dsPIC30F Data Sheet Motor Control and Power Conversion Family

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.

Printed on recycled paper.

DNV Certification, Inc. USA
ANSI • RAB QMS ACCREDITED
DNV MSC The Netherlands Accredited by the RvA
DNV
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 Digital Signal Controllers Motor Control and Power Conversion Family

High Performance Modified RISC CPU:

- Modified Harvard architecture
- C compiler optimized instruction set architecture
- 88 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
- 17-bit x 17-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 42 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
- 3-wire SPI™ modules (supports 4 Frame modes)
- I^2C™ module supports Multi-Master/Slave mode and 7-bit/10-bit addressing

Peripheral Features (Continued):

- 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:

- 10-bit Analog-to-Digital Converter (A/D) with:
  - 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
  - 10,000 erase/write cycle (typical) for industrial temperature range
- Data EEPROM memory
  - 100,000 erase/write cycle (typical) industrial temperature range
- 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

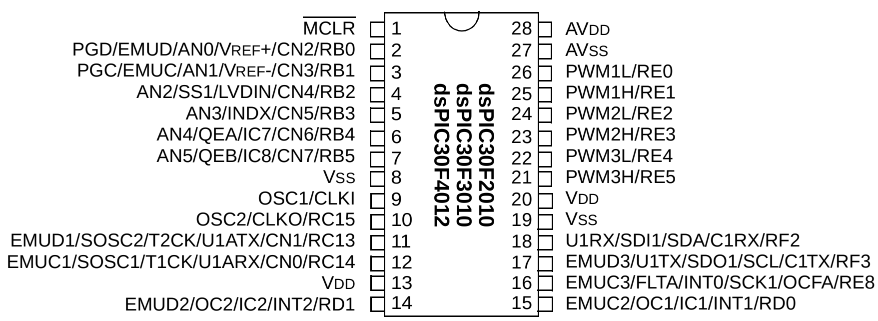
dsPIC30F Motor Control and Power Conversion Family

Device	Pins	Program Mem. Bytes/ Instructions	SRAM Bytes	EEPROM Bytes	Timer 16-bit	Input Cap	Output Comp/Std PWM	Motor Control PWM	A/D 10-bit 500 Ksps	Quad Enc	UART	SPI™	I^2C™	CAN
dsPIC30F2010	28	12K/4K	512	1024	3	4	2	6 ch	6 ch	Yes	1	1	1	-
dsPIC30F3010	28	24K/8K	1024	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	-
dsPIC30F4012	28	48K/16K	2048	1024	5	4	2	6 ch	6 ch	Yes	1	1	1	1
dsPIC30F3011	40/44	24K/8K	1024	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	-
dsPIC30F4011	40/44	48K/16K	2048	1024	5	4	4	6 ch	9 ch	Yes	2	1	1	1
dsPIC30F5015*	64	66K/22K	2048	1024	5	4	4	8 ch	16 ch	Yes	1	2	1	1
dsPIC30F6010	80	144K/48K	8192	4096	5	8	8	8 ch	16 ch	Yes	2	2	1	2

* Pinout for dsPIC30F5015 will be provided at a later date.

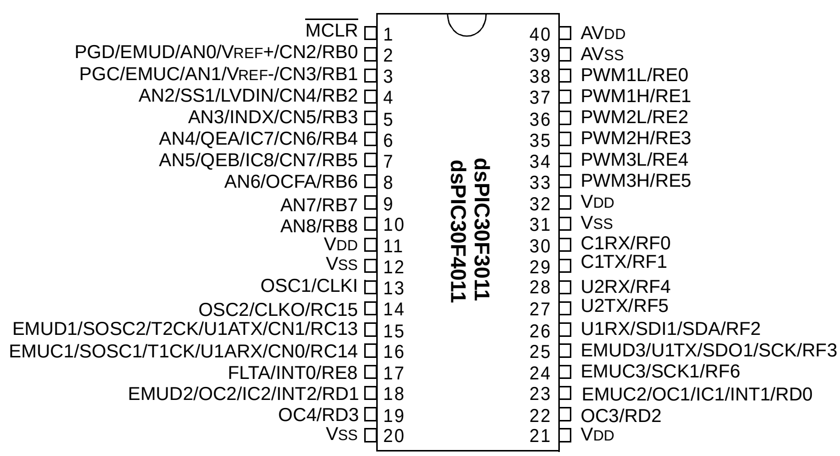
Pin Diagrams

28-Pin SDIP



Note: Pinout subject to change.

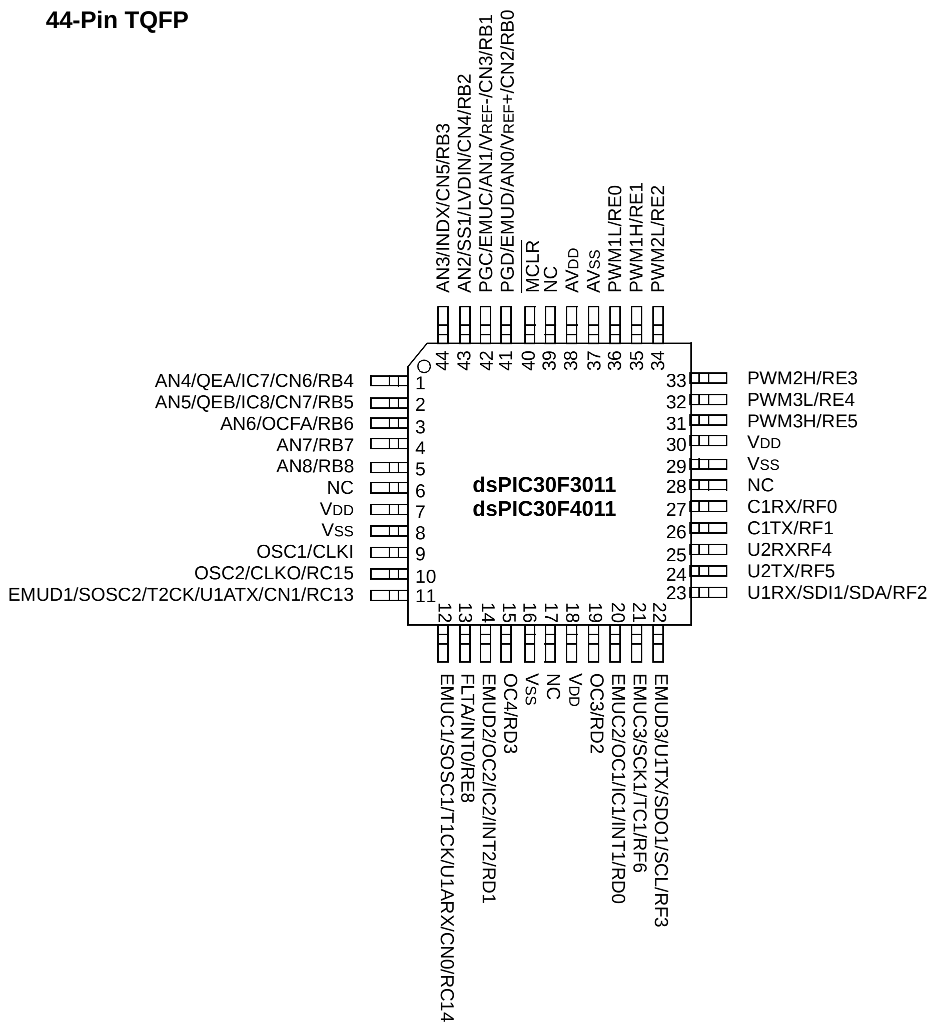
40-Pin PDIP



Note: Pinout subject to change.

Pin Diagrams (Continued)

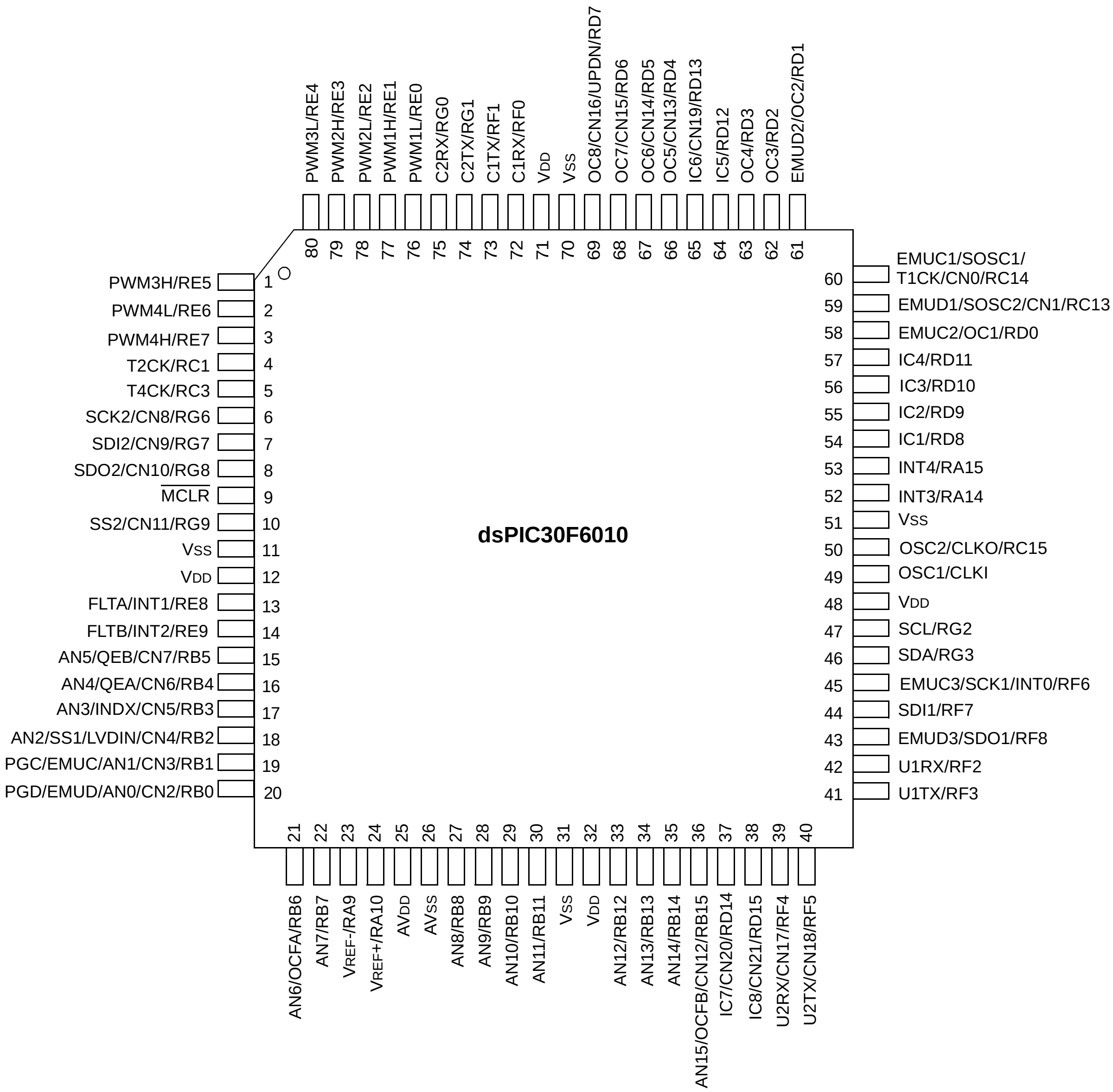
44-Pin TQFP



Note: Pinout subject to change.

Pin Diagrams (Continued)

80-Pin TQFP



Note: Pinout subject to change.

Table of Contents

1.0 Device Overview ... 9
2.0 Core Architecture Overview ... 13
3.0 Memory Organization ... 25
4.0 Address Generator Units ... 37
5.0 Exception Processing ... 45
6.0 FLASH Program Memory ... 51
7.0 Data EEPROM Memory ... 57
8.0 I/O Ports ... 61
9.0 Timer1 Module ... 67
10.0 Timer2/3 Module ... 71
11.0 Timer4/5 Module ... 77
12.0 Input Capture Module ... 81
13.0 Output Compare Module ... 85
14.0 Quadrature Encoder Interface (QEI) Module ... 89
15.0 Motor Control PWM Module ... 95
16.0 SPI Module ... 107
17.0 I2C Module ... 111
18.0 Universal Asynchronous Receiver Transmitter (UART) Module ... 119
19.0 CAN Module ... 127
20.0 10-bit High speed Analog-to-Digital Converter (A/D) Module ... 133
21.0 System Integration ... 141
22.0 Instruction Set Summary ... 153
23.0 Development Support ... 161
24.0 Electrical Characteristics ... 163
25.0 DC and AC Characteristics Graphs and Tables ... 165
26.0 Packaging Information ... 167
Index ... 175
On-Line Support ... 181
Reader Response ... 182
Product Identification System ... 183

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.

Figure 1-1 shows a sample device block diagram.

Note:	The device(s) depicted in this block diagram are representative of the corresponding device family. Other devices of the same family may vary in terms of number of pins and multiplexing of pin functions. Typically, smaller devices in the family contain a subset of the peripherals present in the device(s) shown in this diagram.

FIGURE 1-1: dsPIC30F6010 BLOCK DIAGRAM

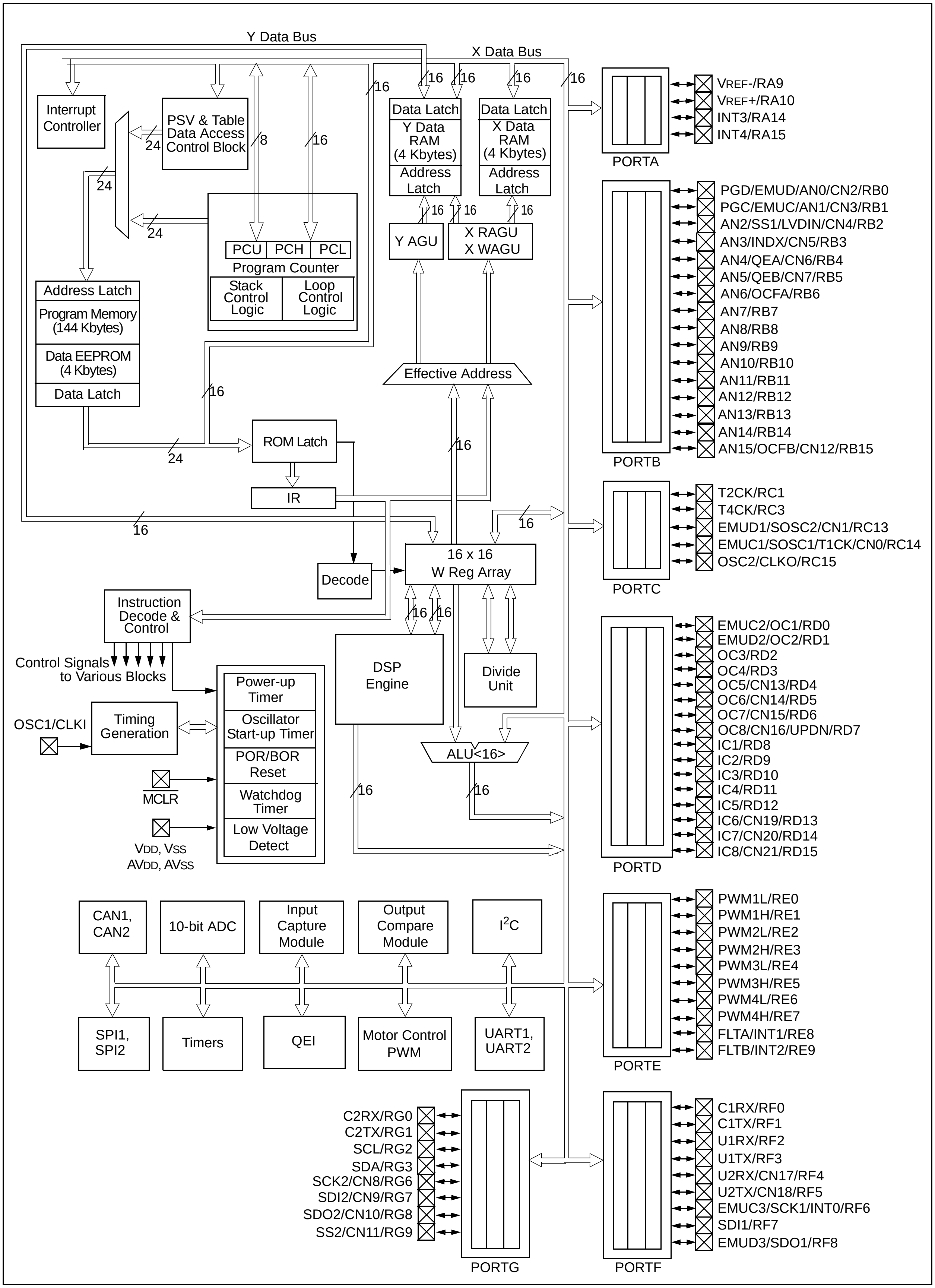


Table 1-1 provides a brief description of device I/O pinouts and the functions that may be multiplexed to a port pin. Multiple functions may exist on one port pin. When multiplexing occurs, the peripheral module's functional requirements may force an override of the data direction of the port pin.

