10 µs) is applied. This is the smallest delay possible on wake-up from SLEEP.

Moreover, if LP oscillator was active during SLEEP, and LP is the oscillator used on wake-up, then the start-up delay will be equal to TPOR. PWRT delay and OST timer delay are not applied. In order to have the smallest possible start-up delay when waking up from SLEEP, then one of these faster wake-up options should be selected before entering SLEEP.

Any interrupt that is individually enabled (using the corresponding IE bit) and meets the prevailing priority level will be able to wake-up the processor. If control bit GIE = 1, then the processor will process the interrupt and branch to the ISR. If GIE = 0, then the processor will simply continue execution, starting with the instruction immediately following the `SLEEP` instruction. The SLEEP status bit in 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, then the device will detect this as a clock failure and process the Clock Failure Trap, the FRC oscillator will be enabled, and the user will have to re-enable the crystal oscillator. If FSCM is not enabled, then the device will simply suspend execution of code until the clock is stable, and will remain in SLEEP until the oscillator clock has started.

All RESETS will wake-up the processor from SLEEP mode. Any RESET, other than POR, will set the SLEEP status bit. In a POR Reset, the SLEEP bit is cleared.

If Watchdog Timer is enabled, then upon WDT time-out, the processor will wake-up from SLEEP mode. SLEEP and WDTO status bits are both set.

24.5.2 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 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, will be able to wake-up the processor. If control bit GIE = 1, then the processor will process the interrupt and branch to the ISR. If GIE = 0, then the processor will continue execution with the instruction immediately following the `PWRSAV` instruction. The IDLE status bit in RCON register is set upon wake-up.

Any RESET other than POR will set the IDLE status bit. On a POR Reset, the IDLE bit is cleared.

If Watchdog Timer is enabled, then upon WDT time-out, the processor will wake-up from IDLE mode. IDLE and WDTO status bits are both set.

Unlike wake-up from SLEEP, there are no time delays involved in wake-up from IDLE.

24.6 In-Circuit Serial Programming

(TBD)

TABLE 24-6: SIB 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	0400	—	—	—	LVDEN	LVDL<3:0>				EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR	uuu0 0000 0000 0000
OSCCON	0402	—	—	COSC<1:0>		—	—	NOSC<1:0>		POST<1:0>		LOCK	—	CF	—	LPOSCEN	OSWEN	uu00 uu00 000u 0u00

NOTES:

25.0 INSTRUCTION SET SUMMARY

The dsPIC30F instruction set adds many enhancements to the previous PICmicro® instruction sets, while maintaining an easy migration from PICmicro instruction sets.

Most instructions are a single program memory word (24-bits), but there are three instructions that 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 25-1 shows the general symbols used in describing the instructions.

The dsPIC30F instruction set summary in Table 25-2 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’

The destination designator ‘d’ specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in WREG. If 'd' is one, the result is placed in the file register specified in the instruction.

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 desired W register or file register to load the literal value into (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
- The W registers to be used as the two operands
- The X and Y address space pre-fetch operations
- The X and Y address space pre-fetch destinations
- The accumulator write-back destination

The other DSP instructions do not involve any multiplication, and may include:

- The accumulator to be used
- The source or destination operand (designated as Wso or Wdo, respectively) with or without an address modifier
- The amount of shift (in the case of accumulator shift instructions), 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
- No operand required

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 MSb’s are 0’s. If this second word is executed as an instruction (by itself), it will execute as a `NOP`.

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, `TRAP`, and `RETURN/RETFIE` instructions, which are single-word instructions but take two 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.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 120 MHz, the normal instruction execution time is 0.033 µs. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 0.066 µs.

Note:	For more details on the instruction set, refer to the Programmer’s Reference Manual.

TABLE 25-1: SYMBOLS USED IN ROADRUNNER OPCODE DESCRIPTIONS

Field	Description
`#text`	Means literal defined by “`text`“
`(text)`	Means “content of `text`“
`.b`	Byte mode selection
`.d`	Double-word mode selection
`.S`	Shadow register select
`.w`	Word mode selection (default)
`[text]`	Means “the location addressed by `text`”
`{ }`	Optional field or operation
`<n:m>`	Register bit field
`Acc`	One of two accumulators {A, B}
`AWB`	Accumulator write back destination address register ∈ {W13, [W13]+=2}
`bit3`	3-bit bit selection field (used in byte addressed instructions) ∈ {0...7}
`bit4`	4-bit bit selection field (used in word addressed instructions) ∈ {0...15}
`C, DC, N, OV, SZ`	MCU status bits: Carry, Digit Carry, Negative, Overflow, Sticky-Zero
`d`	File register destination d ∈ {WREG, none}
`Expr`	Absolute address, label or expression (resolved by the linker)
`f`	File register address ∈ {0x0000...0x1FFF}
`lit1`	1-bit unsigned literal ∈ {0,1}
`lit14`	14-bit unsigned literal ∈ {0...16384}
`lit16`	16-bit unsigned literal ∈ {0...65535}
`lit23`	23-bit unsigned literal ∈ {0...8388608}; LSB must be 0
`lit4`	4-bit unsigned literal ∈ {0...15}
`lit5`	5-bit unsigned literal ∈ {0...31}
`lit8`	8-bit unsigned literal ∈ {0...255}
`lit10`	10-bit unsigned literal ∈ {0...255} for Byte mode, {0:1023} for Word mode
`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 ∈ {-512...511}
`Slit16`	16-bit signed literal ∈ {-32768...32767}
`Slit5`	5-bit signed literal ∈ {-16...15}
`text1 Œ {text2, text3, ...}`	`text1` must be in the set of `text2`, `text3`, ...
`Wb`	Base W register ∈ {W0..W15}

TABLE 25-1: SYMBOLS USED IN ROADRUNNER OPCODE DESCRIPTIONS (CONTINUED)

Field	Description
Wd	Destination W register ∈ { Wd, [Wd], [Wd++], [Wd--], [++Wd], [--Wd] }
Wdo	Destination W register ∈ { 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 ∈ {W4*W4,W5*W5,W6*W6,W7*W7}
Wm*Wn	Multiplicand and Multiplier working register pair for DSP instructions ∈ {W4*W5,W4*W6,W4*W7,W5*W6,W5*W7,W6*W7}
Wn	One of 16 working registers ∈ {W0..W15}
Wnd	One of 16 destination working registers ∈ {W0..W15}
Wns	One of 16 source working registers ∈ {W0..W15}
WREG	W0 (working register used in file register instructions)
Ws	Source W register ∈ { Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws] }
Wso	Source W register ∈ { Wns, [Wns], [Wns++], [Wns--], [++Wns], [--Wns], [Wns+Wb] }
Wx	X data space pre-fetch address register for DSP instructions ∈ {[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],none}
Wxd	X data space pre-fetch destination register for DSP instructions ∈ {W4..W7}
Wy	Y data space pre-fetch address register for DSP instructions ∈ {[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],none}
Wyd	Y data space pre-fetch destination register for DSP instructions ∈ {W4..W7}