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.
CLKI CLKO	I O	ST/CMOS —	External clock source input. Always associated with OSC1 pin function. Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. Optionally functions as CLKO in RC and EC modes. Always associated with OSC2 pin function.
CN0 - CN23	I	ST	Input change notification inputs. Can be software programmed for internal weak pull-ups on all inputs.
COFS CSCK CSDI CSDO	I/O I/O I O	ST ST ST —	Data Converter Interface frame synchronization pin. Data Converter Interface serial clock input/output pin. Data Converter Interface serial data input pin. Data Converter Interface serial data output pin.
C1RX C1TX C2RX C2TX	I O I O	ST — ST —	CAN1 bus receive pin. CAN1 bus transmit pin. CAN2 bus receive pin. CAN2 bus transmit pin.
EMUD EMUC EMUD1 EMUC1 EMUD2 EMUC2 EMUD3 EMUC3	I/O I I/O I I/O I I/O I	ST ST ST ST ST ST ST ST	ICD Primary Communication Channel data input/output pin. ICD Primary Communication Channel clock input pin. ICD Secondary Communication Channel data input/output pin. ICD Secondary Communication Channel clock input pin. ICD Tertiary Communication Channel data input/output pin. ICD Tertiary Communication Channel clock input pin. ICD Quaternary Communication Channel data input/output pin. ICD Quaternary Communication Channel clock input pin.
IC1 - IC8	I	ST	Capture inputs 1 through 8.
INDX QEA QEB UPDN	I I I O	ST ST ST CMOS	Quadrature Encoder Index Pulse input. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode. Quadrature Encoder Phase A input in QEI mode. Auxiliary Timer External Clock/Gate input in Timer mode. Position Up/Down Counter Direction State.
INT0 INT1 INT2 INT3 INT4	I I I I I	ST ST ST ST ST	External interrupt 0. External interrupt 1. External interrupt 2. External interrupt 3. External interrupt 4.
LVDIN	I	Analog	Low Voltage Detect Reference Voltage input pin.
FLTA FLTB PWM1L PWM1H PWM2L PWM2H PWM3L PWM3H PWM4L PWM4H	I I O O O O O O O O	ST ST — — — — — — — —	PWM Fault A input. PWM Fault B input. PWM 1 Low output. PWM 1 High output. PWM 2 Low output. PWM 2 High output. PWM 3 Low output. PWM 3 High output. PWM 4 Low output. PWM 4 High output.

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

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

Pin Name	Pin Type	Buffer Type	Description
$\overline{MCLR}$	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
OCFA OCFB OC1 - OC8	I I 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). Compare outputs 1 through 8.
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 CLKO in RC and EC modes.
PGD PGC	I/O I	ST ST	In-Circuit Serial Programming data input/output pin. In-Circuit Serial Programming clock input pin.
RA9 - RA10 RA14 - 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.
RC1 RC3 RC13 - RC15	I/O I/O I/O	ST 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	I/O	ST	PORTF is a bi-directional I/O port.
RG0 - RG3 RG6 - RG9	I/O I/O	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.
T1CK T2CK T3CK T4CK T5CK	I I I I I	ST ST ST ST ST	Timer1 external clock input. Timer2 external clock input. Timer3 external clock input. Timer4 external clock input. Timer5 external clock input.
U1RX U1TX U1ARX U1ATX U2RX U2TX	I O I O I O	ST — ST — ST —	UART1 Receive. UART1 Transmit. UART1 Alternate Receive. UART1 Alternate Transmit. UART2 Receive. UART2 Transmit.
V_{DD}	P	—	Positive supply for logic and I/O pins.
V_{SS}	P	—	Ground reference for logic and I/O pins.
V_{REF+}	I	Analog	Analog Voltage Reference (High) input.
V_{REF-}	I	Analog	Analog Voltage Reference (Low) input.

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

2.0 CORE ARCHITECTURE OVERVIEW

2.1 Core Overview

The core has a 24-bit instruction word. The Program Counter (PC) is 23-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 4.0 for details on modulo and bit-reversed addressing.

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

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

A DSP engine has been included to significantly enhance the core arithmetic capability and throughput. It features a high speed 17-bit by 17-bit multiplier, a 40-bit ALU, two 40-bit saturating accumulators and a 40-bit 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 pre-decodes 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.

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

2.2 Programmer's Model

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

- `PUSH.S` and `POP.S`
  W0...W14, TBLPAG, PSVPAG, SR (DC, N, OV, Z and C bits only) transferred
- `DO` instruction
  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

The dsPIC™ devices contain a software stack. 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.

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 SR Low Byte (SRL) and the MS Byte as the SR High Byte (SRH). See Figure 2-1 for SR layout.

SRL contains all the MCU ALU operation status flags (including the Z bit), as well as the CPU Interrupt Priority Level status bits, IPL<2:0>, and the REPEAT active status bit, RA. During exception processing, SRL is concatenated with the 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. Once set, these flags remain set until cleared by the user, irrespective of the results from any subsequent DSP operations.

Note	
1:	Clearing the SAB bit will also clear both the SA and SB bits.
2:	When the memory mapped status register (SR) is the destination address for an operation which affects any of the SR bits, data writes are disabled to all bits.

2.2.2.1 Z Status Bit

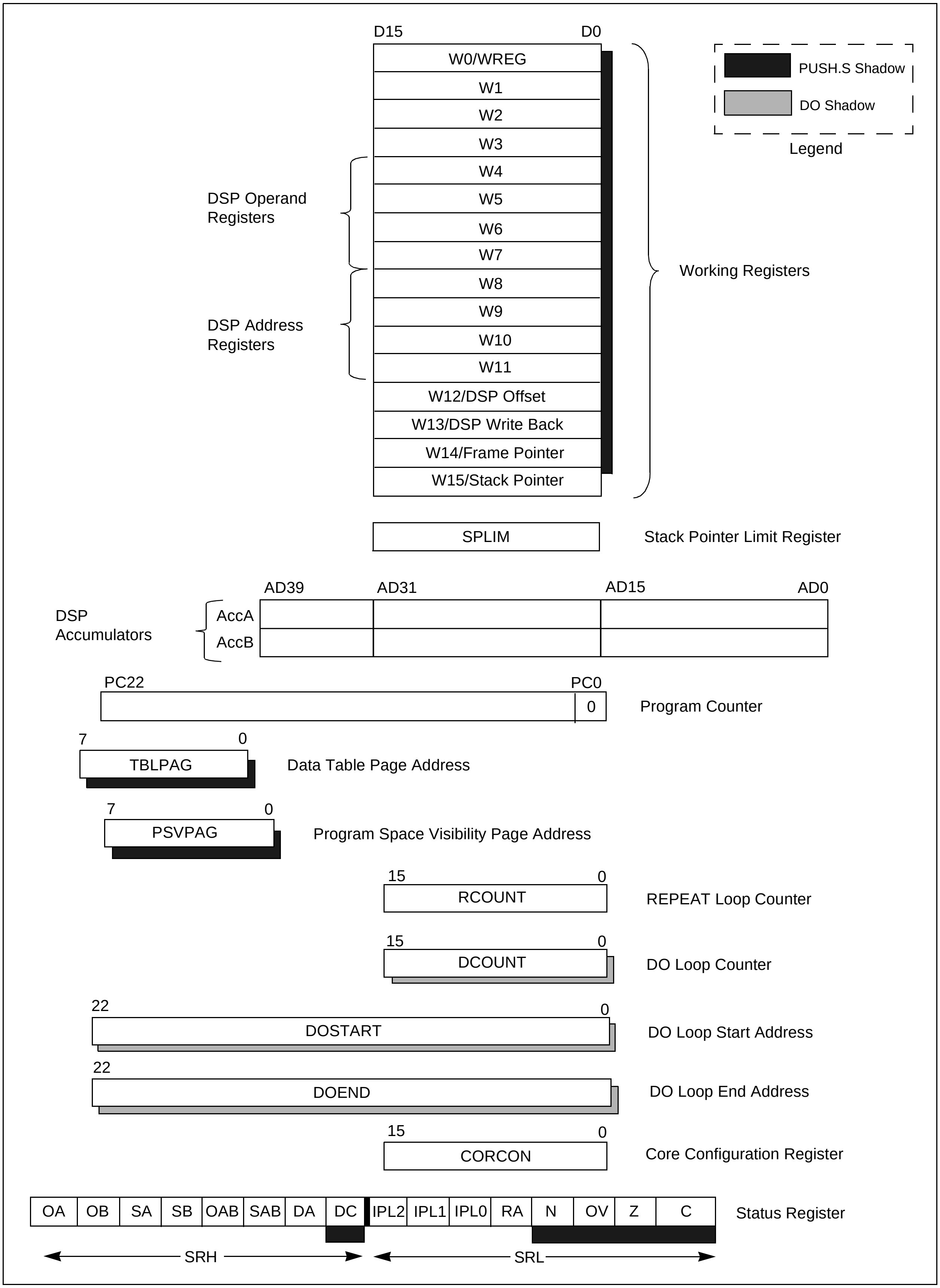
Instructions that use a carry/borrow input (`ADDC`, `CPB`, `SUBB` and `SUBBR`) will only be able to clear Z (for a non-zero result) and can never set it. A multiprecision sequence of instructions (starting with an instruction with no carry/borrow input) will thus, automatically logically AND the successive results of the zero test. All results must be zero for the Z flag to remain set by the end of the sequence.

All other instructions can set as well as clear the Z bit.

2.2.3 PROGRAM COUNTER

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

FIGURE 2-1: PROGRAMMER'S MODEL



2.3 Instruction Flow

There are 8 types of instruction flows:

1. Normal one-word, one-cycle instructions. These instructions take one effective cycle to execute, as shown in Figure 2-2.

FIGURE 2-2: 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		

2. One-word, two-cycle (or three-cycle) 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 2-3. Some program flow change instructions, such as skip instructions that skip over 2-word instructions, and the `RETURN`, `RETFIE` and `RETLW` instructions that are used to return from a subroutine call or Interrupt Service Routine, require 3 cycles.

FIGURE 2-3: 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

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 2-4.

FIGURE 2-4: 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

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 2-5.

FIGURE 2-5: 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

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 2-6.

FIGURE 2-6: 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

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 2-7.

FIGURE 2-7: 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

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 2-7. Instruction stalls caused by data dependencies are further discussed in Section 4.0.

FIGURE 2-8: 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	

8. Interrupt recognition execution. Refer to Section 5.0 for details on interrupts.

2.4 Divide Support

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 the dividend is either zero-extended or sign-extended 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 the 16-bit divisor, but the 32-bit dividend must be in an aligned W register pair, such as W1:W0, W3:W2, etc.

The non-restoring divide algorithm requires one cycle for an initial dividend shift (for integer divides only), one cycle per divisor bit, and 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 depends on RCOUNT. The divide instruction does not automatically set up the RCOUNT value, and it must, therefore, be explicitly and correctly specified in the `REPEAT` instruction, as shown in Table 2-1 (`REPEAT` will execute the target instruction {operand value+1} times). The REPEAT loop count must be set up for 18 iterations of the `DIV/DIVF` instruction. Thus, a complete divide operation requires 19 cycles.

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

TABLE 2-1: DIVIDE INSTRUCTIONS

Instruction	Function
`DIVF`	Signed fractional divide: Wm/Wn → W0; Rem → W1
`DIV.sd`	Signed divide: (Wm+1:Wm)/Wn → W0; Rem → W1
`DIV.sw`	Signed divide: Wm/Wn → W0; Rem → W1
`DIV.ud`	Unsigned divide: (Wm+1:Wm)/Wn → W0; Rem → W1
`DIV.uw`	Unsigned divide: Wm/Wn → W0; Rem → W1

2.5 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 17-bit x 17-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.

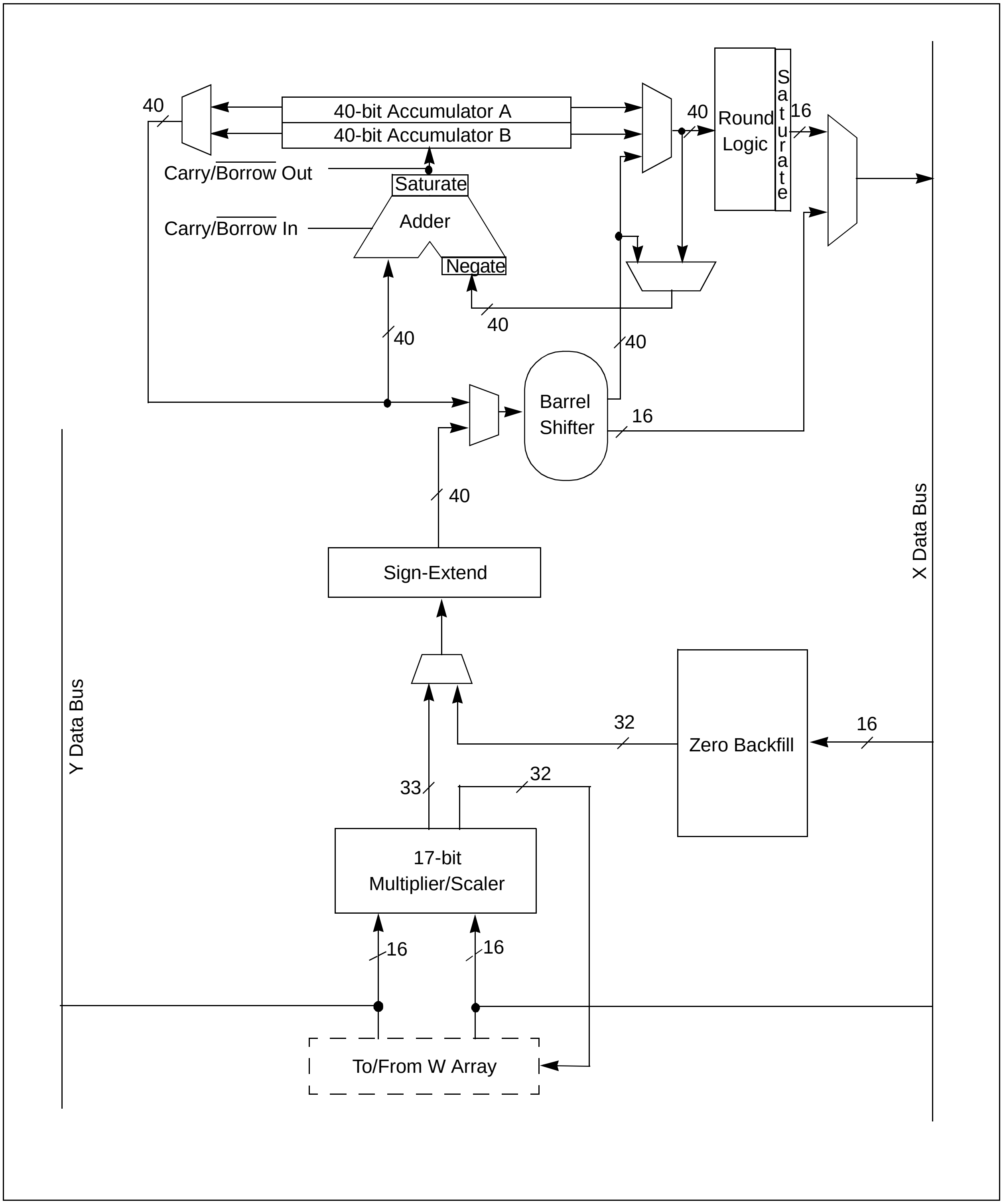
The DSP engine has various options selected through various bits in the CPU Core Configuration Register (CORCON), as listed below:

1. Fractional or integer multiply (IF).
2. Conventional or convergent rounding (RND).
3. Automatic saturation on/off for AccA (SATA).
4. Automatic saturation on/off for AccB (SATB).
5. Automatic saturation on/off for writes to data memory (SATDW).
6. Accumulator Saturation mode selection (ACCSAT).

Note:	For CORCON layout, see Table 4-3.

A block diagram of the DSP engine is shown in Figure 2-9.