TABLE 25-2: INSTRUCTION SET OVERVIEW

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of words	# of cycles	Status Flags Affected
1	ADD	ADD Acc	Add Accumulators	1	1	OA,OB,SA,SB
		ADD f	f = f + WREG	1	1	C,DC,N,OV,SZ
		ADD f,WREG	WREG = f + WREG	1	1	C,DC,N,OV,SZ
		ADD #lit10,Wn	Wd = lit10 + Wd	1	1	C,DC,N,OV,SZ
		ADD Wb,Ws,Wd	Wd = Wb + Ws	1	1	C,DC,N,OV,SZ
		ADD Wb,#lit5,Wd	Wd = Wb + lit5	1	1	C,DC,N,OV,SZ
		ADD Wso,#Slit4,Acc	16-bit Signed Add to Accumulator	1	1	OA,OB,SA,SB
2	ADDC	ADDC f	f = f + WREG + (C)	1	1	C,DC,N,OV,SZ
		ADDC f,WREG	WREG = f + WREG + (C)	1	1	C,DC,N,OV,SZ
		ADDC #lit10,Wn	Wd = lit10 + Wd + (C)	1	1	C,DC,N,OV,SZ
		ADDC Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1	C,DC,N,OV,SZ
		ADDC Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1	C,DC,N,OV,SZ
3	AND	AND f	f = f .AND. WREG	1	1	N,SZ
		AND f,WREG	WREG = f .AND. WREG	1	1	N,SZ
		AND #lit10,Wn	Wd = lit10 .AND. Wd	1	1	N,SZ
		AND Wb,Ws,Wd	Wd = Wb .AND. Ws	1	1	N,SZ
		AND Wb,#lit5,Wd	Wd = Wb .AND. lit5	1	1	N,SZ
4	ASR	ASR f	f = Arithmetic Right Shift f	1	1	C,N,OV,SZ
		ASR f,WREG	WREG = Arithmetic Right Shift f	1	1	C,N,OV,SZ
		ASR Ws,Wd	Wd = Arithmetic Right Shift Ws	1	1	C,N,OV,SZ
		ASR Wb,Wns,Wnd	Wnd = Arithmetic Right Shift Wb by Wns	1	1	N,SZ
		ASR Wb,#lit5,Wnd	Wnd = Arithmetic Right Shift Wb by lit5	1	1	N,SZ
5	BCLR	BCLR.b f,#bit3	Bit Clear f	1	1	None
		BCLR Ws,#bit4	Bit Clear Ws	1	1	None
6	BRA	BRA C,Expr	Branch if Carry	1	1 (2)	None
		BRA GE,Expr	Branch if greater than or equal	1	1 (2)	None
		BRA GEU,Expr	Branch if unsigned greater than or equal	1	1 (2)	None
		BRA GT,Expr	Branch if greater than	1	1 (2)	None
		BRA GTU,Expr	Branch if unsigned greater than	1	1 (2)	None
		BRA LE,Expr	Branch if less than or equal	1	1 (2)	None
		BRA LEU,Expr	Branch if unsigned less than or equal	1	1 (2)	None
		BRA LT,Expr	Branch if less than	1	1 (2)	None
		BRA LTU,Expr	Branch if unsigned less than	1	1 (2)	None
		BRA N,Expr	Branch if Negative	1	1 (2)	None
		BRA NC,Expr	Branch if Not Carry	1	1 (2)	None
		BRA NN,Expr	Branch if Not Negative	1	1 (2)	None
		BRA NOV,Expr	Branch if Not Overflow	1	1 (2)	None
		BRA NZ,Expr	Branch if Not Zero	1	1 (2)	None
		BRA OA,Expr	Branch if accumulator A overflow	1	1 (2)	None
		BRA OB,Expr	Branch if accumulator B overflow	1	1 (2)	None
		BRA OV,Expr	Branch if Overflow	1	1 (2)	None
		BRA SA,Expr	Branch if accumulator A saturated	1	1 (2)	None
		BRA SB,Expr	Branch if accumulator B saturated	1	1 (2)	None
		BRA Expr	Branch Unconditionally	1	2	None
		BRA Z,Expr	Branch if Zero	1	1 (2)	None
		BRA Wn	Computed Branch	1	2	None
7	BSET	BSET.b f,#bit3	Bit Set f	1	1	None
		BSET Ws,#bit4	Bit Set Ws	1	1	None
8	BSW	BSW.C Ws,Wb	Write C bit to Ws<Wb>	1	1	None
		BSW.Z Ws,Wb	Write SZ bit to Ws<Wb>	1	1	None
9	BTG	BTG.b f,#bit3	Bit Toggle f	1	1	None
		BTG Ws,#bit4	Bit Toggle Ws	1	1	None
10	BTSC	BTSC.b f,#bit3	Bit Test f, Skip if Clear	1	1 (2 or 3)	None
		BTSC Ws,#bit4	Bit Test Ws, Skip if Clear	1	1 (2 or 3)	None

TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax		Description	# of words	# of cycles	Status Flags Affected
11	BTSS	BTSS.b	f,#bit3	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.b	f,#bit3	Bit Test f	1	1	SZ
		BTST.C	Ws,#bit4	Bit Test Ws to C	1	1	C
		BTST.Z	Ws,#bit4	Bit Test Ws to SZ	1	1	SZ
		BTST.C	Ws,Wb	Bit Test Ws<Wb> to C	1	1	C
		BTST.Z	Ws,Wb	Bit Test Ws<Wb> to SZ	1	1	SZ
13	BTSTS	BTSTS.b	f,#bit3	Bit Test then Set f	1	1	SZ
		BTSTS.C	Ws,#bit4	Bit Test Ws to C, then Set	1	1	C
		BTSTS.Z	Ws,#bit4	Bit Test Ws to SZ, then Set	1	1	SZ
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 = $\overline{f}$	1	1	N,SZ
		COM	f,WREG	WREG = $\overline{f}$	1	1	N,SZ
		COM	Ws,Wd	Wd = $\overline{Ws}$	1	1	N,SZ
18	CP	CP	f	Compare f with WREG	1	1	C,DC,N,OV,SZ
		CP	Wb,#lit5	Compare Wb with lit5	1	1	C,DC,N,OV,SZ
		CP	Wb,Ws	Compare Wb with Ws (Wb - Ws)	1	1	C,DC,N,OV,SZ
19	CP0	CP0	f	Compare f with 0x0000	1	1	C,DC,N,OV,SZ
		CP0	Ws	Compare Ws with 0x0000	1	1	C,DC,N,OV,SZ
20	CP1	CP1	f	Compare f with 0xFFFF	1	1	C,DC,N,OV,SZ
		CP1	Ws	Compare Ws with 0xFFFF	1	1	C,DC,N,OV,SZ
21	CPB	CPB	f	Compare f with WREG, with Borrow	1	1	C,DC,N,OV,SZ
		CPB	Wb,#lit5	Compare Wb with lit5, with Borrow	1	1	C,DC,N,OV,SZ
		CPB	Wb,Ws	Compare Wb with Ws, with Borrow (Wb - Ws - $\overline{C}$)	1	1	C,DC,N,OV,SZ
22	CPSEQ	CPSEQ	f	Compare f with WREG, skip if =	1	1 (2 or 3)	None
23	CPSGT	CPSGT	f	Compare f with WREG, skip if >	1	1 (2 or 3)	None
24	CPSLT	CPSLT	f	Compare f with WREG, skip if <	1	1 (2 or 3)	None
25	CPSNE	CPSNE	f	Compare f with WREG, skip if ≠	1	1 (2 or 3)	None
26	DAW.b	DAW.b	Wn	Wn = decimal adjust Wn	1	1	C
27	DEC	DEC	f	f = f -1	1	1	C,DC,N,OV,SZ
		DEC	f,WREG	WREG = f -1	1	1	C,DC,N,OV,SZ
		DEC	Ws,Wd	Wd = Ws - 1	1	1	C,DC,N,OV,SZ
28	DEC2	DEC2	f	f = f -2	1	1	C,DC,N,OV,SZ
		DEC2	f,WREG	WREG = f -2	1	1	C,DC,N,OV,SZ
		DEC2	Ws,Wd	Wd = Ws - 2	1	1	C,DC,N,OV,SZ
29	DECSNZ	DECSNZ	f	f = f-1, Skip if Not 0	1	1 (2 or 3)	None
		DECSNZ	f,WREG	WREG = f-1, Skip if Not 0	1	1 (2 or 3)	None
30	DECSZ	DECSZ	f	f = f-1, Skip if 0	1	1 (2 or 3)	None
		DECSZ	f,WREG	WREG = f-1, Skip if 0	1	1 (2 or 3)	None
31	DISI	DISI	#lit14	Disable Interrupts for k instruction cycles	1	1	None

TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of words	# of cycles	Status Flags Affected
32	DIV	DIV.S Wm,Wn	Signed 16/16-bit Integer Divide	1	18	N,SZ,C
		DIV.SD Wm,Wn	Signed 32/16-bit Integer Divide	1	18	N,SZ,C
		DIV.U Wm,Wn	Unsigned 16/16-bit Integer Divide	1	18	N,SZ,C
		DIV.UD Wm,Wn	Unsigned 32/16-bit Integer Divide	1	18	N,SZ,C
33	DIVF	DIVF Wm,Wn	Signed 16/16-bit Fractional Divide	1	18	N,SZ,C
34	DO	DO #lit14,Expr	Do code to PC+Expr, lit14+1 times	2	2 + (lit14 + 1)*loop	None
		DO Wn,Expr	Do code to PC+Expr, (Wn)+1 times	2	2 + [(Wn) + 1]*loop	None
35	ED	ED Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean Distance	1	1	OA,OB
36	EDAC	EDAC Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean Distance Accumulate	1	1	OA,OB,SA,SB
37	EXCH	EXCH Wns,Wnd	Swap Wns with Wnd	1	1	None
38	FBCL	FBCL Ws,Wnd	Find Bit Change from Left (MSb) Side	1	1	C
39	FF1L	FF1L Ws,Wnd	Find First One from Left (MSb) Side	1	1	C
40	FF1R	FF1R Ws,Wnd	Find First One from Right (LSb) Side	1	1	C
41	GOTO	GOTO Expr	Go to address	2	2	None
		GOTO Wn	Go to indirect	1	2	None
42	INC	INC f	f = f + 1	1	1	C,DC,N,OV,SZ
		INC f,WREG	WREG = f + 1	1	1	C,DC,N,OV,SZ
		INC Ws,Wd	Wd = Ws + 1	1	1	C,DC,N,OV,SZ
43	INC2	INC2 f	f = f + 2	1	1	C,DC,N,OV,SZ
		INC2 f,WREG	WREG = f + 2	1	1	C,DC,N,OV,SZ
		INC2 Ws,Wd	Wd = Ws + 2	1	1	C,DC,N,OV,SZ
44	INCSNZ	INCSNZ f	f = f+1, Skip if Not 0	1	1 (2 or 3)	None
		INCSNZ f,WREG	WREG = f+1, Skip if Not 0	1	1 (2 or 3)	None
45	INCSZ	INCSZ f	f = f+1, Skip if 0	1	1 (2 or 3)	None
		INCSZ f,WREG	WREG = f+1, Skip if 0	1	1 (2 or 3)	None
46	IOR	IOR f	f = f .IOR. WREG	1	1	N,SZ
		IOR f,WREG	WREG = f .IOR. WREG	1	1	N,SZ
		IOR #lit10,Wn	Wd = lit10 .IOR. Wd	1	1	N,SZ
		IOR Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1	N,SZ
		IOR Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1	N,SZ
47	LAC	LAC Wso,#Slit4,Acc	Load Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
48	LNK	LNK #lit14	Link frame pointer	1	1	None
49	LSR	LSR f	f = Logical Right Shift f	1	1	C,N,OV,SZ
		LSR f,WREG	WREG = Logical Right Shift f	1	1	C,N,OV,SZ
		LSR Ws,Wd	Wd = Logical Right Shift Ws	1	1	C,N,OV,SZ
		LSR Wb,Wns,Wnd	Wnd = Logical Right Shift Wb by Wns	1	1	N,SZ
		LSR Wb,#lit5,Wnd	Wnd = Logical Right Shift Wb by lit5	1	1	N,SZ
50	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

TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax		Description	# of words	# of cycles	Status Flags Affected
51	MOV	MOV	f,Wn	Move f to Wn	1	1	None
		MOV	f	Move f to f	1	1	N,SZ
		MOV	f,WREG	Move f to WREG	1	1	N,SZ
		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,SZ
		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
52	MOVSAC	MOVSAC	Acc,Wx,Wxd,Wy,Wyd,AWB	Move Special	1	1	None
53	MPY	MPY	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wm by Wn to Accumulator	1	1	OA,OB,OAB
		MPY	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square Wm to Accumulator	1	1	OA,OB,OAB
54	MPY.N	MPY.N	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wm by Wn) to Accumulator	1	1	OA,OB,OAB
55	MSC	MSC	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd, AWB	Multiply and Subtract from Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
56	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
57	NEG	NEG	Acc	Negate Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
		NEG	f	f = $\overline{f}$ + 1	1	1	C,DC,N,OV,SZ
		NEG	f,WREG	WREG = $\overline{f}$ + 1	1	1	C,DC,N,OV,SZ
		NEG	Ws,Wd	Wd = $\overline{Ws}$ + 1	1	1	C,DC,N,OV,SZ
58	NOP	NOP		No Operation	1	1	None
		NOPR		No Operation	1	1	None
59	POP	POP	f	Pop f from top-of-stack (TOS)	1	1	None
		POP	Wdo	Pop from top-of-stack (TOS) to Wdo	1	1	None
		POP.D	Wnd	Pop from top-of-stack (TOS) to W(nd):W(nd+1)	1	2	None
		POP.S		Pop Shadow Registers	1	1	All
60	PUSH	PUSH	f	Push f to top-of-stack (TOS)	1	1	None
		PUSH	Wso	Push Wso to top-of-stack (TOS)	1	1	None
		PUSH.D	Wns	Push W(ns):W(ns+1) to top-of-stack (TOS)	1	2	None
		PUSH.S		Push Shadow Registers	1	1	None
61	PWRSAV	PWRSAV	#lit1	Go into Standby mode	1	1	WDTO,SLEEP
62	RCALL	RCALL	Expr	Relative Call	1	2	None
		RCALL	Wn	Computed Call	1	2	None
63	REPEAT	REPEAT	#lit14	Repeat Next Instruction lit14+1 times	1	2 + lit14	None
		REPEAT	Wn	Repeat Next Instruction (Wn)+1 times	1	2 + (Wn)	None
64	RESET	RESET		Software device RESET	1	1	None
65	RETFIE	RETFIE		Return from interrupt enable	1	3 (2)	None
66	RETLW	RETLW	#lit10,Wn	Return with literal in Wn	1	3 (2)	None
67	RETURN	RETURN		Return from Subroutine	1	3 (2)	None
68	RLC	RLC	f	f = Rotate Left through Carry f	1	1	C,N,SZ
		RLC	f,WREG	WREG = Rotate Left through Carry f	1	1	C,N,SZ
		RLC	Ws,Wd	Wd = Rotate Left through Carry Ws	1	1	C,N,SZ

TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of words	# of cycles	Status Flags Affected
69	RLNC	RLNC f	f = Rotate Left (No Carry) f	1	1	N,SZ
		RLNC f,WREG	WREG = Rotate Left (No Carry) f	1	1	N,SZ
		RLNC Ws,Wd	Wd = Rotate Left (No Carry) Ws	1	1	N,SZ
70	RRC	RRC f	f = Rotate Right through Carry f	1	1	C,N,SZ
		RRC f,WREG	WREG = Rotate Right through Carry f	1	1	C,N,SZ
		RRC Ws,Wd	Wd = Rotate Right through Carry Ws	1	1	C,N,SZ
71	RRNC	RRNC f	f = Rotate Right (No Carry) f	1	1	N,SZ
		RRNC f,WREG	WREG = Rotate Right (No Carry) f	1	1	N,SZ
		RRNC Ws,Wd	Wd = Rotate Right (No Carry) Ws	1	1	N,SZ
72	SAC	SAC Acc,#Slit4,Wdo	Store Accumulator	1	1	None
		SAC.R Acc,#Slit4,Wdo	Store Rounded Accumulator	1	1	None
73	SE	SE Ws,Wnd	Wnd = sign extended Ws	1	1	C,N,SZ
74	SETM	SETM f	f = 0xFFFF	1	1	None
		SETM WREG	WREG = 0xFFFF	1	1	None
		SETM Ws	Ws = 0xFFFF	1	1	None
75	SFTAC	SFTAC Acc,Wn	Arithmetic Shift Accumulator by (Wn)	1	1	OA,OB,OAB, SA,SB,SAB
		SFTAC Acc,#Slit5	Arithmetic Shift Accumulator by Slit5	1	1	OA,OB,OAB, SA,SB,SAB
76	SL	SL f	f = Left Shift f	1	1	C,N,OV,SZ
		SL f,WREG	WREG = Left Shift f	1	1	C,N,OV,SZ
		SL Ws,Wd	Wd = Left Shift Ws	1	1	C,N,OV,SZ
		SL Wb,Wns,Wnd	Wnd = Left Shift Wb by Wns	1	1	N,SZ
		SL Wb,#lit5,Wnd	Wnd = Left Shift Wb by lit5	1	1	N,SZ
77	SUB	SUB Acc	Subtract Accumulators	1	1	OA,OB,OAB, SA,SB,SAB
		SUB f	f = f - WREG	1	1	C,DC,N,OV,SZ
		SUB f,WREG	WREG = f - WREG	1	1	C,DC,N,OV,SZ
		SUB Wn,#lit10	Wn = Wn - lit10	1	1	C,DC,N,OV,SZ
		SUB Wb,Ws,Wd	Wd = Wb - Ws	1	1	C,DC,N,OV,SZ
		SUB Wb,#lit5,Wd	Wd = Wb - lit5	1	1	C,DC,N,OV,SZ
78	SUBB	SUBB f	f = f - WREG - ($\overline{\text{C}}$)	1	1	C,DC,N,OV,SZ
		SUBB f,WREG	WREG = f - WREG - ($\overline{\text{C}}$)	1	1	C,DC,N,OV,SZ
		SUBB Wn,#lit10	Wn = Wn - lit10 - ($\overline{\text{C}}$)	1	1	C,DC,N,OV,SZ
		SUBB Wb,Ws,Wd	Wd = Wb - Ws - ($\overline{\text{C}}$)	1	1	C,DC,N,OV,SZ
		SUBB Wb,#lit5,Wd	Wd = Wb - lit5 - ($\overline{\text{C}}$)	1	1	C,DC,N,OV,SZ
79	SUBR	SUBR f	f = WREG - f	1	1	C,DC,N,OV,SZ
		SUBR f,WREG	WREG = WREG - f	1	1	C,DC,N,OV,SZ
		SUBR Wb,Ws,Wd	Wd = Ws - Wb	1	1	C,DC,N,OV,SZ
		SUBR Wb,#lit5,Wd	Wd = lit5 - Wb	1	1	C,DC,N,OV,SZ
80	SUBBR	SUBBR f	f = WREG - f - ($\overline{\text{C}}$)	1	1	C,DC,N,OV,SZ
		SUBBR f,WREG	WREG = WREG -f - ($\overline{\text{C}}$)	1	1	C,DC,N,OV,SZ
		SUBBR Wb,Ws,Wd	Wd = Ws - Wb - ($\overline{\text{C}}$)	1	1	C,DC,N,OV,SZ
		SUBBR Wb,#lit5,Wd	Wd = lit5 - Wb - ($\overline{\text{C}}$)	1	1	C,DC,N,OV,SZ
81	SWAP	SWAP.b Wn	Wn = nibble swap Wn	1	1	None
		SWAP Wn	Wn = byte swap Wn	1	1	None
82	TBLRDH	TBLRDH Ws,Wd	Read Prog<23:16> to Wd<7:0>	1	2	None
83	TBLRDL	TBLRDL Ws,Wd	Read Prog<15:0> to Wd	1	2	None
84	TBLWTH	TBLWTH Ws,Wd	Write Ws<7:0> to Prog<23:16>	1	2	None
85	TBLWTL	TBLWTL Ws,Wd	Write Ws to Prog<15:0>	1	2	None
86	TRAP	TRAP #lit16	Trap to vector with literal	1	5	None
87	ULNK	ULNK	Unlink frame pointer	1	1	None