FIGURE 2-9: DSP ENGINE BLOCK DIAGRAM



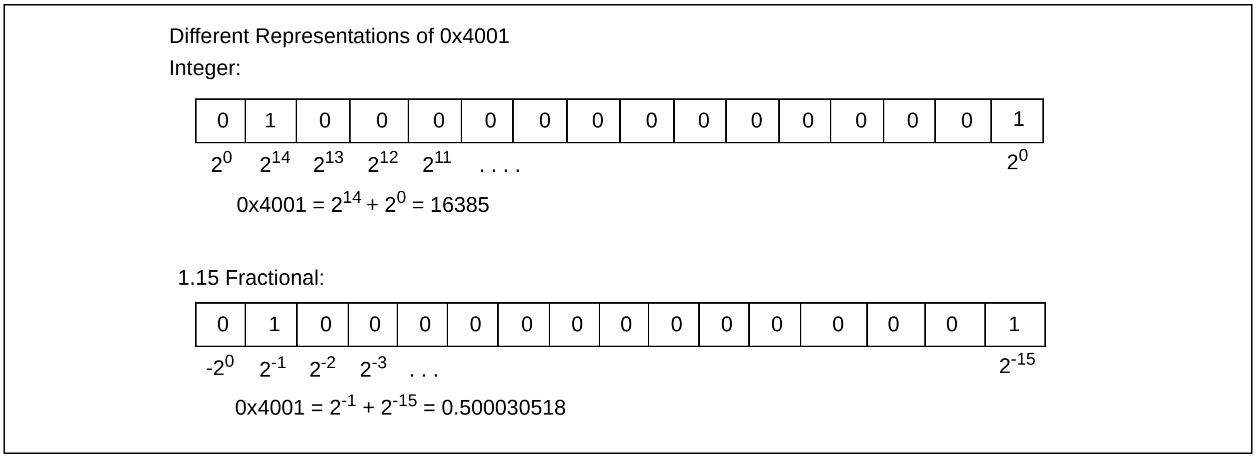
 Advance Information

2.5.1 MULTIPLIER

The 17x17-bit multiplier is capable of signed or unsigned operation and can multiplex its output using a scaler to support either 1.31 fractional (Q31), or 32-bit integer results. Unsigned operands are zero-extended into the 17th bit of the multiplier input value. Signed operands are sign-extended into the 17th bit of the multiplier input value. The output of the 17x17-bit multiplier/scaler is a 33-bit value, which is sign-extended to 40 bits. The respective number representation formats are shown in Figure 2-10. 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 2-10). For a 32-bit integer, the data range is -2,147,483,648 (0x8000 0000) to 2,147,483,645 (0x7FFF FFFF).

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

FIGURE 2-10: 16-BIT INTEGER AND FRACTIONAL MODES



Certain multiply operations always operate on signed data. These include the `MAC/MSC`, `MPY[.N]` and `ED[AC]` instructions. 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.

2.5.2 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.

2.5.2.1 Adder/Subtractor, Overflow and Saturation

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

- Overflow from bit 39: this is a catastrophic overflow in which the sign of the accumulator is destroyed.
- Overflow into guard bits 32 through 39: this is a recoverable overflow. This bit is set whenever all the guard bits 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 (refer to Section 5.0) is set. This allows the user to take immediate action, for example, to correct system gain.

The SA and SB bits are modified each time data passes through the adder/subtractor, but can only be cleared by the user. When set, they indicate that the accumulator has overflowed its maximum range (bit 31 for 32-bit saturation, or bit 39 for 40-bit saturation) and 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 (SR) as the logical OR of OA and OB (in bit OAB) and the logical OR of SA and SB (in bit SAB). This allows programmers to check one bit in the Status Register to determine if either accumulator has overflowed, or one bit to determine if either accumulator has saturated. This would be useful for complex number arithmetic which typically uses both the accumulators.

The device supports three Saturation and Overflow modes.

1. Bit 39 Overflow and Saturation:
   When bit 39 overflow and saturation occurs, the saturate logic 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:
   When bit 31 overflow and saturation occurs, 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.

2.5.2.2 Accumulator 'Write Back'

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

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

2.5.2.3 Round Logic

The round logic is a combinational block, which performs a conventional (biased) or convergent (unbiased) round function during an accumulator write (store). The Round mode is determined by the state of the RND bit in the CORCON register. It generates a 16-bit, 1.15 data value which is passed to the data space write saturation logic. If rounding is not indicated by the instruction, a truncated 1.15 data value is stored and the 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 ACCxH word (bits 16 through 31 of the accumulator). If the ACCxL word (bits 0 through 15 of the accumulator) is between 0x8000 and 0xFFFF (0x8000 included), ACCxH is incremented. If ACCxL is between 0x0000 and 0x7FFF, ACCxH is left unchanged. A consequence of this algorithm is that over a succession of random rounding operations, the value will tend to be biased slightly positive.

Convergent (or unbiased) rounding operates in the same manner as conventional rounding, except when ACCxL equals 0x8000. If this is the case, the LS bit (bit 16 of the accumulator) of ACCxH is examined. If it is 1, ACCxH is incremented. If it is 0, ACCxH is not modified. Assuming that bit 16 is effectively random in nature, this scheme 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 2.5.2.4). 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.

2.5.2.4 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.

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

NOTES:

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.

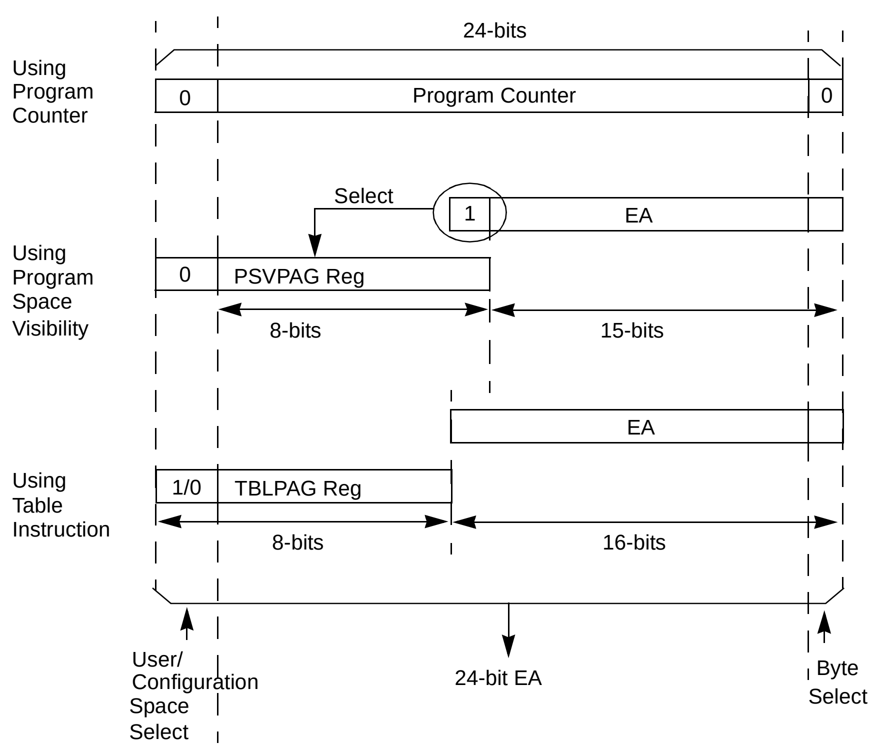
User program space access is restricted to the lower 4M instruction word address range (0x000000 to 0x7FFFFE), for all accesses other than `TBLRD/TBLWT`, which use TBLPAG<7> to determine user or configuration space access. In Table 3-1, Read/Write instructions, bit 23 allows access to the Device ID,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

> **Note:** The address map shown in Figure 3-5 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>		
Program Space Visibility	User	0	PSVPAG<7:0>		Data EA <14:0>	

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



Note: Program Space Visibility 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 space word can be accessed as data.

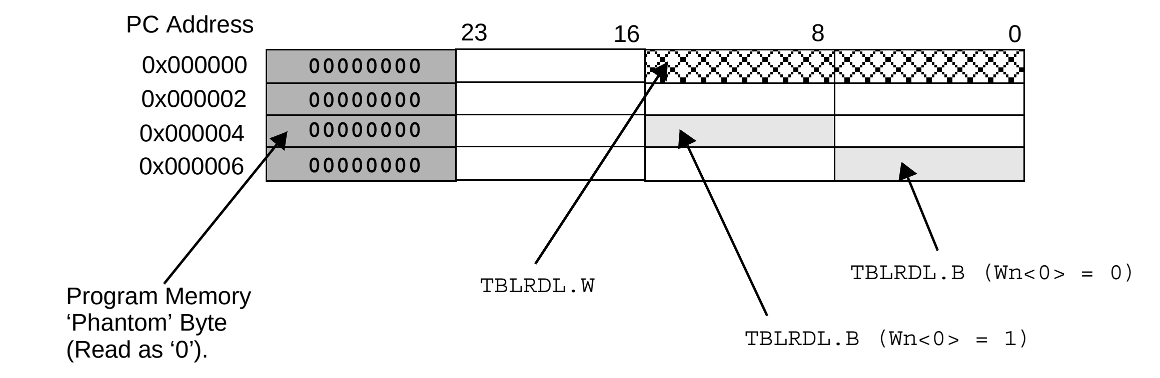
The PC is incremented by two for each successive 24-bit program word. This allows program memory addresses to directly map to data space addresses. Program memory can thus be regarded as two 16-bit word wide address spaces, residing side by side, each with the same address range. `TBLRDL` and `TBLWTL`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

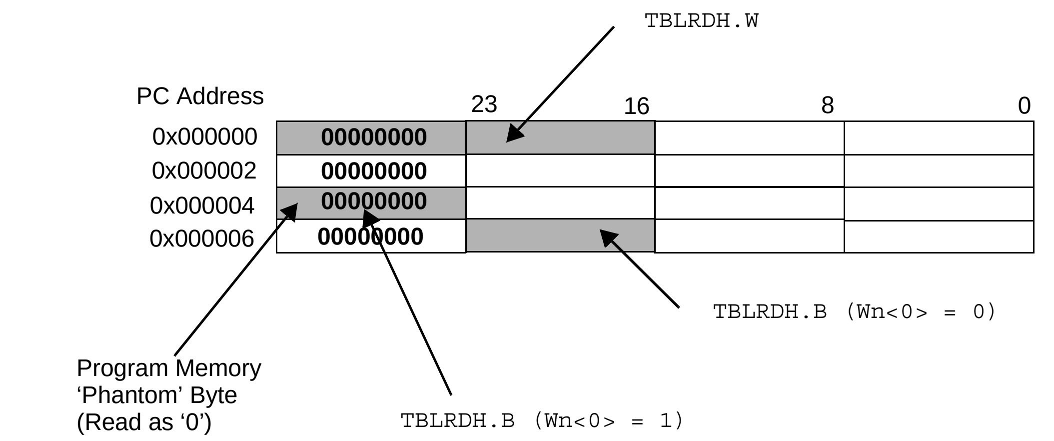
1. **`TBLRDL:`** Table Read Low
   Word: Read the LS Word of the program address;
   P<15:0> maps to D<15:0>.
   Byte: Read one of the LS Bytes of the program address;
   P<7:0> maps to the destination byte when byte select = 0;
   P<15:8> maps to the destination byte when byte select = 1.
2. **`TBLWTL:`** Table Write Low (refer to Section 6.0 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 be = 0.
   Byte: Read one of the MS Bytes of the program address;
   P<23:16> maps to the destination byte when byte select = 0;
   The destination byte will always be = 0 when byte select = 1.
4. **`TBLWTH:`** Table Write High (refer to Section 6.0 for details on FLASH Programming).

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



 Advance Information

FIGURE 3-3: PROGRAM DATA TABLE ACCESS (MS BYTE)



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 3-4), only the lower 16-bits of the 24-bit program word are used to contain the data. The upper 8 bits should be programmed to force an illegal instruction to maintain machine robustness. Refer to the 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 15 bits (16 bits for the `TBLRDL/H, TBLWTL/H` instructions) of data space addresses directly map to the LS 15 bits in the corresponding program space addresses. The remaining bits are provided by the Program Space Visibility Page register, PSVPAG<7:0>, as shown in Figure 3-4.

Note:	PSV access is temporarily disabled during Table Reads/Writes.

3.1.2.1 Data Pre-Fetch from Program Space Within a REPEAT Loop

Consider the special case of any instruction executed within the 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. In this example, the initial 2 data words for the first iteration of the instruction to be repeated (`MAC)` are fetched by a prior instruction (e.g., `CLR` instruction). The subsequent `MAC` instructions, therefore, only need to fetch two more data pairs to complete the loop. The initial fetch of the `MAC` instruction is performed by the `REPEAT` instruction. As a result, only the last iteration of the REPEAT loop requires 2 cycles to execute. Each of the other iterations executes in 1 cycle.

EXAMPLE 3-1: PROGRAM SPACE DATA READ THROUGH DATA SPACE WITHIN A REPEAT LOOP

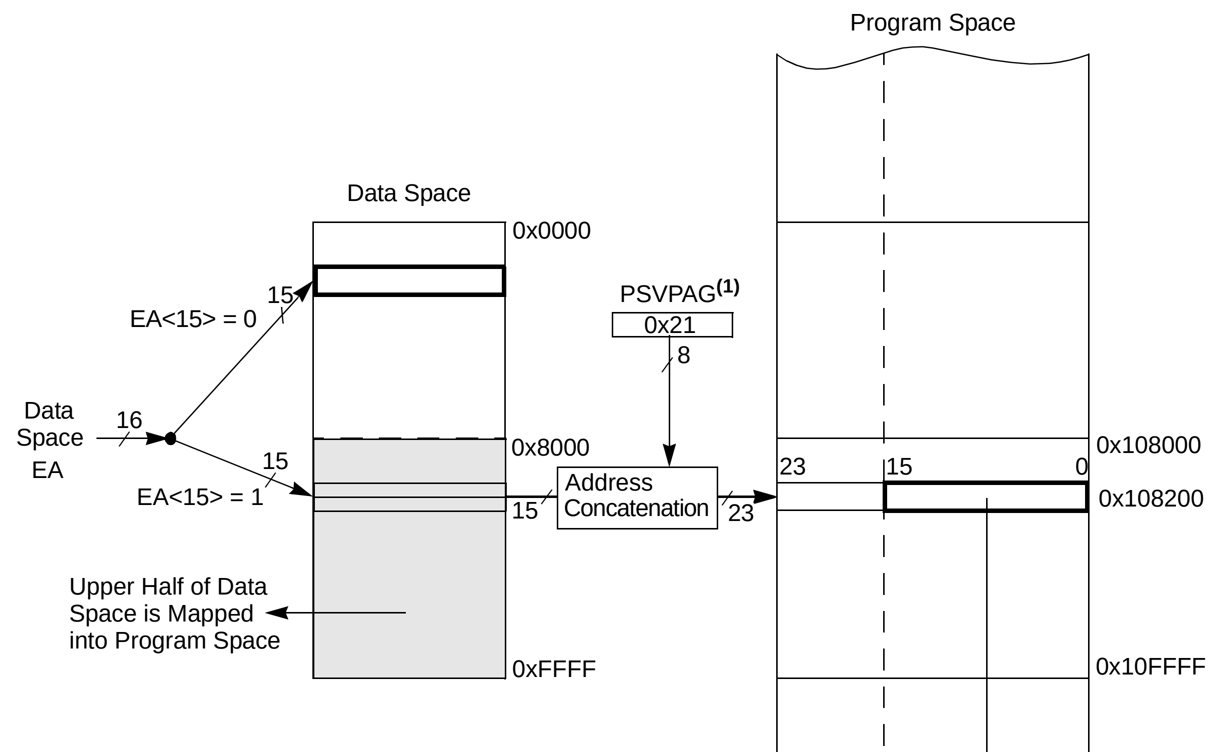
```
; In this example, data for MAC operations is stored in Program Space.

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 (2 cycles for 1st and 100th iterations, 1 cycle for 2nd - 99th
iterations)
```

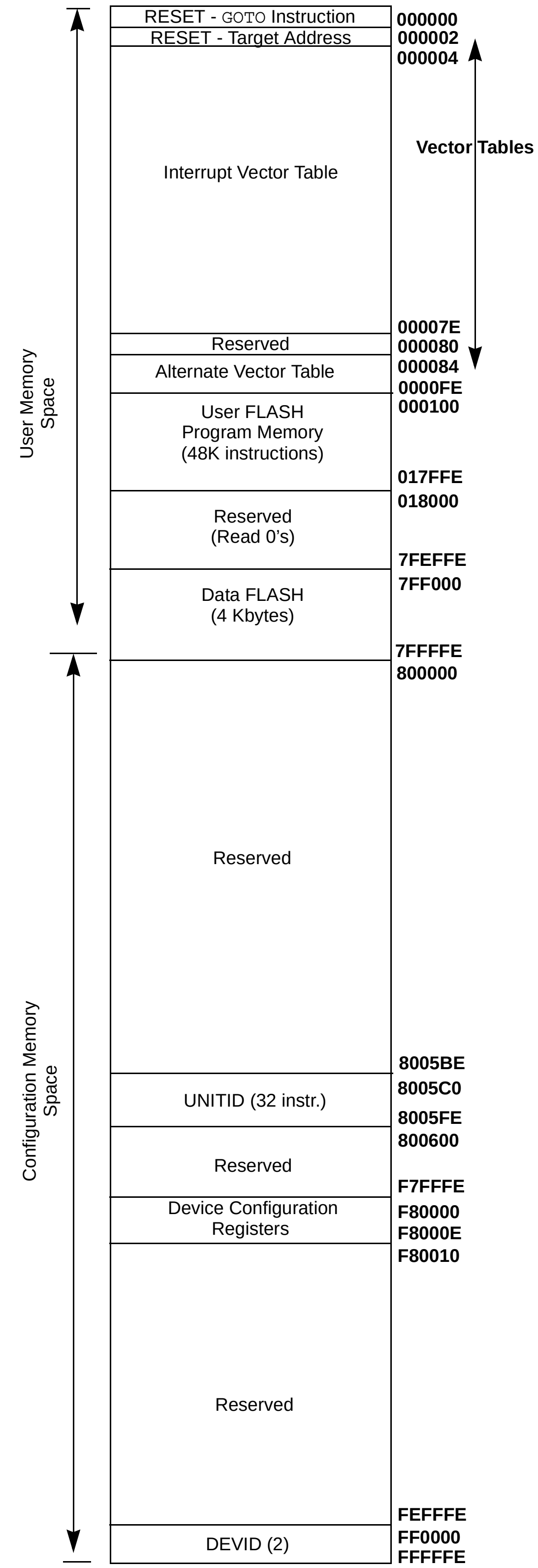
FIGURE 3-4: DATA SPACE WINDOW INTO PROGRAM SPACE OPERATION




```
BSET   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: 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 3-5: SAMPLE PROGRAM SPACE MEMORY MAP



Note: These address boundaries may vary from one device to another.

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

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

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

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

The boundary between the X and Y data spaces is defined as shown in Figure 3-8 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. Misaligned word data fetches are not supported, so care must be taken when mixing byte and word operations, or translating from 8-bit MCU code. Should a misaligned read or write be attempted, an Address Error trap will be generated. 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 3-6: DATA ALIGNMENT

	MS Byte (15 … 8)	LS Byte (7 … 0)	
0001	Byte 1	Byte 0	0000
0003	Byte 3	Byte 2	0002
0005	Byte 5	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.

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 one of the `MAC` class of instructions, the X block consists of the entire 64 Kbyte data address space excluding the Y address block (for data reads only). In other words, all other instructions regard the entire data memory as one composite address space. The `MAC` class instructions extract the Y address space from data space and 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 instructions.

An example data space memory map is shown in Figure 3-8.

3.2.5 NEAR DATA SPACE

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

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 3-7. 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.

3.2.6 SOFTWARE STACK

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

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 (Stack Error) trap is generated.

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

FIGURE 3-7: `CALL` STACK FRAME

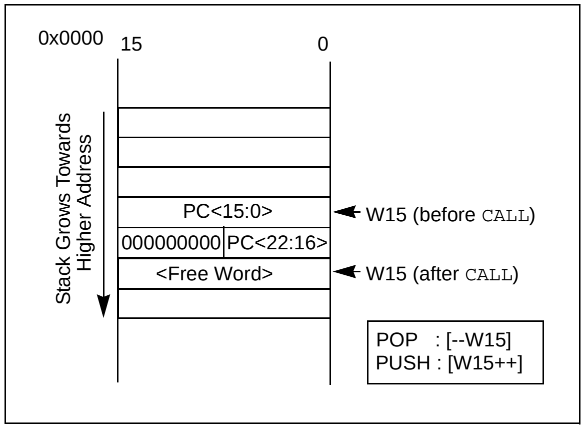
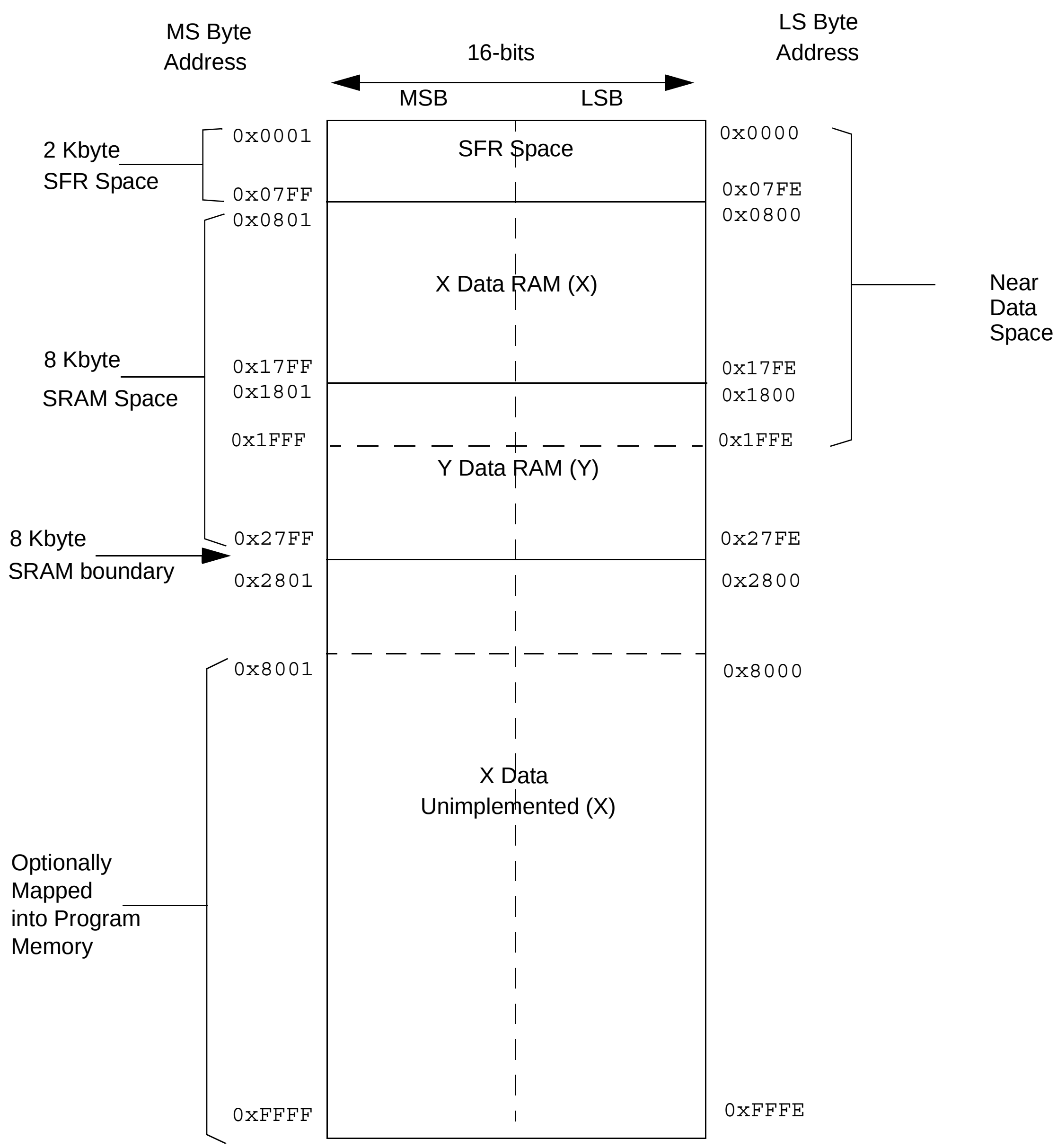


FIGURE 3-8: SAMPLE DATA SPACE MEMORY MAP



Note: The address map shown in Figure 3-8 is conceptual, and may vary across individual devices depending on available memory.

FIGURE 3-9: DATA SPACE FOR MCU AND DSP (MAC CLASS) INSTRUCTIONS EXAMPLE

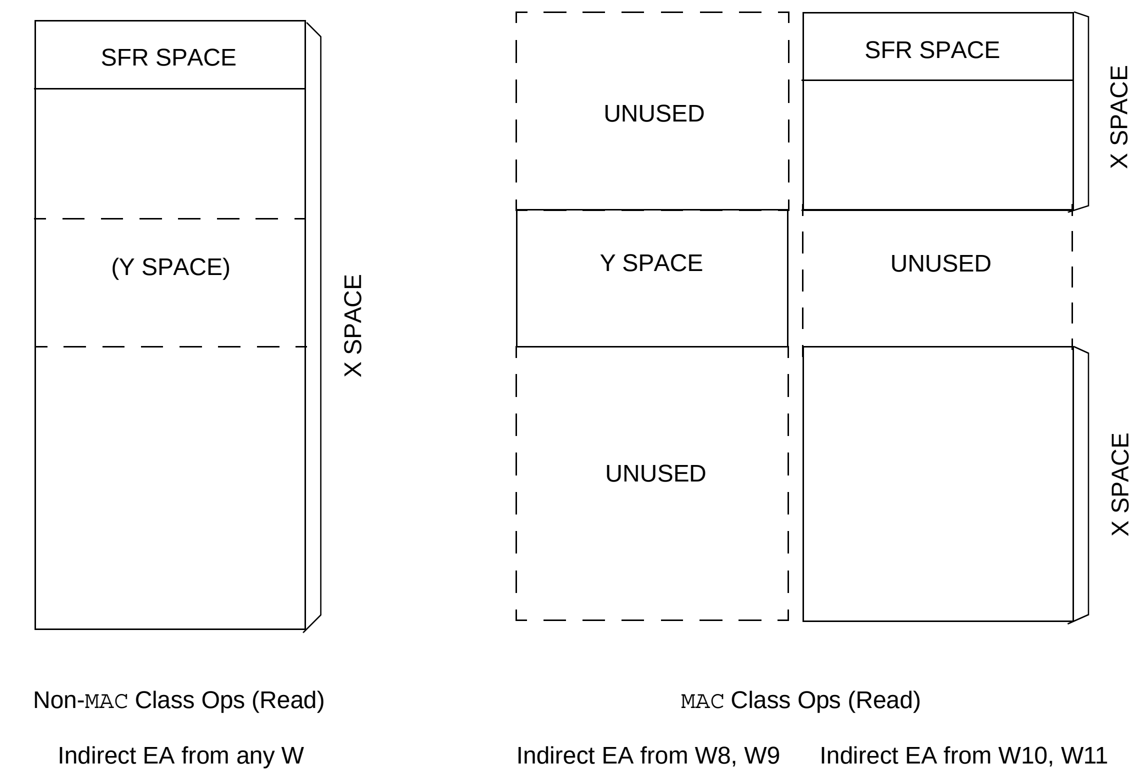


TABLE 3-3: CORE REGISTER MAP

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	Sign-Extension (ACCA<39>)								ACCAU								0000 0000 0000 0000
ACCBL	0028	ACCBL																0000 0000 0000 0000
ACCBH	002A	ACCBH																0000 0000 0000 0000
ACCBU	002C	Sign-Extension (ACCB<39>)								ACCBU								0000 0000 0000 0000
PCL	002E	PCL																0000 0000 0000 0000
PCH	0030	—	—	—	—	—	—	—	—	—	PCH							0000 0000 0000 0000
TBLPAG	0032	—	—	—	—	—	—	—	—	TBLPAG								0000 0000 0000 0000
PSVPAG	0034	—	—	—	—	—	—	—	—	PSVPAG								0000 0000 0000 0000
RCOUNT	0036	RCOUNT																uuuu uuuu uuuu uuuu
DCOUNT	0038	DCOUNT																uuuu uuuu uuuu uuuu
DOSTARTL	003A	DOSTARTL															0	uuuu uuuu uuuu uuu0
DOSTARTH	003C	—	—	—	—	—	—	—	—	—	DOSTARTH							0000 0000 0uuu uuuu
DOENDL	003E	DOENDL															0	uuuu uuuu uuuu uuu0
DOENDH	0040	—	—	—	—	—	—	—	—	—	DOENDH							0000 0000 0uuu uuuu
SR	0042	OA	OB	SA	SB	OAB	SAB	DA	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C	0000 0000 0000 0000
CORCON	0044	—	—	—	—	EDT	DL2	DL1	DL0	SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF	0000 0000 0000 0000

Legend: u = uninitialized bit

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	—	—	BWM<3:0>				YWM<3:0>				XWM<3:0>				0000 0000 0000 0000
XMODSRT	0048	XS<15:1>															0	uuuu uuuu uuuu uuu0
XMODEND	004A	XE<15:1>															1	uuuu uuuu uuuu uuu1
YMODSRT	004C	YS<15:1>															0	uuuu uuuu uuuu uuu0
YMODEND	004E	YE<15:1>															1	uuuu uuuu uuuu uuu1
XBREV	0050	BREN	XB<14:0>															uuuu uuuu uuuu uuuu
DISICNT	0052	—	—	DISICNT<13:0>														0000 0000 0000 0000

Legend: u = uninitialized bit

NOTES:

4.0 ADDRESS GENERATOR UNITS

The dsPIC core contains two independent address generator units: the X AGU and Y AGU. Further, the X AGU has two parts: X RAGU (Read AGU) and X WAGU (Write AGU). The X RAGU and X WAGU support byte and word sized data space reads and writes, respectively, for both 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.

4.1 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).

4.2 Instruction Addressing Modes

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

Some Addressing mode combinations may lead to a one-cycle stall during instruction execution, or are not allowed, as discussed in Section 4.3.

TABLE 4-1: FUNDAMENTAL ADDRESSING MODES SUPPORTED

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

4.2.1 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. The MOV instruction can use a 16-bit address field.

4.2.2 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-modified
- Register Indirect Pre-modified
- 5-bit or 10-bit Literal

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

4.2.3 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 are supported by Move and Accumulator instructions:

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

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

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

4.2.5 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.

4.3 Instruction Stalls

4.3.1 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, also known as 'Read After Write' (RAW) dependencies, may therefore arise between successive read and write operations using common registers. They occur across instruction boundaries and are detected by the hardware.

An example of a RAW dependency is a write operation (in the current instruction) that modifies W5, followed by a read operation (in the next instruction) that uses W5 as a source address pointer. 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.

4.3.2 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 prefetched 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, 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 4-2 lists out the various RAW conditions which cause an instruction execution stall.

TABLE 4-2: RAW DEPENDENCY RULES (DETECTION BY HARDWARE)

Destination Addressing Mode Using Wn	Source Addressing Mode Using Wn	Status	Examples (Wn = W2)
Direct	Direct	No Stall	`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	No Stall	`ADD.w W0, W1, [W2]` `MOV.w W2, W3`
Indirect	Indirect	No Stall	`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	No Stall	`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	No Stall	`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`

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

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

4.4.1 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 (see Table 3-3).

> **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). Moreover, the start address must be even and the end address must be odd (for both word and byte buffers).

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:** 'Start 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 4.4.4).

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-space Modulo Addressing EA calculations assume word sized data (LS bit of every EA is always clear).

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

4.4.2 W ADDRESS REGISTER SELECTION

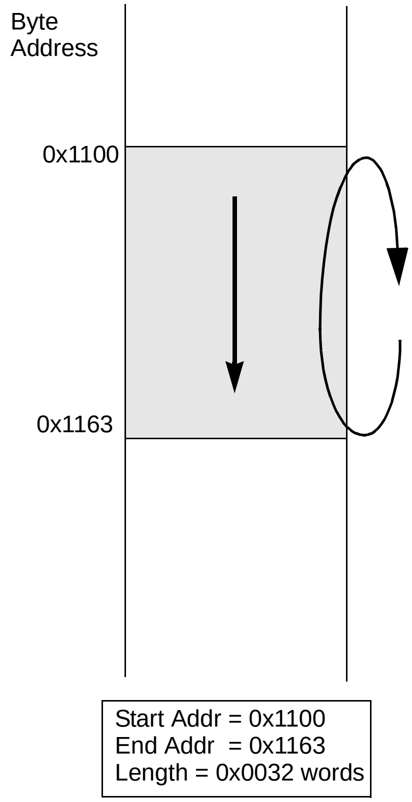
The Modulo and Bit-Reversed Addressing control register MODCON<15:0> contains enable flags plus W register field to specify the W address registers. The XWM and YWM fields select which registers will operate with Modulo Addressing. If XWM = 15, X RAGU and X WAGU Modulo Addressing are disabled. Similarly, 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 Table 3-3). Modulo Addressing is enabled for X data space when XWM is set to any value other than 15 and the XMODEN bit is set at MODCON<15>.

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

FIGURE 4-1: INCREMENTING BUFFER MODULO ADDRESSING OPERATION EXAMPLE



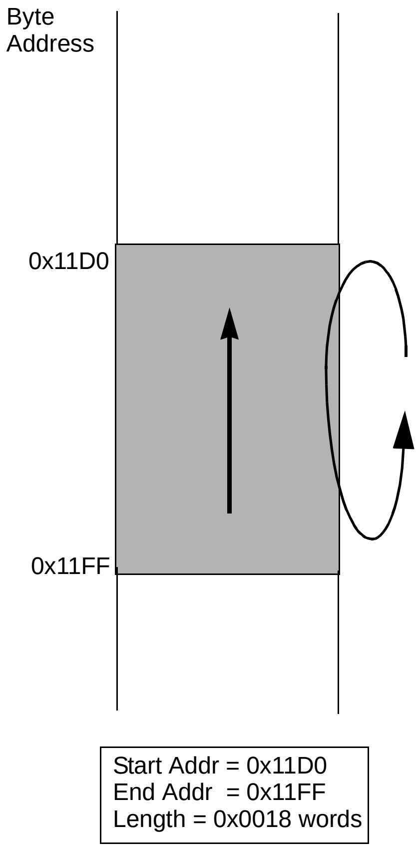

```
MOV     #0x1100,W0
MOV     W0,XMODSRT      ;set modulo start address
MOV     #0x1163,W0
MOV     W0,MODEND       ;set modulo end address
MOV     #0x8001,W0
MOV     W0,MODCON       ;enable W1, X AGU for modulo

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

MOV     #0x1110,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 4-2: DECREMENTING BUFFER MODULO ADDRESSING OPERATION EXAMPLE




```
MOV    #0x11D0,W0
MOV    #0,XMODSRT      ;set modulo start address
MOV    0x11FF,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    #0x11E0,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
```

4.4.3 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 4.4.4 for restrictions)

.

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

4.4.4 MODULO ADDRESSING RESTRICTIONS

As stated in Section 4.4.1, 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.