TABLE 25-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax		Description	# of words	# of cycles	Status Flags Affected
88	XOR	XOR	f	f = f .XOR. WREG	1	1	N,SZ
		XOR	f,WREG	WREG = f .XOR. WREG	1	1	N,SZ
		XOR	#lit10,Wn	Wd = lit10 .XOR. Wd	1	1	N,SZ
		XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1	N,SZ
		XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1	N,SZ
89	ZE	ZE	Ws,Wnd	Wnd = Zero Extend Ws	1	1	None

NOTES:

26.0 DEVELOPMENT SUPPORT

Microchip is offering a comprehensive package of development tools and libraries to support the dsPIC30F architecture. In addition, the company is partnering with many third party tools manufacturers for additional dsPIC30F support. The Microchip tools will include:

- MPLAB® 6.00 Integrated Development Environment (IDE)
- dsPIC30F Language Suite including MPLAB C30 C Compiler, Assembler, Linker and Librarian
- MPLAB SIM Software Simulator
- MPLAB ICE 4000 In-Circuit Emulator
- MPLAB ICD 2 In-Circuit Debugger
- PRO MATE® II Universal Device Programmer
- PICSTART® Plus Development Programmer

26.1 MPLAB V6.00 Integrated Development Environment Software

The MPLAB Integrated Development Environment is available at no cost. The IDE gives users the flexibility to edit, compile and emulate all from a single user interface. Engineers can design and develop code for the dsPIC30F in the same design environment that they have used for PICmicro microcontrollers.

The MPLAB IDE is a 32-bit Windows® based application. It provides many advanced features for the critical engineer in a modern, easy to use interface. MPLAB IDE integrates:

- Full featured, color coded text editor
- Easy to use project manager with visual display
- Source level debugging
- Enhanced source level debugging for ‘C’ (Structures, automatic variables, and so on)
- Customizable toolbar and key mapping
- Dynamic status bar displays processor condition at a glance
- Context sensitive, interactive on-line help
- Integrated MPLAB SIM instruction simulator
- User interface for PRO MATE II and PICSTART Plus device programmers (sold separately)
- User interface for MPLAB ICE 4000 In-Circuit Emulator (sold separately)
- User interface for MPLAB ICD 2 In-Circuit Debugger

The MPLAB IDE allows the engineer to:

- Edit your source files in either assembly or ‘C’
- One-touch compile and download to dsPIC30F program memory on emulator or simulator. Updates all project information.
- Debug using:
  - Source files
  - Machine code
  - Mixed mode source and machine code

The ability to use the MPLAB IDE with multiple development and debugging targets, allows users to easily switch from the cost effective simulator to a full featured emulator with minimal retraining.

26.2 dsPIC30F Language Suite

The Microchip Technology MPLAB C30 C compiler is a fully ANSI compliant product with standard libraries for the dsPIC30F architecture. It is highly optimizing and takes advantage of many dsPIC30F architecture specific features to provide efficient software code generation.

MPLAB C30 also provides extensions that allow for excellent support of the hardware, such as interrupts and peripherals. It is fully integrated with the MPLAB IDE for high level, source debugging.

- 16-bit native data types
- Efficient use of register based, 3-operand instructions
- Complex Addressing modes
- Efficient multi-bit shift operations
- Efficient signed/unsigned comparisons

MPLAB C30 comes complete with its own assembler, linker and librarian. These allow the user to write Mixed mode C and assembly programs and link the resulting object files into a single executable file. The compiler is sold separately. The assembler, linker and librarian are available for free with MPLAB IDE.

26.3 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC-hosted environment by simulating the dsPIC30F on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any of the pins.

The execution can be performed in single step, execute until break, or Trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C30 compiler and assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent multi-project software development tool.

26.4 MPLAB ICE 4000 In-Circuit Emulator

The MPLAB ICE 4000 In-Circuit Emulator is intended to provide the product development engineer with a complete hardware design tool for the dsPIC30F. Software control of the emulator is provided by MPLAB, allowing editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 4000 is a full featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The MPLAB ICE 4000 supports the extended, high-end PICmicro microcontrollers, the PIC18CXXX and PIC18FXXX devices, as well as the dsPIC30F family of digital signal controllers. The modular architecture of the MPLAB ICE in-circuit emulator allows expansion to support new devices.

The MPLAB ICE in-circuit emulator system has been designed as a real-time emulation system, with advanced features that are generally found on more expensive development tools.

- Full speed emulation, up to 50 MHz bus speed, or 200 MHz external clock speed
- Low voltage emulation down to 1.8 volts
- Configured with 2 Mb program emulation memory, additional modular memory up to 16 Mb
- 64K x 248-bit wide Trace Memory
- Unlimited software breakpoints
- Complex break, trace and trigger logic
- Multi-level trigger up to 4 levels
- Filter trigger functions to trace specific event
- 16-bit Pass counter for triggering on sequential events
- 16-bit Delay counter
- 48-bit time stamp
- Stopwatch feature
- Time between events
- Statistical performance analysis
- Code coverage analysis
- USB and parallel printer port PC connection

26.5 MPLAB ICD 2 In-Circuit Debugger

Microchip’s In-Circuit Debugger, MPLAB ICD, is a powerful, low cost, run-time development tool. This tool is based on the PICmicro and dsPIC30F FLASH devices. The MPLAB ICD utilizes the in-circuit debugging capability built into the various devices. This feature, along with Microchip’s In-Circuit Serial Programming^{TM} protocol, offers cost effective, in-circuit debugging from the graphical user interface of MPLAB ICD. This enables a designer to develop and debug source code by watching variables, single-stepping and setting break points. Running at full speed enables testing hardware in real-time.

- Full speed operation to the range of the device
- Serial or USB PC connector
- Serial interface externally powered
- USB powered from PC interface
- Low noise power (V_{PP} and V_{DD}) for use with analog and other noise sensitive applications
- Operation down to 2.0V
- Can be used as an ICD or inexpensive serial programmer
- Modular application connector as MPLAB ICD
- Limited number of breakpoints
- “Smart watch” variable windows
- Some chip resources required (RAM, program memory and 2 pins)

26.6 PRO MATE II Universal Device Programmer

The PRO MATE II universal device programmer is a full featured programmer, capable of operating in Stand-Alone mode, as well as PC-hosted mode. The PRO MATE II device programmer is CE compliant.

The PRO MATE II device programmer has programmable V_{DD} and V_{PP} supplies, which allow it to verify programmed memory at V_{DDMIN} and V_{DDMAX} for maximum reliability when programming, requiring this capability. It has an LCD display for instructions and error messages, keys to enter commands, and interchangeable socket modules for all package types.

In Stand-Alone mode, the PRO MATE II device programmer can read, verify, or program PICmicro devices. It can also set code protection in this mode.

- Runs under MPLAB IDE
- Field upgradable firmware
- DOS Command Line interface for production
- Host, Safe, and “Stand-Alone” operation
- Automatic downloading of object file
- SQTP^{SM} serialization adds unique serial number to each device programmed
- In-Circuit Serial Programming Kit (sold separately)
- Interchangeable socket modules supports all package options (sold separately)

27.0 ELECTRICAL CHARACTERISTICS

Electrical characteristics are not available at this time.

NOTES:

28.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

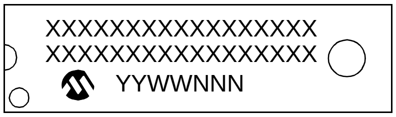
Graphs and tables are not available at this time.