4.5 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.

4.5.1 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 used is Register Indirect with Pre-Increment or 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 Pre-Increment or 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 4-3: BIT-REVERSED ADDRESS EXAMPLE

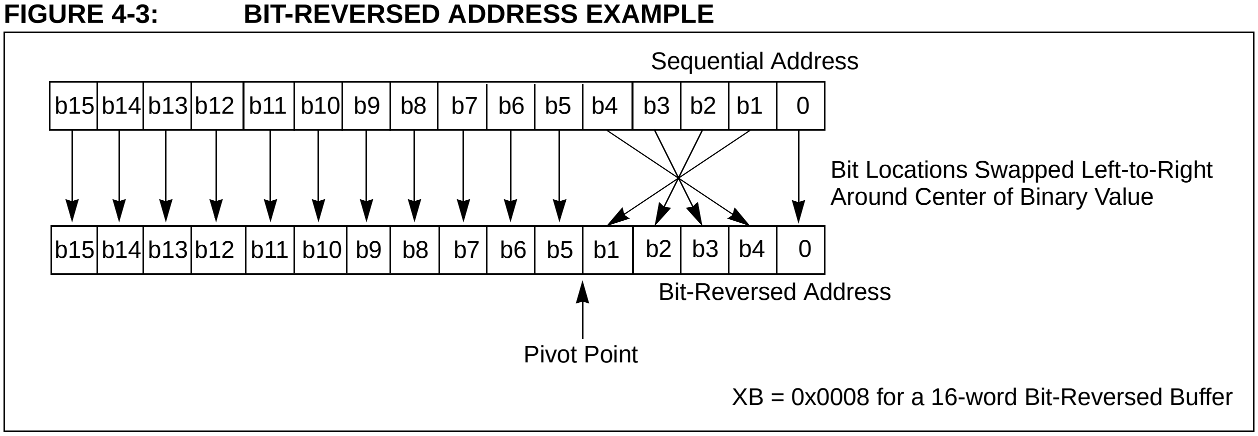


TABLE 4-3: BIT-REVERSED ADDRESS SEQUENCE (16-ENTRY)

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

TABLE 4-4: 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

5.0 EXCEPTION PROCESSING

The dsPIC30F Motor Control and Power Conversion Family has up to 44 interrupt sources and 4 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 5-2.

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

- IFS0<15:0>, IFS1<15:0>, IFS2<15:0>
  All interrupt request flags are maintained in these three registers. The flags are set by their respective peripherals or external signals, and they are cleared via software.
- IEC0<15:0>, IEC1<15:0>, IEC2<15:0>
  All interrupt enable control bits are maintained in these three registers. These control bits are used to individually enable interrupts from the peripherals or external signals.
- IPC0<15:0> ... IPC11<7:0>
  The user assignable priority level associated with each of these 44 interrupts is held centrally in these twelve registers.
- IPL<3:0> The current CPU priority level is explicitly stored in the IPL bits. IPL<3> is present in the CORCON register, whereas IPL<2:0> are present in the status register (SR) in the processor core.
- INTCON1<15:0>, INTCON2<15:0>
  Global interrupt control functions are derived from these two registers. INTCON1 contains the control and status flags for the processor exceptions. The INTCON2 register controls the external interrupt request signal behavior and the use of the alternate vector table.

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

All interrupt sources can be user assigned to one of 7 priority levels, 1 through 7, via the IPCx registers. Each interrupt source is associated with an interrupt vector, as shown in Table 5-2. Levels 7 and 1 represent the highest and lowest maskable priorities, respectively.

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

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

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

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

When an interrupt is serviced, the PC is loaded with the address stored in the vector location in Program memory that corresponds to the interrupt. There are 63 different vectors within the IVT (refer to Table 5-2). These vectors are contained in locations 0x000004 through 0x0000FE of program memory (refer to Table 5-2). These locations contain 24-bit addresses, and in order to preserve robustness, an address error trap will take place should the PC attempt to fetch any of these words during normal execution. This prevents execution of random data 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.

5.1 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 5-1 lists the interrupt numbers and interrupt sources for the dsPIC devices, and their associated vector numbers.

> **Note 1:** The natural order priority scheme has 0 as the highest priority and 53 as the lowest priority.
>
> **2:** The natural order priority number is the same as the INT number.

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

TABLE 5-1: NATURAL ORDER PRIORITY

INT Number	Vector Number	Interrupt Source
Highest Natural Order Priority		
0	8	INT0 - External Interrupt 0
1	9	IC1 - Input Capture 1
2	10	OC1 - Output Compare 1
3	11	T1 - Timer 1
4	12	IC2 - Input Capture 2
5	13	OC2 - Output Compare 2
6	14	T2 - Timer 2
7	15	T3 - Timer 3
8	16	SPI1
9	17	U1RX - UART1 Receiver
10	18	U1TX - UART1 Transmitter
11	19	ADC - ADC Convert Done
12	20	NVM - NVM Write Complete
13	21	I2C - I^2C Transfer Complete
14	22	BCL - I^2C 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 - QEI Interrupt
41	49	Reserved
42	50	LVD - Low Voltage Detect
43	51	FLTA - PWM Fault A
44	52	FLTB - PWM Fault B
45 - 53	53 - 61	Reserved
Lowest Natural Order Priority		

5.2 RESET Sequence

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

5.2.1 RESET SOURCES

In addition to external, Power-on Resets (POR) and software Reset, there are four 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.
- Uninitialized W Register Trap:
  An attempt to use an uninitialized W register as an address pointer will cause a RESET.
- Illegal Instruction Trap:
  Attempted execution of any unused opcodes will result in an illegal instruction trap. Note that a fetch of an illegal instruction does not result in an illegal instruction trap if that instruction is flushed prior to execution due to a flow change.
- Brown-out Reset (BOR):
  A momentary dip in the power supply to the device has been detected, which may result in malfunction.

5.3 Traps

Traps can be considered as non-maskable, non-stable interrupts, which adhere to a predefined priority as shown in Table 5-2. 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 that caused the trap may have to be corrected.

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

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

Note:	The IPL3 bit is read only, which implies that the user can not manually set the IPL<3:0> bits to a value greater than 7.

5.3.1 TRAP SOURCES

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

- Math Error Trap:
  The Math 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, a Math Error trap will be taken when an arithmetic operation on either accumulator A or B, causes an overflow from bit 31 and the accumulator guard bits are not utilized.
  3. If enabled, a Math Error trap will be taken when an arithmetic operation on either accumulator A or B causes a catastrophic overflow from bit 39 and all saturation is disabled.
- 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.
- 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.

5.3.2 HARD AND SOFT TRAPS

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

‘Soft’ traps include exceptions of priority level 8 through level 11, inclusive. The arithmetic error trap (level 11) falls into this category of traps. Soft traps can be treated like non-maskable sources of interrupt that adhere to the priority assigned by their position in the IVT. Soft traps are processed like interrupts and require 2 cycles to be sampled and Acknowledged prior to exception processing. Therefore, additional instructions may be executed before a soft trap is Acknowledged.

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

Like soft traps, hard traps can also be viewed as non-maskable sources of interrupt. The difference between hard traps and soft traps is that hard traps force the CPU to stop code execution after the instruction causing the trap has completed. Normal program execution flow will not resume until after the trap has been acknowledged and processed.

If a higher priority trap occurs while any lower priority trap is in progress, processing of the lower priority trap will be suspended and the higher priority trap will be acknowledged and processed. The lower priority trap will remain pending until processing of the higher priority trap completes.

Each hard trap that occurs must be acknowledged before code execution of any type may continue. If a lower priority hard trap occurs while a higher priority trap is pending, acknowledged, or is being processed, a hard trap conflict will occur. The conflict occurs because the lower priority trap cannot be Acknowledged until processing for the higher priority trap completes.

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

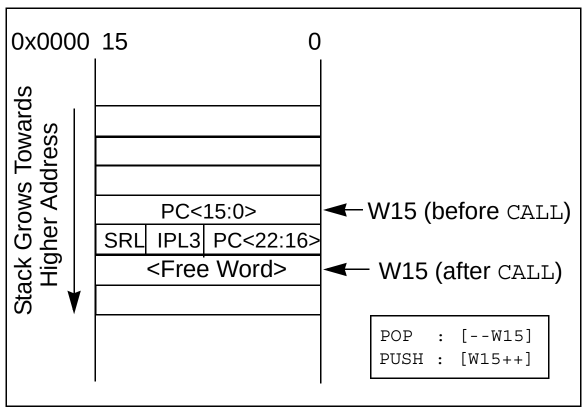
5.4 Interrupt Sequence

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

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

The processor then stacks the current program counter and the low byte of the processor status register (SRL), as shown in Figure 5-1. The low byte of the status register contains the processor priority level at the time, prior to the beginning of the interrupt cycle. The processor then loads the priority level for this interrupt into the status register. This action will disable all lower priority interrupts until the completion of the Interrupt Service Routine.

FIGURE 5-1: INTERRUPT STACK FRAME



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

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

TABLE 5-2: EXCEPTION VECTORS

Decreasing Priority

Table	Vector	Address
	RESET - GOTO Instruction	0x000000
	RESET - GOTO Address	0x000002
IVT	Reserved	0x000004
IVT	Oscillator Fail Trap Vector	
IVT	Address Error Trap Vector	
IVT	Stack Error Trap Vector	
IVT	Math Error Trap Vector	
IVT	Reserved Vector	
IVT	Reserved Vector	
IVT	Reserved Vector	
IVT	Interrupt 0 Vector	0x000014
IVT	Interrupt 1 Vector	
IVT	—	
IVT	—	
IVT	—	
IVT	Interrupt 52 Vector	
IVT	Interrupt 53 Vector	0x00007E
	Reserved	0x000080
	Reserved	0x000082
AIVT	Reserved	0x000084
AIVT	Oscillator Fail Trap Vector	
AIVT	Stack Error Trap Vector	
AIVT	Address Error Trap Vector	
AIVT	Math Error Trap Vector	
AIVT	Reserved Vector	
AIVT	Reserved Vector	
AIVT	Reserved Vector	
AIVT	Interrupt 0 Vector	0x000094
AIVT	Interrupt 1 Vector	
AIVT	—	
AIVT	—	
AIVT	—	
AIVT	Interrupt 52 Vector	
AIVT	Interrupt 53 Vector	0x0000FE

5.5 Alternate Vector Table

In Program Memory, the Interrupt Vector Table (IVT) is followed by the Alternate Interrupt Vector Table (AIVT), as shown in Table 5-2. 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.

5.6 Fast Context Saving

A context saving option is available using shadow registers. Shadow registers are provided for the DC, N, OV, Z and C bits in SR, 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.

5.7 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.

5.8 Wake-up from SLEEP and IDLE

The interrupt controller may be used to wake up the processor from either SLEEP or IDLE modes, if SLEEP or IDLE mode is active when the interrupt is generated.

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

TABLE 5-3: 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	NSTDIS	—	—	—	—	OVATE	OVBTE	COVTE	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000 0000 0000 0000
INTCON2	0082	ALTIVT	—	—	—	—	—	—	—	—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP	0000 0000 0000 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	—	QEIIF	PWMIF	C2IF	INT4IF	INT3IF	OC8IF	OC7IF	OC6IF	OC5IF	0000 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	—	QEIIE	PWMIE	C2IE	INT4IE	INT3IE	OC8IE	OC7IE	OC6IE	OC5IE	0000 0000 0000 0000
IPC0	0094	—	T1IP<2:0>			—	OC1IP<2:0>			—	IC1IP<2:0>			—	INT0IP<2:0>			0100 0100 0100 0100
IPC1	0096	—	T31P<2:0>			—	T2IP<2:0>			—	OC2IP<2:0>			—	IC2IP<2:0>			0100 0100 0100 0100
IPC2	0098	—	ADIP<2:0>			—	U1TXIP<2:0>			—	U1RXIP<2:0>			—	SPI1IP<2:0>			0100 0100 0100 0100
IPC3	009A	—	CNIP<2:0>			—	BCLIP<2:0>			—	I2CIP<2:0>			—	NVMIP<2:0>			0100 0100 0100 0100
IPC4	009C	—	OC3IP<2:0>			—	IC8IP<2:0>			—	IC7IP<2:0>			—	INT1IP<2:0>			0100 0100 0100 0100
IPC5	009E	—	INT2IP<2:0>			—	T5IP<2:0>			—	T4IP<2:0>			—	OC4IP<2:0>			0100 0100 0100 0100
IPC6	00A0	—	C1IP<2:0>			—	SPI2IP<2:0>			—	U2TXIP<2:0>			—	U2RXIP<2:0>			0100 0100 0100 0100
IPC7	00A2	—	IC6IP<2:0>			—	IC5IP<2:0>			—	IC4IP<2:0>			—	IC3IP<2:0>			0100 0100 0100 0100
IPC8	00A4	—	OC8IP<2:0>			—	OC7IP<2:0>			—	OC6IP<2:0>			—	OC5IP<2:0>			0100 0100 0100 0100
IPC9	00A6	—	PWMIP<2:0>			—	C2IP<2:0>			—	INT41IP<2:0>			—	INT3IP<2:0>			0100 0100 0100 0100
IPC10	00A8	—	FLTAIP<2:0>			—	LVDIP<2:0>			—	—	—	—	—	QEIIP<2:0>			0100 0100 0000 0100
IPC11	00AA	—	—	—	—	—	—	—	—	—	—	—	—	—	FLTBIP<2:0>			0000 0000 0000 0100

Legend: u = uninitialized bit

6.0 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™)

6.1 In-Circuit Serial Programming (ICSP)

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

6.2 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.

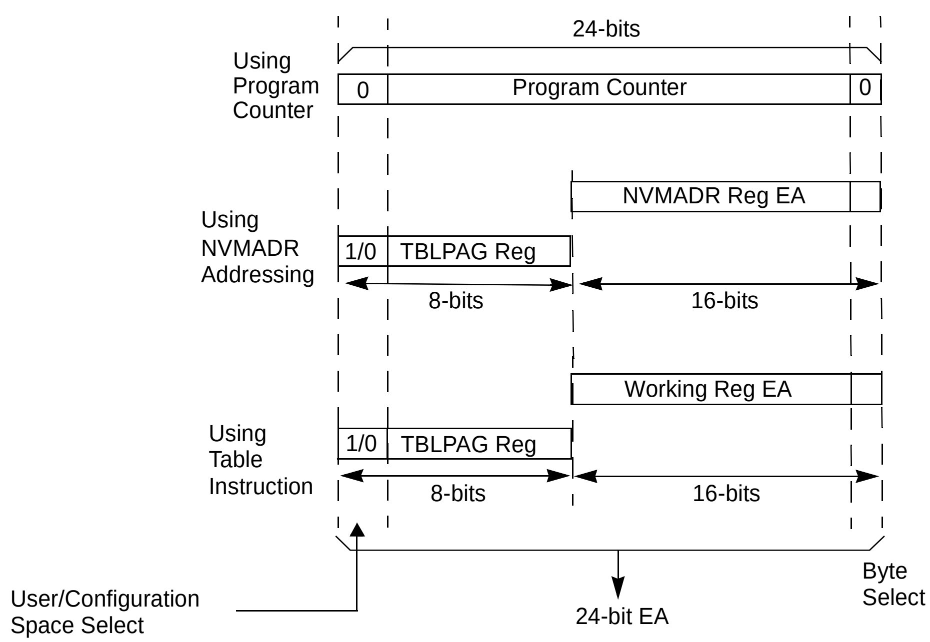
6.3 Table Instruction Operation Summary

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

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

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

FIGURE 6-1: ADDRESSING FOR TABLE AND NVM REGISTERS



6.4 RTSP Operation

The dsPIC30F FLASH program memory is organized into rows and panels. Each row consists of 32 instructions, or 96 bytes. Each panel consists of 128 rows, or 4K x 24 instructions. Run Time Self Programming (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 set up a table pointer, then do a series of `TBLWT` instructions to load the write latches. Programming is performed by setting the special bits in the NVMCON register. 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 V_{DD} range.

6.5 Control Registers

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

- NVMCON
- NVMADR
- NVMADRU
- NVMKEY

6.5.1 NVMCON REGISTER

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

6.5.2 NVMADR REGISTER

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

6.5.3 NVMADRU REGISTER

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

6.5.4 NVMKEY REGISTER

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

6.6 Programming Operations

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

6.6.1 PROGRAMMING ALGORITHM FOR PROGRAM FLASH

The user can erase one row of program FLASH memory at a time. The user can program one block (4 instruction words) of FLASH memory at a time. The general process is:

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

6.6.2 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 6-1: ERASING A ROW OF PROGRAM MEMORY

```
; Setup NVMCON for erase operation, multi word write
; program memory selected, and writes enabled
       MOV     #0x4041,W0                  ;
       MOV     W0,NVMCON                   ; Init NVMCON SFR
; Init pointer to row to be ERASED
       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
       MOV     W0, NVMADR                  ; Intialize NVMADR SFR
       MOV     #0x55,W0                    ;
       MOV     #0xAA,W1                    ;
       MOV     W0,NVMKEY                   ; Write the 0x55 key
       MOV     W1,NVMKEY                   ; Write the 0xAA key
       NOP                                 ;
       BSET    NVMCON,#WR                  ; Start the erase sequence
       NOP                                 ; Insert two NOPs after the erase
       NOP                                 ; command is asserted
```

6.6.3 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 6-2: LOADING WRITE LATCHES

```
; Set up a pointer to the first program memory location to be written
; program memory selected, and writes enabled
       MOV     #0x0000,W0                      ;
       MOV     W0,TBLPAG                       ; Initialize PM Page Boundary SFR
       MOV     #0x6000,W0                      ; An example program memory address
; Perform the TBLWT instructions to write the latches
; 0th_program_word
       MOV     #LOW_WORD_0,W2                  ;
       MOV     #HIGH_BYTE_0,W3                 ;
       TBLWTL  W2,[W0]                         ; Write PM low word into program latch
       TBLWTH  W3,[W0++]                       ; Write PM high byte into program latch
; 1st_program_word
       MOV     #LOW_WORD_1,W2                  ;
       MOV     #HIGH_BYTE_1,W3                 ;
       TBLWTL  W2,[W0]                         ; Write PM low word into program latch
       TBLWTH  W3,[W0++]                       ; Write PM high byte into program latch
;  2nd_program_word
       MOV     #LOW_WORD_2,W2                  ;
       MOV     #HIGH_BYTE_2,W3                 ;
       TBLWTL  W2, [W0]                        ; Write PM low word into program latch
       TBLWTH  W3, [W0++]                      ; Write PM high byte into program latch
; 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 6-2, the contents of the upper byte of W3 has no effect.

6.6.4 INITIATING THE PROGRAMMING SEQUENCE

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

EXAMPLE 6-3: INITIATING A PROGRAMMING SEQUENCE

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

TABLE 6-1: NVM REGISTER MAP

File Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All RESETS
NVMCON	0760	WR	WREN	WRERR	—	—	—	—	—	—	PROGOP<6:0>							0000 0000 0000 0000
NVMADR	0762	NVMADR<15:0>																uuuu uuuu uuuu uuuu
NVMADRU	0764	—	—	—	—	—	—	—	—	NVMADR<23:16>								0000 0000 uuuu uuuu
NVMKEY	0766	—	—	—	—	—	—	—	—	KEY<7:0>								0000 0000 0000 0000

Legend: u = uninitialized bit

7.0 DATA EEPROM MEMORY

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

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

- NVMCON
- NVMADR
- NVMADRU
- NVMKEY

The EEPROM data memory allows read and write of single words and 16-word blocks. When interfacing to data memory, NVMADR in conjunction with the TBLPAG register, are used to address the EEPROM location being accessed. TBLRDL and TBLWTL 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 0x7FF000 to 0x7FFFFE.

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

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

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

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ 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.

7.1 Reading the Data EEPROM

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

EXAMPLE 7-1: DATA EEPROM READ

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

7.2 Erasing Data EEPROM

7.2.1 ERASING A BLOCK OF DATA EEPROM

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

EXAMPLE 7-2: DATA EEPROM BLOCK ERASE

```
; Select data EEPROM block, ERASE, WREN bits
    MOV     #4045,W0
    MOV     W0,NVMCON                  ; Initialize NVMCON SFR

; Start erase cycle by setting WR after writing key sequence
    MOV     #0x55,W0                   ;
    MOV     #0xAA,W1                   ;
    MOV     W0,NVMKEY                  ; Write the 0x55 key
    MOV     W1,NVMKEY                  ; Write the 0xAA key
    NOP                                ;
    BSET    NVMCON,#WR                 ; Initiate erase sequence

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

7.2.2 ERASING A WORD OF DATA EEPROM

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

EXAMPLE 7-3: DATA EEPROM WORD ERASE

```
; Select data EEPROM word, ERASE, WREN bits
    MOV     #4044,W0
    MOV     W0,NVMCON

; Start erase cycle by setting WR after writing key sequence
    MOV     #0x55,W0             ;
    MOV     #0xAA,W1             ;
    MOV     W0,NVMKEY            ; Write the 0x55 key
    MOV     W1,NVMKEY            ; Write the 0xAA key
    NOP                          ;
    BSET    NVMCON,#WR           ; Initiate erase sequence

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

7.3 Writing to the Data EEPROM

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

1. Erase data EEPROM word.
   a. Select word, data EEPROM, erase, and set WREN bit in NVMCON register.
   b. Write address of word to be erased into 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.

7.3.1 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 7-4.

EXAMPLE 7-4: DATA EEPROM WORD WRITE

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

; Operate key to allow write operation
    MOV         #0x55,W0
    MOV         #0xAA,W1
    MOV         W0,NVMKEY                    ; Write the 0x55 key
    MOV         W1,NVMKEY                    ; Write the 0xAA key
    NOP
    BSET        NVMCON,#WR                   ; Initiate program sequence

; 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
```

7.3.2 WRITING A BLOCK OF DATA EEPROM

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

EXAMPLE 7-5: DATA EEPROM BLOCK WRITE

```
    MOV     #LOW_ADDR_WORD,W0  ; Init pointer
    MOV     #HIGH_ADDR_WORD,W1
    MOV     W1,TBLPAG
    MOV     #data1,W2          ; Get 1st data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data2,W2          ; Get 2nd data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data3,W2          ; Get 3rd data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data4,W2          ; Get 4th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data5,W2          ; Get 5th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data6,W2          ; Get 6th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data7,W2          ; Get 7th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data8,W2          ; Get 8th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data9,W2          ; Get 9th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data10,W2         ; Get 10th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data11,W2         ; Get 11th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data12,W2         ; Get 12th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data13,W2         ; Get 13th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data14,W2         ; Get 14th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data15,W2         ; Get 15th data
    TBLWTL  W2,[W0]++          ; write data
    MOV     #data16,W2         ; Get 16th data
    TBLWTL  W2,[W0]++          ; write data. The NVMADR captures last table access address.
    MOV     #0x400A,W0         ; Select data EEPROM for multi word op
    MOV     W0,NVMCON          ; Operate Key to allow program operation
    MOV     #0x55,W0
    MOV     #0xAA,W1
    MOV     W0,NVMKEY          ; Write the 0x55 key
    MOV     W1,NVMKEY          ; Write the 0xAA key
    BSET    NVMCON,#WR         ; Start write cycle
```

7.4 Write Verify

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

7.5 Protection Against Spurious Write

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

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

8.0 I/O PORTS

All of the device pins (except VDD, VSS, $\overline{MCLR}$, 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.

8.1 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 are shown in Table 8-1.

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. 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 modifies the contents of the LATA register.

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

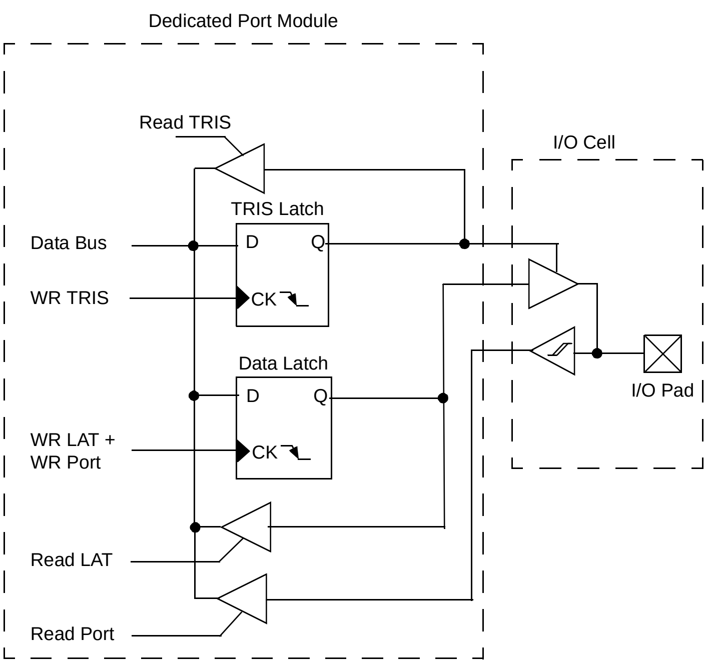


TABLE 8-1: PORTA REGISTER MAP

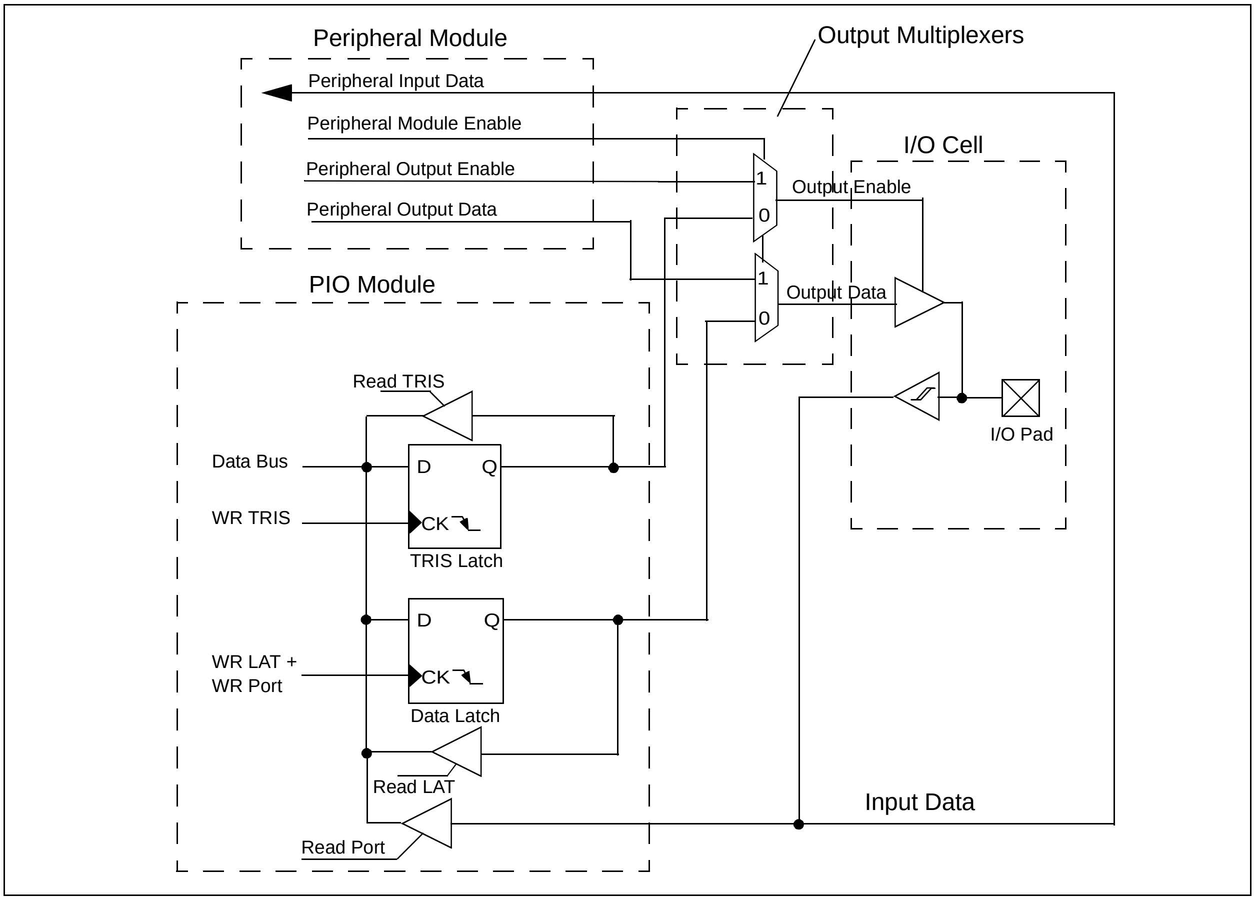
SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
TRISA	02C0	TRISA15	TRISA14	—	—	—	TRISA10	TRISA9	—	—	—	—	—	—	—	—	—	1100 0110 0000 0000
PORTA	02C2	RA15	RA14	—	—	—	RA10	RA9	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000
LATA	02C4	LATA15	LATA14	—	—	—	LATA10	LATA9	—	—	—	—	—	—	—	—	—	0000 0000 0000 0000

Legend: u = uninitialized bit

A parallel I/O (PIO) port that shares a pin with a peripheral is, in general, subservient to the peripheral. The peripheral's output buffer data and control signals are provided to a pair of multiplexers. The multiplexers select whether the peripheral or the associated port has ownership of the output data and control signals of the I/O pad cell. Figure 8-2 shows how ports are shared with other peripherals, and the associated I/O cell (pad) to which they are connected. Table 8-2 through Table 8-7 show the formats of the registers for the shared ports, PORTB through PORTG.

Note:	The actual bits in use vary between devices.

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



8.2 Configuring Analog Port Pins

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

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

TABLE 8-2: 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

Legend: u = uninitialized bit

TABLE 8-3: 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	—	—	—	—	—	—	—	—	—	TRISC3	—	TRISC1	—	1110 0000 0000 1010
PORTC	02CE	RC15	RC14	RC13	—	—	—	—	—	—	—	—	—	RC3	—	RC1	—	0000 0000 0000 0000
LATC	02D0	LATC15	LATC14	LATC13	—	—	—	—	—	—	—	—	—	LATC3	—	LATC1	—	0000 0000 0000 0000

Legend: u = uninitialized bit

TABLE 8-4: 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

Legend: u = uninitialized bit

TABLE 8-5: 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	0000 0011 1111 1111
PORTE	02DB	—	—	—	—	—	—	RE9	RE8	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0	0000 0000 0000 0000
LATE	02DC	—	—	—	—	—	—	LATE9	LATE8	LATE7	LATE6	LATE5	LATE4	LATE3	LATE2	LATE1	LATE0	0000 0000 0000 0000

Legend: u = uninitialized bit

TABLE 8-6: 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	—	—	—	—	—	—	—	TRISF8	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	0000 0001 1111 1111
PORTF	02E0	—	—	—	—	—	—	—	RF8	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	0000 0000 0000 0000
LATF	02E2	—	—	—	—	—	—	—	LATF8	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0	0000 0000 0000 0000

Legend: u = uninitialized bit

TABLE 8-7: 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	—	—	—	—	—	—	TRISG9	TRISG8	TRISG7	TRISG6	—	—	TRISG3	TRISG2	TRISG1	TRISG0	0000 0011 1100 1111
PORTG	02E6	—	—	—	—	—	—	RG9	RG8	RG7	RG6	—	—	RG3	RG2	RG1	RG0	0000 0000 0000 0000
LATG	02E8	—	—	—	—	—	—	LATG9	LATG8	LATG7	LATG6	—	—	LATG3	LATG2	LATG1	LATG0	0000 0000 0000 0000

Legend: u = uninitialized bit

8.3 Input Change Notification Module

The Input Change Notification module provides the dsPIC30F devices the ability to generate interrupt requests to the processor in response to a change of state on selected input pins. This module is capable of detecting input change of states even in SLEEP mode, when the clocks are disabled. There are up to 22 external signals (CN0 through CN21) that may be selected (enabled) for generating an interrupt request on a change of state.

TABLE 8-8: 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

Legend: u = uninitialized bit

TABLE 8-9: 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	—	—	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	—	—	CN21PUE	CN20PUE	CN19PUE	CN18PUE	CN17PUE	CN16PUE	0000 0000 0000 0000

Legend: u = uninitialized bit

9.0 TIMER1 MODULE

This section describes the 16-bit General Purpose (GP) Timer1 module and associated Operational modes. Figure 9-1 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 9-1 presents a block diagram of the 16-bit timer module.