NOTES:

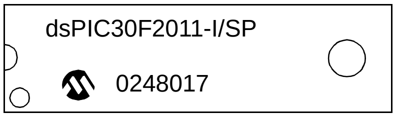
29.0 PACKAGING INFORMATION

29.1 Package Marking Information

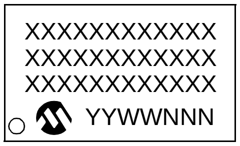
18-Lead PDIP (Skinny DIP)



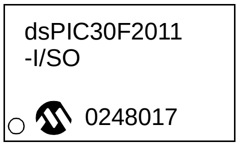
Example



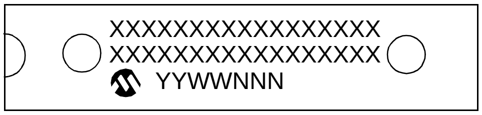
18-Lead SOIC



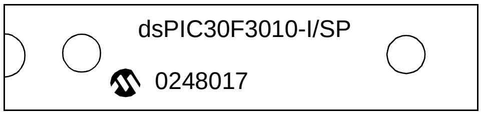
Example



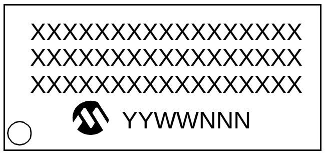
28-Lead PDIP (Skinny DIP)



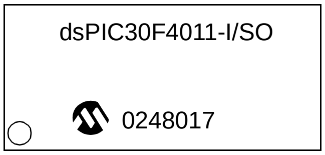
Example



28-Lead SOIC



Example



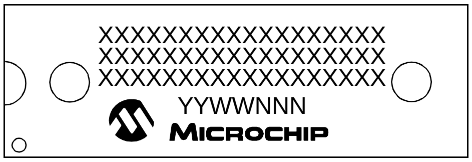
Legend:	XX...X	Customer specific information*
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code

Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

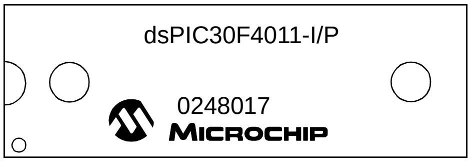
* Standard PICmicro device marking consists of Microchip part number, year code, week code, and traceability code. For PICmicro device marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

29.2 Package Marking Information (Continued)

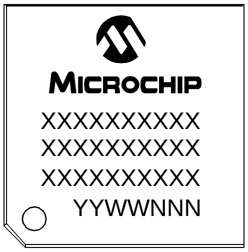
40-Lead PDIP



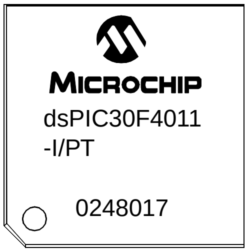
Example



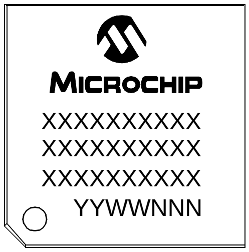
44-Lead TQFP



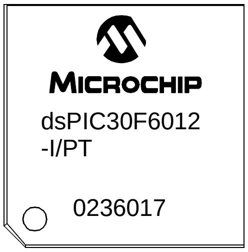
Example



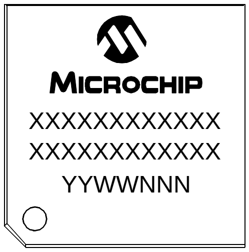
64-Lead TQFP



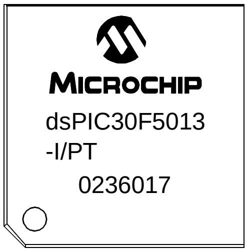
Example



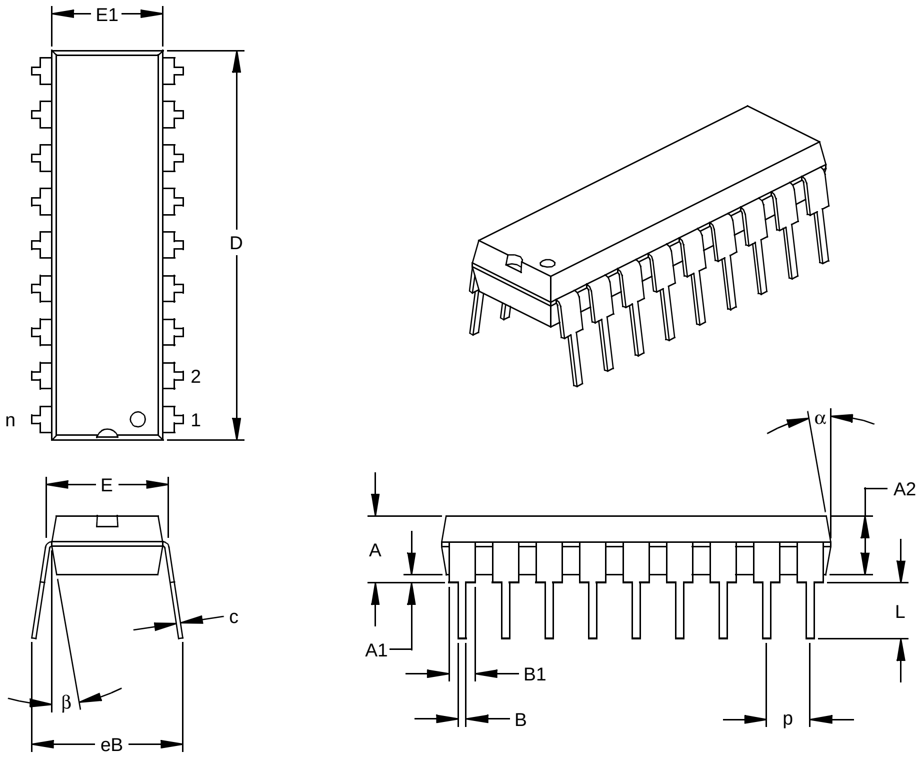
80-Lead TQFP



Example



18-Lead Plastic Dual In-line (P) – 300 mil (PDIP)

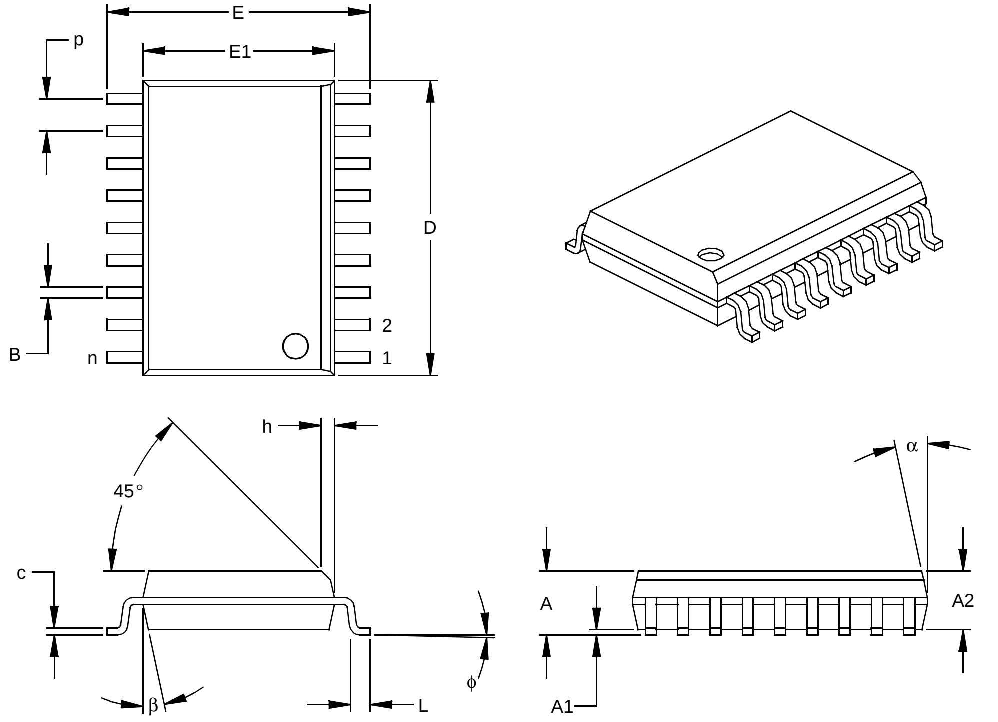


	Units	INCHES*			MILLIMETERS		
	Dimension Limits	MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.140	.155	.170	3.56	3.94	4.32
Molded Package Thickness	A2	.115	.130	.145	2.92	3.30	3.68
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.300	.313	.325	7.62	7.94	8.26
Molded Package Width	E1	.240	.250	.260	6.10	6.35	6.60
Overall Length	D	.890	.898	.905	22.61	22.80	22.99
Tip to Seating Plane	L	.125	.130	.135	3.18	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.045	.058	.070	1.14	1.46	1.78
Lower Lead Width	B	.014	.018	.022	0.36	0.46	0.56
Overall Row Spacing §	eB	.310	.370	.430	7.87	9.40	10.92
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-001
Drawing No. C04-007

18-Lead Plastic Small Outline (SO) – Wide, 300 mil (SOIC)