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

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

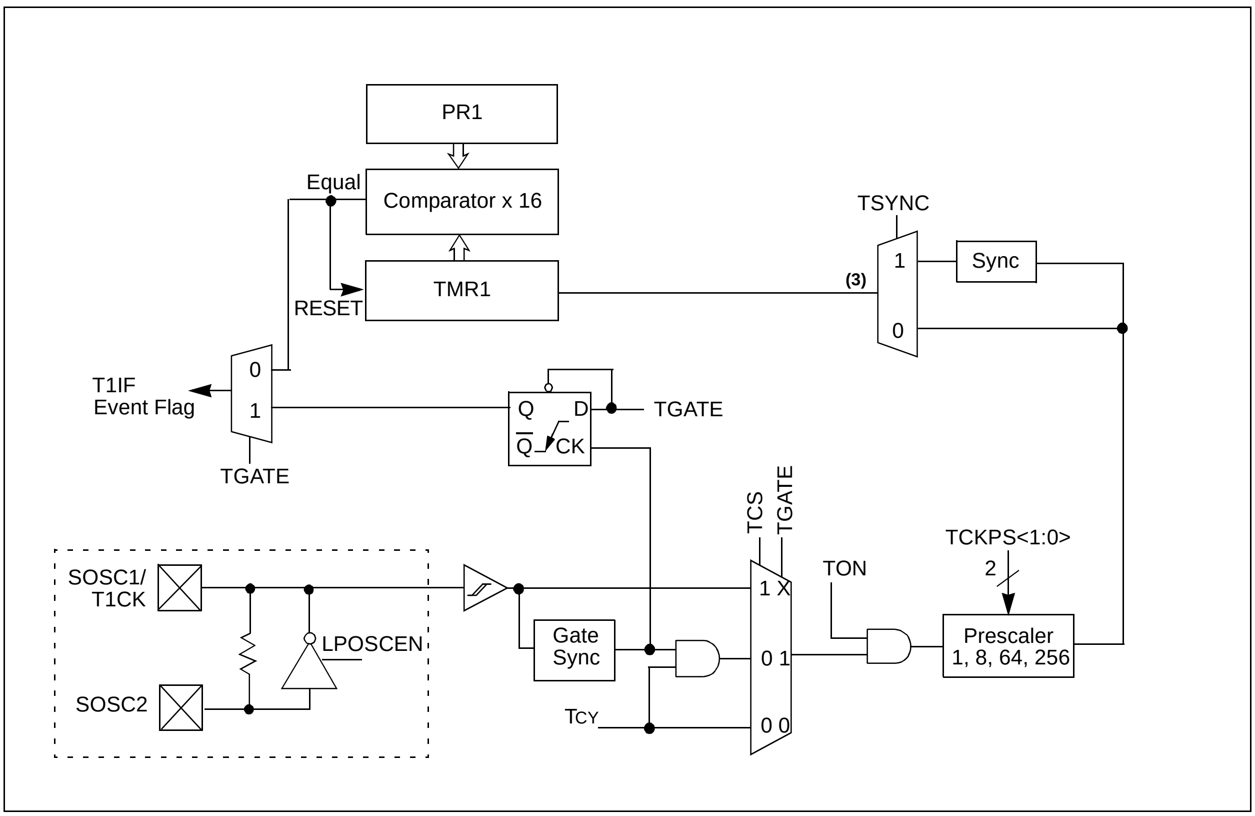
16-bit Synchronous Counter Mode: In the 16-bit Synchronous Counter mode, the timer increments on the rising edge of the applied external clock signal, which is synchronized with the internal phase clocks. The timer counts up to a match value preloaded in PR1, then resets to 0 and continues.

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

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

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

FIGURE 9-1: 16-BIT TIMER1 MODULE BLOCK DIAGRAM



9.1 Timer Gate Operation

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

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

9.2 Timer Prescaler

The input clock (F_{OSC}/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.

9.3 Timer Operation During SLEEP Mode

During CPU SLEEP mode, the timer will operate if:

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

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

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

9.4 Timer Interrupt

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

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

Enabling an interrupt is accomplished via the respective timer interrupt enable bit, T1IE. The timer interrupt enable bit is located in the IEC0 control register in the Interrupt Controller.

9.5 Real-Time Clock

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

- Operation from 32 KHz LP oscillator
- 8-bit prescaler
- Low power
- Real-Time Clock Interrupts
- These Operating modes are determined by setting the appropriate bit(s) in the T1CON Control register

9.5.1 RTC OSCILLATOR OPERATION

When 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.

9.5.2 RTC 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 9-1: TIMER1 REGISTER MAP

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

Legend: u = uninitialized bit

NOTES:

10.0 TIMER2/3 MODULE

This section describes the 32-bit General Purpose (GP) Timer module (Timer2/3) and associated Operational modes. Figure 10-1 depicts the simplified block diagram of the 32-bit Timer2/3 Module. Figure 10-2 and Figure 10-3 show Timer2/3 configured as two independent 16-bit timers, Timer2 and Timer3, respectively.

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

- Input Capture
- Output Compare/Simple PWM

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

The 32-bit timer has the following modes:

- Two independent 16-bit timers (Timer2 and Timer3) with all 16-bit Operating modes (except Asynchronous Counter mode)
- Single 32-bit Timer operation
- Single 32-bit Synchronous Counter

Further, the following operational characteristics are supported:

- ADC Event Trigger
- Timer Gate Operation
- Selectable Prescaler Settings
- Timer Operation during IDLE and SLEEP modes
- Interrupt on a 32-bit Period Register Match

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

For 32-bit timer/counter operation, Timer2 is the LS Word and Timer3 is the MS Word of the 32-bit timer.

Note:	For 32-bit timer operation, T3CON control bits are ignored. Only T2CON control bits are used for setup and control. 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) and the interrupt is enabled with the Timer3 interrupt enable bit (T3IE).

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

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

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

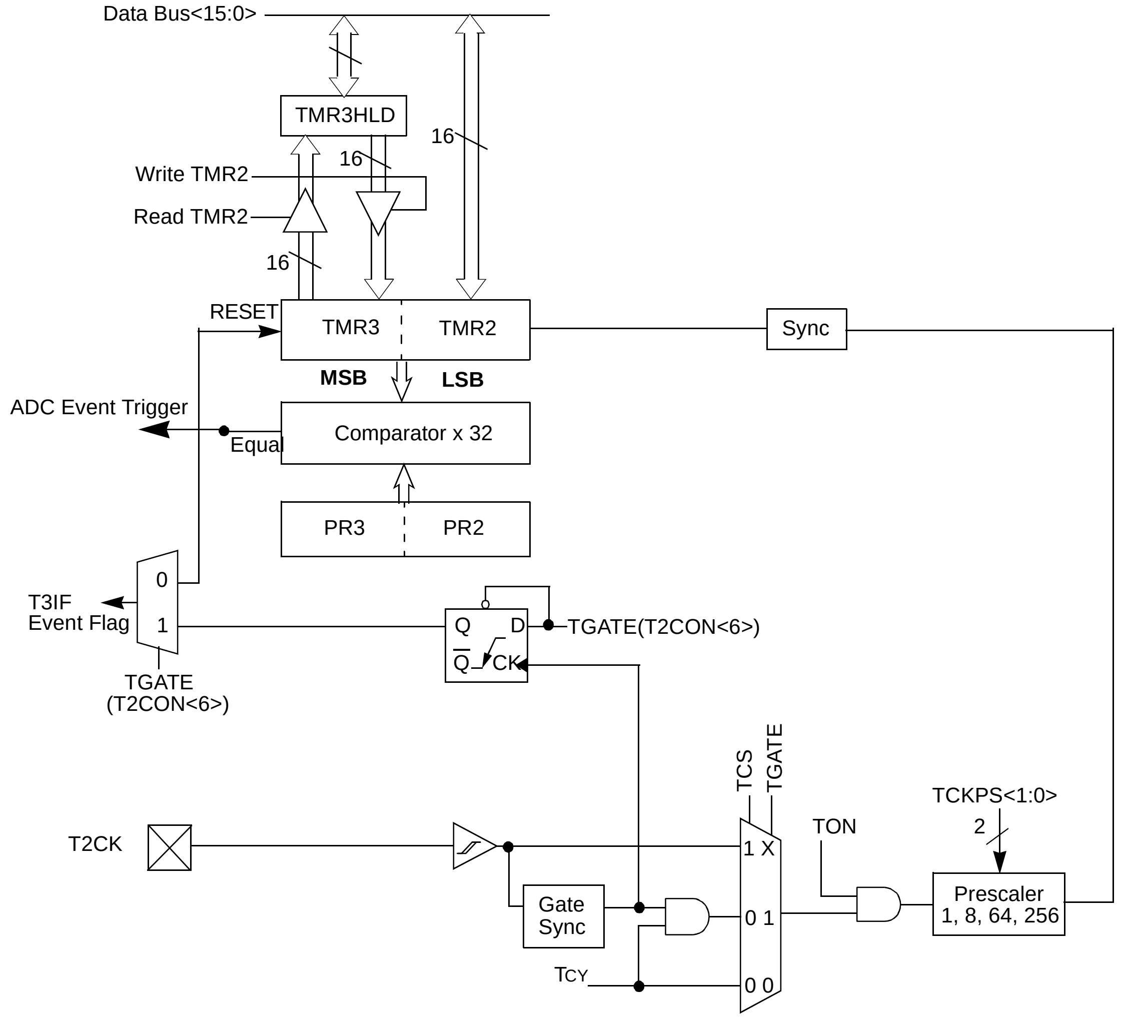
For synchronous 32-bit reads of the Timer2/Timer3 pair, reading the LS word (TMR2 register) will cause the MS word to be read and latched into a 16-bit holding register, termed TMR3HLD.

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



Note 1: Timer configuration bit T32, T2CON(<3>) must be set to 1 for a 32-bit timer/counter operation. All control bits are respective to the T2CON register.

FIGURE 10-2: 16-BIT TIMER2 BLOCK DIAGRAM

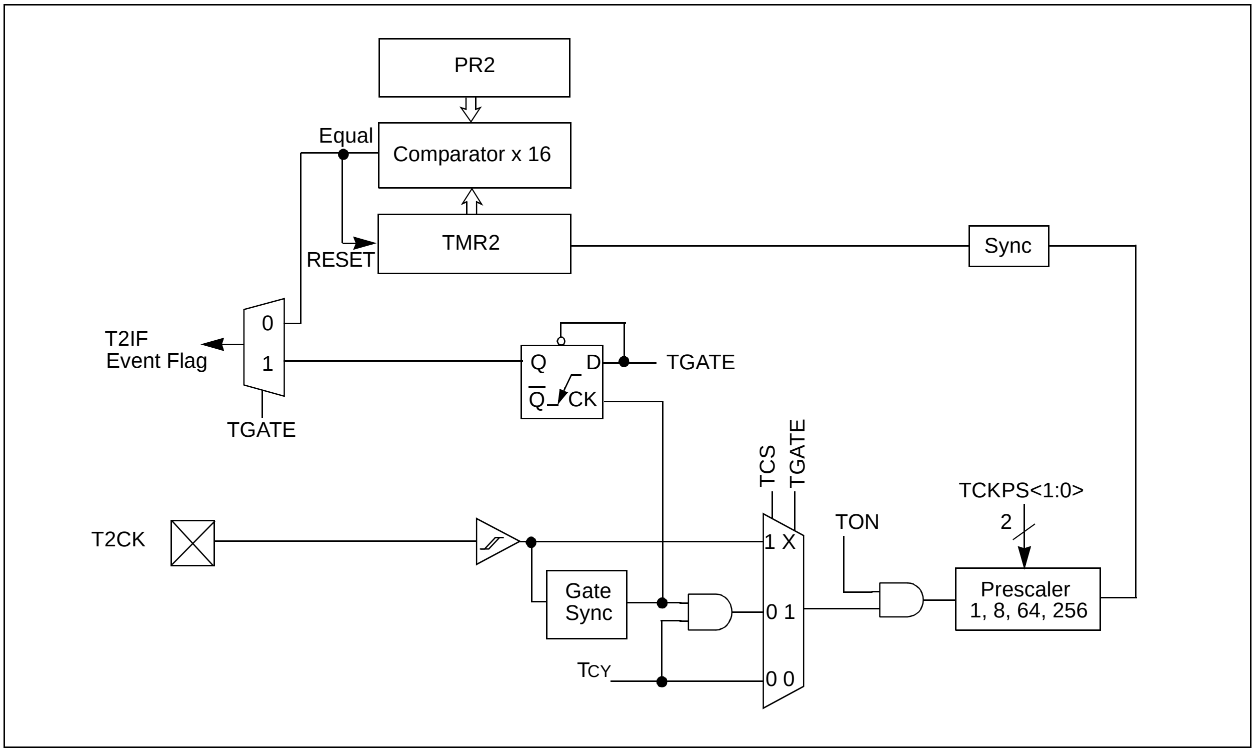
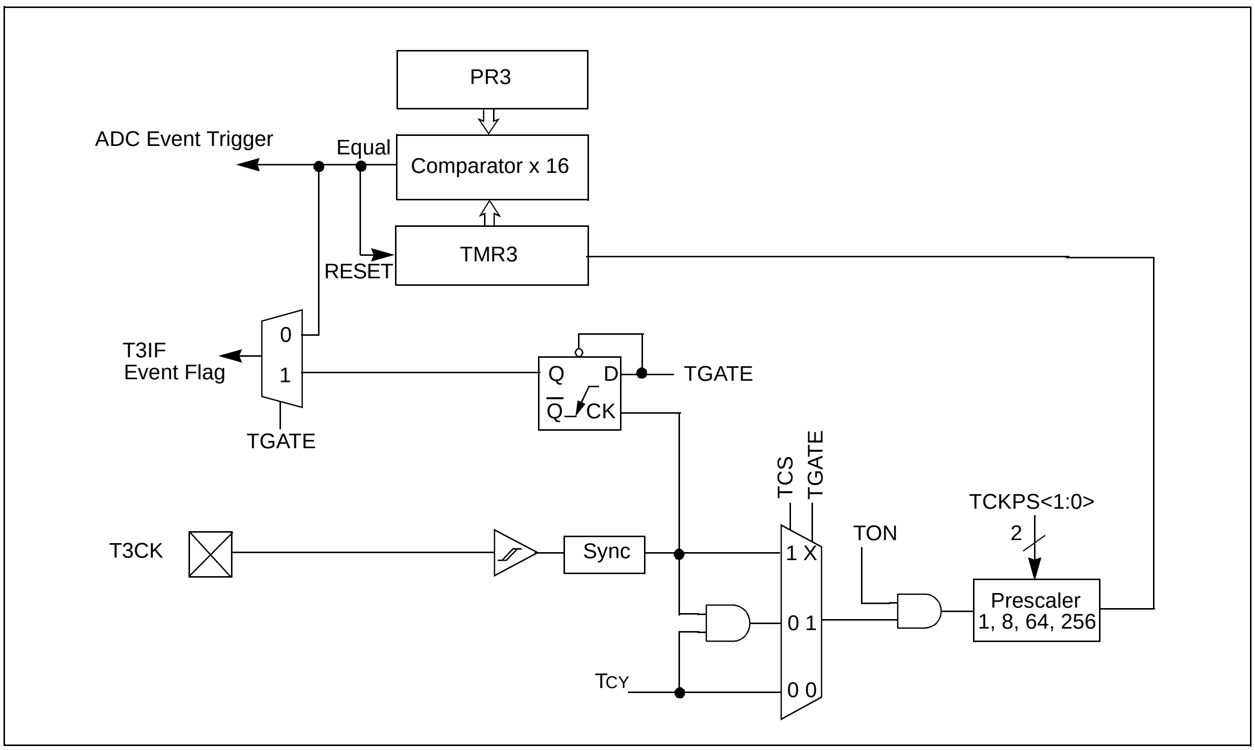


FIGURE 10-3: 16-BIT TIMER3 BLOCK DIAGRAM



10.1 Timer Gate Operation

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

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

10.2 ADC Event Trigger

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

10.3 Timer Prescaler

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

- a write to the TMR2/TMR3 register
- 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.

10.4 Timer Operation During SLEEP Mode

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

10.5 Timer Interrupt

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

Enabling an interrupt is accomplished via the respective timer interrupt enable bit, T3IE (IEC0<7>).

TABLE 10-1: TIMER2/3 REGISTER MAP

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

Legend: u = uninitialized bit

NOTES:

11.0 TIMER4/5 MODULE

This section describes the second 32-bit General Purpose (GP) Timer module (Timer4/5) and associated Operational modes. Figure 11-1 depicts the simplified block diagram of the 32-bit Timer4/5 Module. Figure 11-2 and Figure 11-3 show Timer4/5 configured as two independent 16-bit timers, Timer4 and Timer5, respectively.

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

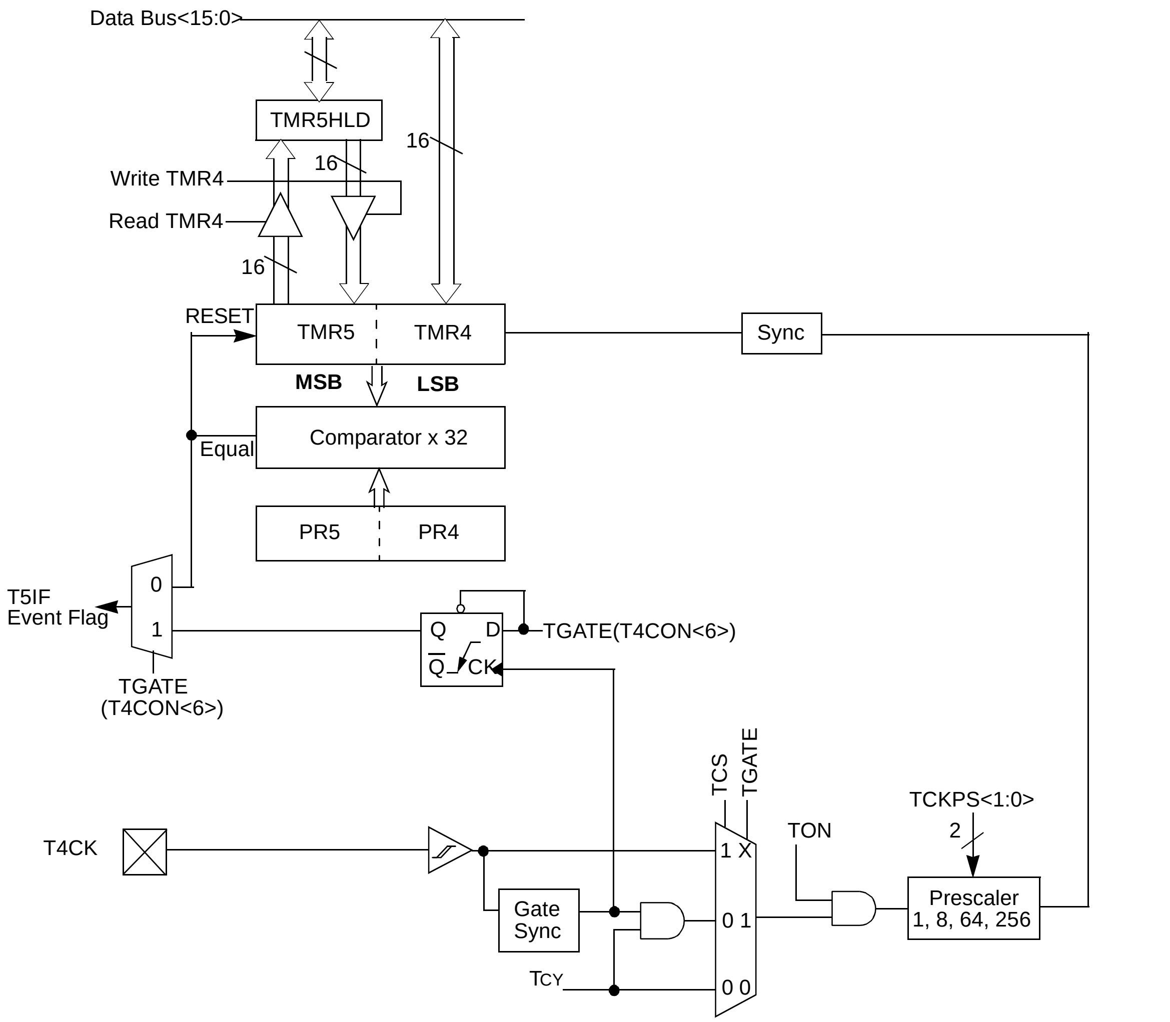
- The Timer4/5 module does not support the ADC Event Trigger feature
- Timer4/5 can not be utilized by other peripheral modules such as Input Capture and Output Compare

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

For 32-bit timer/counter operation, Timer4 is the LS Word and Timer5 is the MS Word of the 32-bit timer.