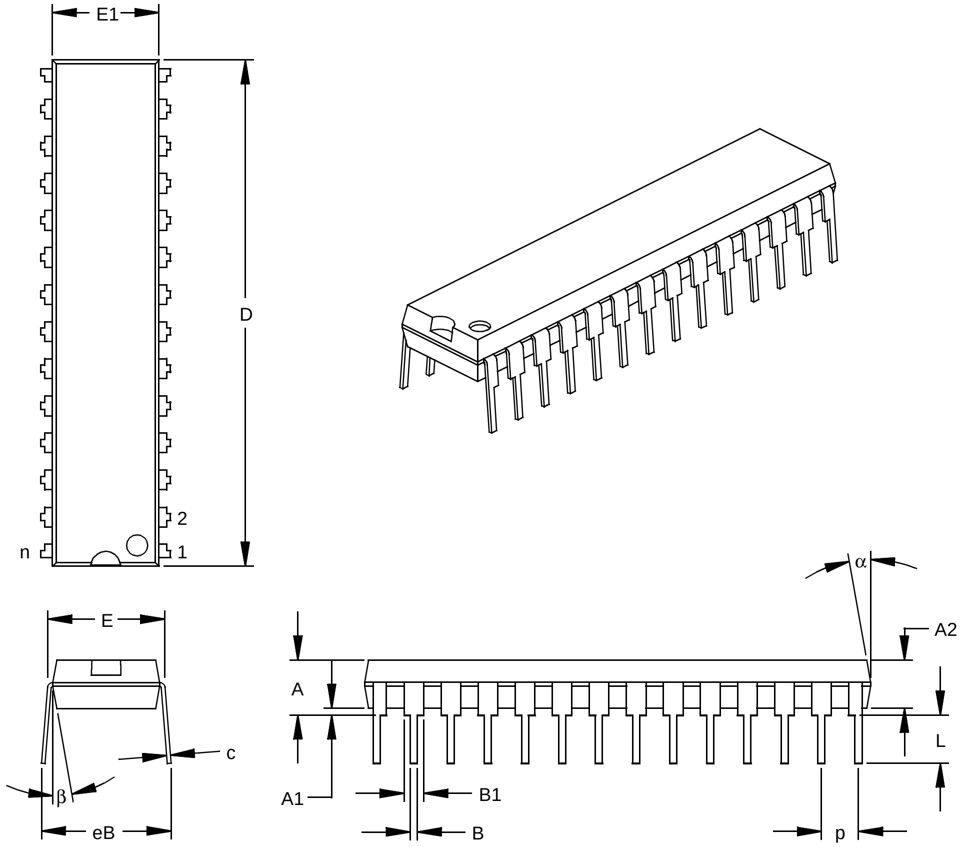


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.050			1.27	
Overall Height	A	.093	.099	.104	2.36	2.50	2.64
Molded Package Thickness	A2	.088	.091	.094	2.24	2.31	2.39
Standoff §	A1	.004	.008	.012	0.10	0.20	0.30
Overall Width	E	.394	.407	.420	10.01	10.34	10.67
Molded Package Width	E1	.291	.295	.299	7.39	7.49	7.59
Overall Length	D	.446	.454	.462	11.33	11.53	11.73
Chamfer Distance	h	.010	.020	.029	0.25	0.50	0.74
Foot Length	L	.016	.033	.050	0.41	0.84	1.27
Foot Angle	φ	0	4	8	0	4	8
Lead Thickness	c	.009	.011	.012	0.23	0.27	0.30
Lead Width	B	.014	.017	.020	0.36	0.42	0.51
Mold Draft Angle Top	α	0	12	15	0	12	15
Mold Draft Angle Bottom	β	0	12	15	0	12	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-013
Drawing No. C04-051

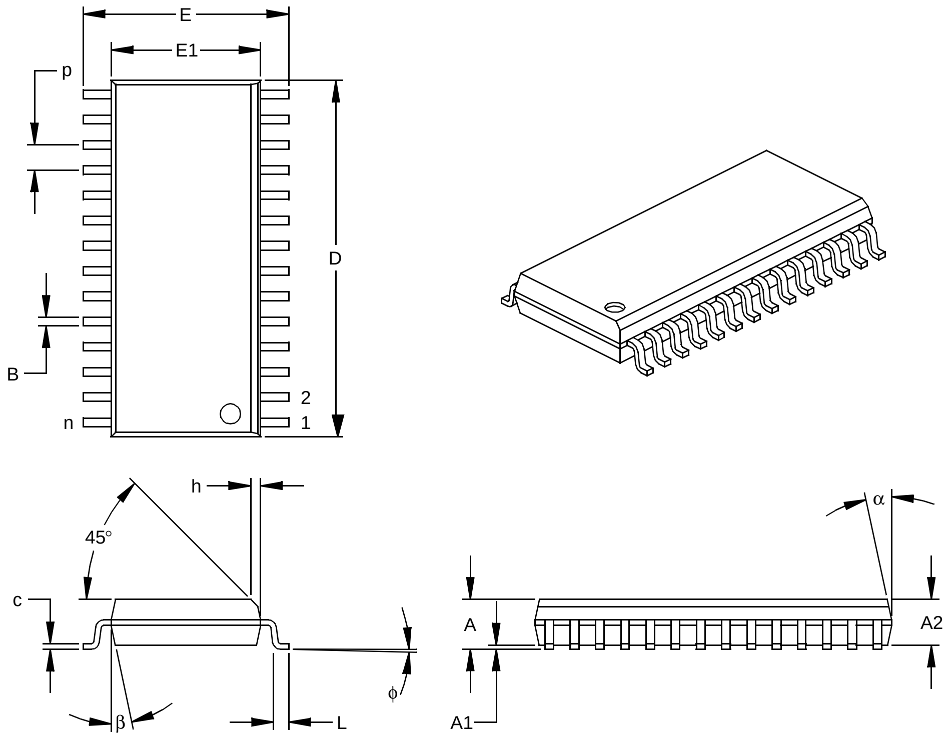
28-Lead Skinny Plastic Dual In-line (SP) – 300 mil (PDIP)



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.140	.150	.160	3.56	3.81	4.06
Molded Package Thickness	A2	.125	.130	.135	3.18	3.30	3.43
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.300	.310	.325	7.62	7.87	8.26
Molded Package Width	E1	.275	.285	.295	6.99	7.24	7.49
Overall Length	D	1.345	1.365	1.385	34.16	34.67	35.18
Tip to Seating Plane	L	.125	.130	.135	3.18	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.040	.053	.065	1.02	1.33	1.65
Lower Lead Width	B	.016	.019	.022	0.41	0.48	0.56
Overall Row Spacing §	eB	.320	.350	.430	8.13	8.89	10.92
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimension D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MO-095
Drawing No. C04-070

28-Lead Plastic Small Outline (SO) – Wide, 300 mil (SOIC)

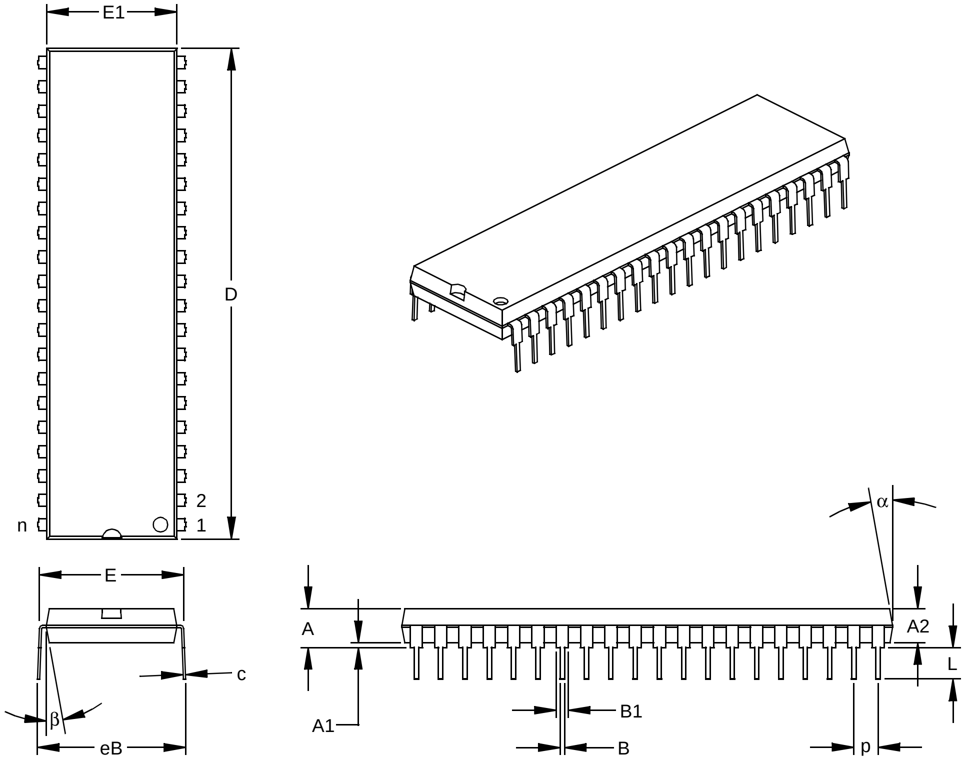


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.050			1.27	
Overall Height	A	.093	.099	.104	2.36	2.50	2.64
Molded Package Thickness	A2	.088	.091	.094	2.24	2.31	2.39
Standoff §	A1	.004	.008	.012	0.10	0.20	0.30
Overall Width	E	.394	.407	.420	10.01	10.34	10.67
Molded Package Width	E1	.288	.295	.299	7.32	7.49	7.59
Overall Length	D	.695	.704	.712	17.65	17.87	18.08
Chamfer Distance	h	.010	.020	.029	0.25	0.50	0.74
Foot Length	L	.016	.033	.050	0.41	0.84	1.27
Foot Angle Top	φ	0	4	8	0	4	8
Lead Thickness	c	.009	.011	.013	0.23	0.28	0.33
Lead Width	B	.014	.017	.020	0.36	0.42	0.51
Mold Draft Angle Top	α	0	12	15	0	12	15
Mold Draft Angle Bottom	β	0	12	15	0	12	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-013
Drawing No. C04-052

40-Lead Plastic Dual In-line (P) – 600 mil (PDIP)

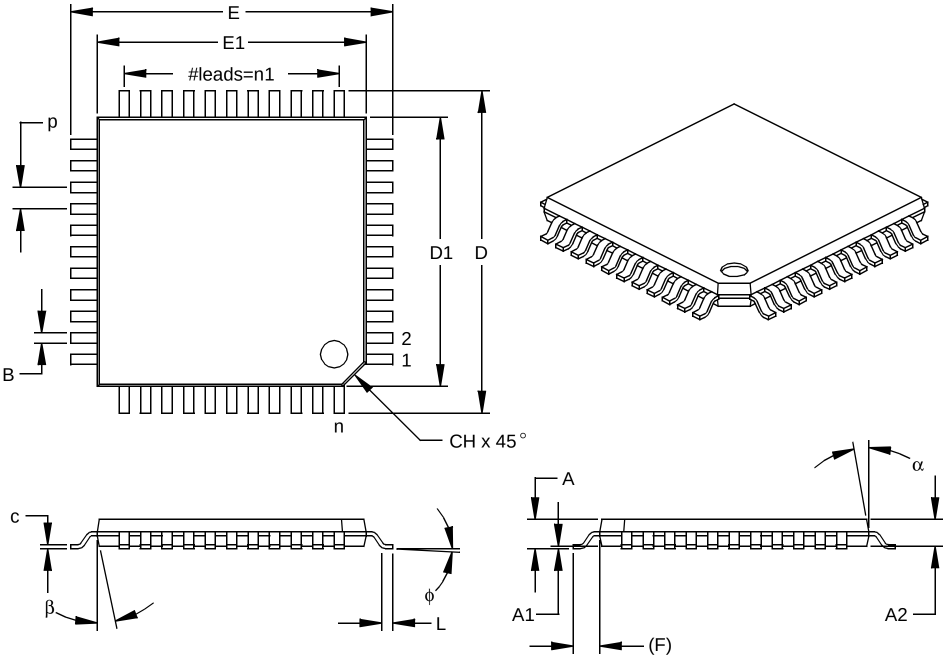


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		40			40	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.160	.175	.190	4.06	4.45	4.83
Molded Package Thickness	A2	.140	.150	.160	3.56	3.81	4.06
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.595	.600	.625	15.11	15.24	15.88
Molded Package Width	E1	.530	.545	.560	13.46	13.84	14.22
Overall Length	D	2.045	2.058	2.065	51.94	52.26	52.45
Tip to Seating Plane	L	.120	.130	.135	3.05	3.30	3.43
Lead Thickness	c	.008	.012	.015	0.20	0.29	0.38
Upper Lead Width	B1	.030	.050	.070	0.76	1.27	1.78
Lower Lead Width	B	.014	.018	.022	0.36	0.46	0.56
Overall Row Spacing §	eB	.620	.650	.680	15.75	16.51	17.27
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MO-011
Drawing No. C04-016

44-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

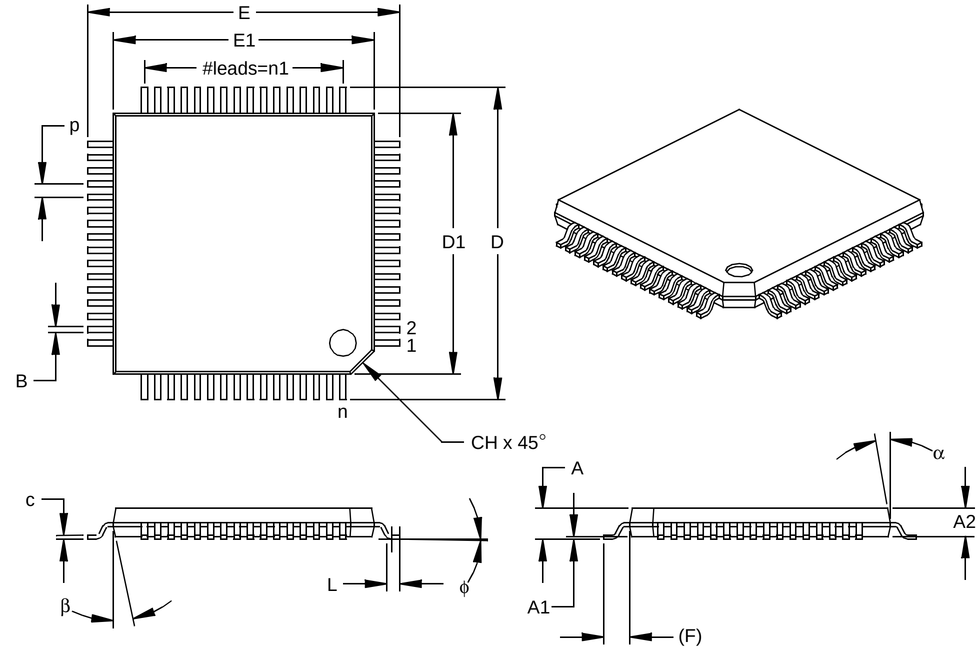


Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		44			44	
Pitch	p		.031			0.80	
Pins per Side	n1		11			11	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.004	.006	0.05	0.10	0.15
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039		1.00		
Foot Angle	φ	0	3.5	7	0	3.5	7
Overall Width	E	.463	.472	.482	11.75	12.00	12.25
Overall Length	D	.463	.472	.482	11.75	12.00	12.25
Molded Package Width	E1	.390	.394	.398	9.90	10.00	10.10
Molded Package Length	D1	.390	.394	.398	9.90	10.00	10.10
Lead Thickness	c	.004	.006	.008	0.09	0.15	0.20
Lead Width	B	.012	.015	.017	0.30	0.38	0.44
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-076

64-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

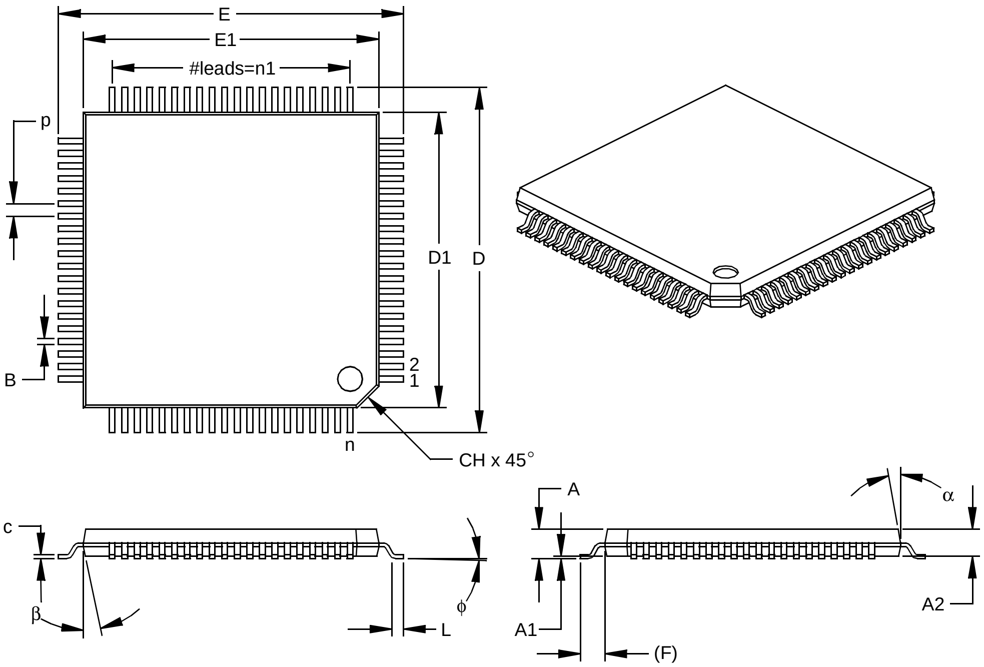


Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		64			64	
Pitch	p		.020			0.50	
Pins per Side	n1		16			16	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.006	.010	0.05	0.15	0.25
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039			1.00	
Foot Angle	φ	0	3.5	7	0	3.5	7
Overall Width	E	.463	.472	.482	11.75	12.00	12.25
Overall Length	D	.463	.472	.482	11.75	12.00	12.25
Molded Package Width	E1	.390	.394	.398	9.90	10.00	10.10
Molded Package Length	D1	.390	.394	.398	9.90	10.00	10.10
Lead Thickness	c	.005	.007	.009	0.13	0.18	0.23
Lead Width	B	.007	.009	.011	0.17	0.22	0.27
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-085

80-Lead Plastic Thin Quad Flatpack (PT) 12x12x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)



Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		80			80	
Pitch	p		.020			0.50	
Pins per Side	n1		20			20	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.004	.006	0.05	0.10	0.15
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039			1.00	
Foot Angle	ϕ	0	3.5	7	0	3.5	7
Overall Width	E	.541	.551	.561	13.75	14.00	14.25
Overall Length	D	.541	.551	.561	13.75	14.00	14.25
Molded Package Width	E1	.463	.472	.482	11.75	12.00	12.25
Molded Package Length	D1	.463	.472	.482	11.75	12.00	12.25
Lead Thickness	c	.004	.006	.008	0.09	0.15	0.20
Lead Width	B	.007	.009	.011	0.17	0.22	0.27
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-092

INDEX

NOTES:

ON-LINE SUPPORT

Microchip provides on-line support on the Microchip World Wide Web (WWW) site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape or Microsoft Explorer. Files are also available for FTP download from our FTP site.

Connecting to the Microchip Internet Web Site

The Microchip web site is available by using your favorite Internet browser to attach to:

www.microchip.com

The file transfer site is available by using an FTP service to connect to:

ftp://ftp.microchip.com

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events

Systems Information and Upgrade Hot Line

The Systems Information and Upgrade Line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive any currently available upgrade kits.The Hot Line Numbers are:

1-800-755-2345 for U.S. and most of Canada, and

1-480-792-7302 for the rest of the world.

013001

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this Data Sheet.

To: Technical Publications Manager Total Pages Sent

RE: Reader Response

From: Name ____________________

Company ____________________

Address ____________________

City / State / ZIP / Country ____________________

Telephone: (_______) _________ - _________ FAX: (______) _________ - _________

Application (optional):

Would you like a reply?___Y ___N

Device: **dsPIC30F** Literature Number: **DS70032B**

Questions:

1. What are the best features of this document?

____

____

2. How does this document meet your hardware and software development needs?

____

____

3. Do you find the organization of this data sheet easy to follow? If not, why?

____

____

4. What additions to the data sheet do you think would enhance the structure and subject?

____

____

5. What deletions from the data sheet could be made without affecting the overall usefulness?

____

____

6. Is there any incorrect or misleading information (what and where)?

____

____

7. How would you improve this document?

____

____

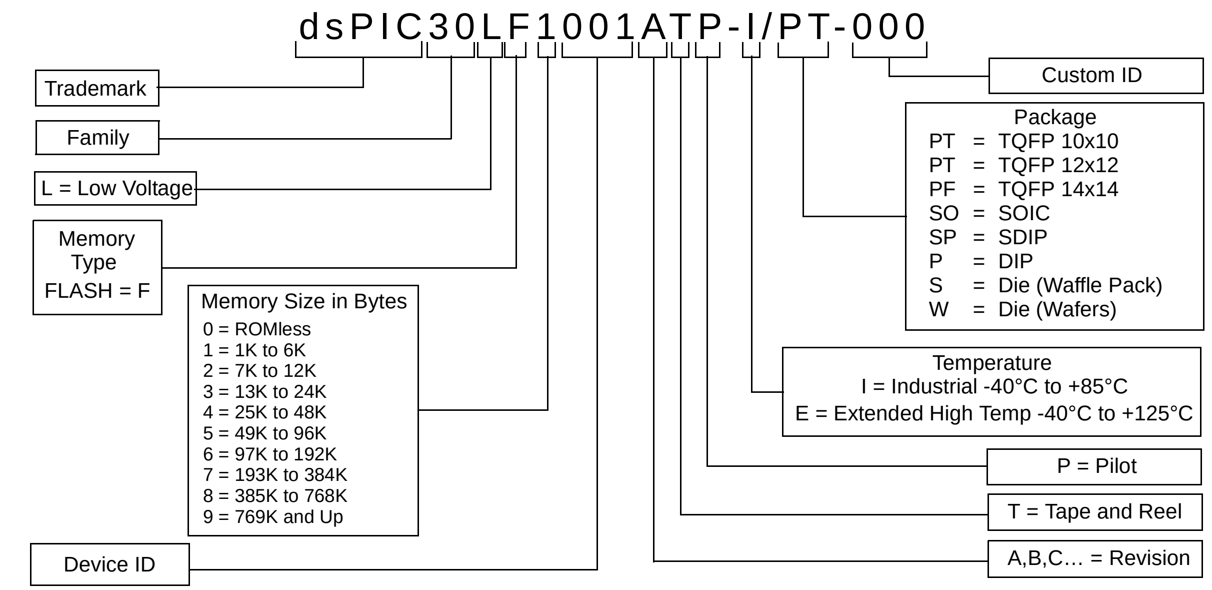
8. How would you improve our software, systems, and silicon products?

____

____

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



dsPIC30LF1001ATP-I/PT-000

Trademark

Family

L = Low Voltage

Memory Type
FLASH = F

Memory Size in Bytes

0	= ROMless
1	= 1K to 6K
2	= 7K to 12K
3	= 13K to 24K
4	= 25K to 48K
5	= 49K to 96K
6	= 97K to 192K
7	= 193K to 384K
8	= 385K to 768K
9	= 769K and Up

Device ID

Custom ID

Package

PT	=	TQFP 10x10
PT	=	TQFP 12x12
PF	=	TQFP 14x14
SO	=	SOIC
SP	=	SDIP
P	=	DIP
S	=	Die (Waffle Pack)
W	=	Die (Wafers)

Temperature
I = Industrial -40°C to +85°C
E = Extended High Temp -40°C to +125°C

P = Pilot

T = Tape and Reel

A,B,C… = Revision

Examples:

a) dsPIC30F2011ATP-E/SO = Extended temp., SOIC package, Rev. A.

b) dsPIC30F6014ATP-I/PT = Industrial temp., TQFP package, Rev. A.

c) dsPIC30F3011ATP-I/P = Industrial temp., PDIP package, Rev. A.

* JW Devices are UV erasable and can be programmed to any device configuration. JW Devices meet the electrical requirement of each oscillator type.

Sales and Support

Data Sheets

Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office
2. The Microchip Corporate Literature Center U.S. FAX: (480) 792-7277
3. The Microchip Worldwide Site (www.microchip.com)

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

New Customer Notification System

Register on our web site (www.microchip.com/cn) to receive the most current information on our products.

MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain

2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta

500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Boston

2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Chicago

333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas

4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Detroit

Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Kokomo

2767 S. Albright Road
Kokomo, Indiana 46902
Tel: 765-864-8360 Fax: 765-864-8387

Los Angeles

18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

New York

150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose

Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto

6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia

Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing

Microchip Technology Consulting (Shanghai)
Co., Ltd., Beijing Liaison Office
Unit 915
Bei Hai Wan Tai Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Chengdu

Microchip Technology Consulting (Shanghai)
Co., Ltd., Chengdu Liaison Office
Rm. 2401, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-6766200 Fax: 86-28-6766599

China - Fuzhou

Microchip Technology Consulting (Shanghai)
Co., Ltd., Fuzhou Liaison Office
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506 Fax: 86-591-7503521

China - Shanghai

Microchip Technology Consulting (Shanghai)
Co., Ltd.
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

China - Shenzhen

Microchip Technology Consulting (Shanghai)
Co., Ltd., Shenzhen Liaison Office
Rm. 1315, 13/F, Shenzhen Kerry Centre,
Renminnan Lu
Shenzhen 518001, China
Tel: 86-755-2350361 Fax: 86-755-2366086

Hong Kong

Microchip Technology Hongkong Ltd.
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India

Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan

Microchip Technology Japan K.K.
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea

Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5934

Singapore

Microchip Technology Singapore Pte Ltd.
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-334-8870 Fax: 65-334-8850

Taiwan

Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark

Microchip Technology Nordic ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France

Microchip Technology SARL
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

Germany

Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy

Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

United Kingdom

Arizona Microchip Technology Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

01/18/02