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

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



Note 1: Timer configuration bit T32, T4CON(<3>) must be set to '1' for a 32-bit timer/counter operation. All control bits are respective to the T4CON register.

FIGURE 11-2: 16-BIT TIMER4 BLOCK DIAGRAM

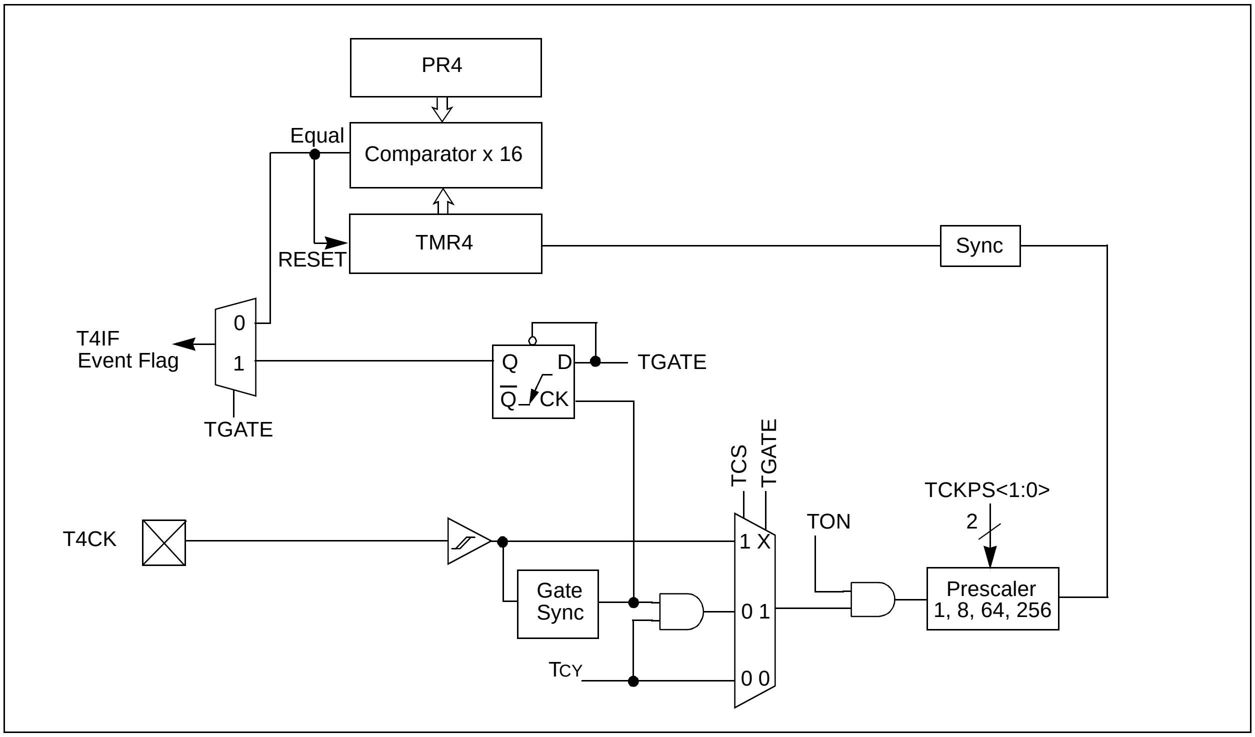


FIGURE 11-3: 16-BIT TIMER5 BLOCK DIAGRAM

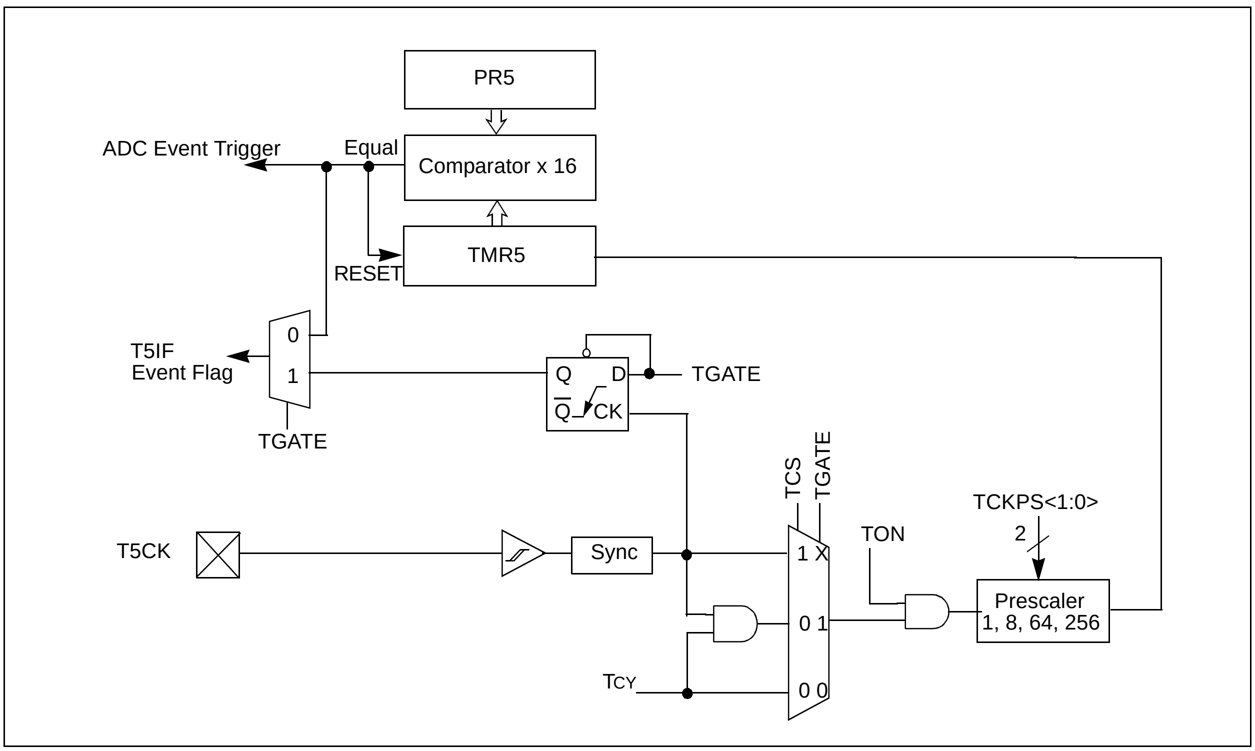


TABLE 11-1: TIMER4/5 REGISTER MAP

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

Legend: u = uninitialized bit

NOTES:

12.0 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 12-1 depicts a block diagram of the Input Capture Module. Input capture is useful for such modes as:

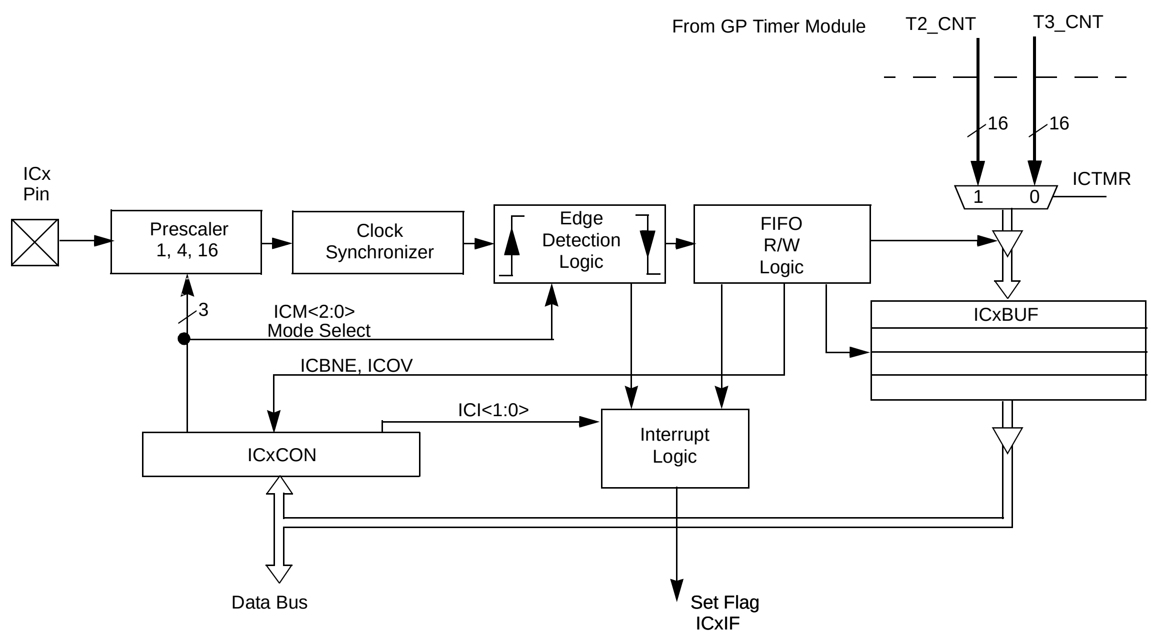
- Frequency/Period/Pulse Measurements
- Additional sources of External Interrupts

The key operational features of the Input Capture module are:

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

These operating modes are determined by setting the appropriate bits in the ICxCON register (where x = 1,2,...,N). The dsPIC devices contain up to 8 capture channels, i.e., the maximum value of N is 8.

FIGURE 12-1: INPUT CAPTURE MODE BLOCK DIAGRAM



Note 1: Where ‘x’ is shown, reference is made to the registers or bits associated to the respective input capture channels 1 through N.

12.1 Simple Capture Event Mode

The simple capture events in the dsPIC30F product family are:

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

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

12.1.1 CAPTURE PRESCALER

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

12.1.2 CAPTURE BUFFER OPERATION

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

- 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.

12.1.3 TIMER2 AND TIMER3 SELECTION MODE

The input capture module consists of up to 8 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.

12.1.4 HALL SENSOR MODE

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

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

12.2 Input Capture Operation During SLEEP and IDLE Modes

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

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

12.2.1 INPUT CAPTURE IN CPU SLEEP MODE

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

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

12.2.2 INPUT CAPTURE IN CPU IDLE MODE

CPU IDLE mode allows input capture module operation with full functionality. In the CPU IDLE mode, the interrupt mode selected by the 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.

12.3 Input Capture Interrupts

The input capture channels have the ability to generate an interrupt, based upon the selected number of capture events. The selection number is set by control bits 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 12-1: INPUT CAPTURE REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
IC1BUF	0140	Input 1 Capture Register																uuuu uuuu uuuu uuuu
IC1CON	0142	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC2BUF	0144	Input 2 Capture Register																uuuu uuuu uuuu uuuu
IC2CON	0146	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC3BUF	0148	Input 3 Capture Register																uuuu uuuu uuuu uuuu
IC3CON	014A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC4BUF	014C	Input 4 Capture Register																uuuu uuuu uuuu uuuu
IC4CON	014E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC5BUF	0150	Input 5 Capture Register																uuuu uuuu uuuu uuuu
IC5CON	0152	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC6BUF	0154	Input 6 Capture Register																uuuu uuuu uuuu uuuu
IC6CON	0156	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC7BUF	0158	Input 7 Capture Register																uuuu uuuu uuuu uuuu
IC7CON	015A	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000
IC8BUF	015C	Input 8 Capture Register																uuuu uuuu uuuu uuuu
IC8CON	015E	—	—	ICSIDL	—	—	—	—	—	ICTMR	ICI<1:0>		ICOV	ICBNE	ICM<2:0>			0000 0000 0000 0000

Legend: u = uninitialized bit

NOTES:

13.0 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 13-1 depicts a block diagram of the Output Compare module.

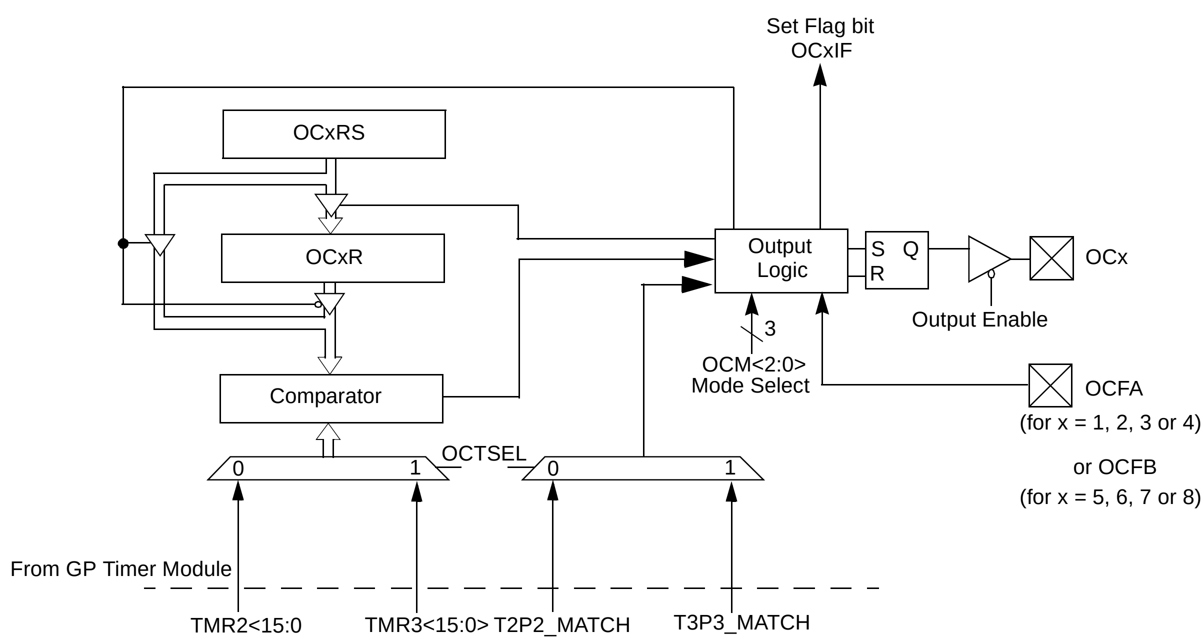
The key operational features of the Output Compare module include:

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

These operating modes are determined by setting the appropriate bits in the 16-bit OCxCON SFR (where x = 1,2,3,...,N). The 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.

FIGURE 13-1: OUTPUT COMPARE MODE BLOCK DIAGRAM



Note: Where 'x' is shown, reference is made to the registers associated with the respective output compare channels 1 through N.

13.1 Timer2 and Timer3 Selection Mode

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

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

13.2 Simple Output Compare Match Mode

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

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

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

13.3 Dual Output Compare Match Mode

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

- Single Output Pulse mode
- Continuous Output Pulse mode

13.3.1 SINGLE PULSE MODE

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

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

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

13.3.2 CONTINUOUS PULSE MODE

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

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

13.4 Simple PWM Mode

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

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

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

13.4.1 INPUT PIN FAULT PROTECTION FOR PWM

When control bits OCM<2:0> (OCxCON<2:0>) = `111`, the selected output compare channel is again configured for the PWM mode of operation, with the additional feature of input fault protection. While in this mode, if a logic 0 is detected on the OCFA/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.

13.4.2 PWM PERIOD

The PWM period is specified by writing to the PRx register. The PWM period can be calculated using the following formula.

EQUATION 13-1:

$$\text{PWM period} = [(\text{PRx}) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{TMRx prescale value})$$

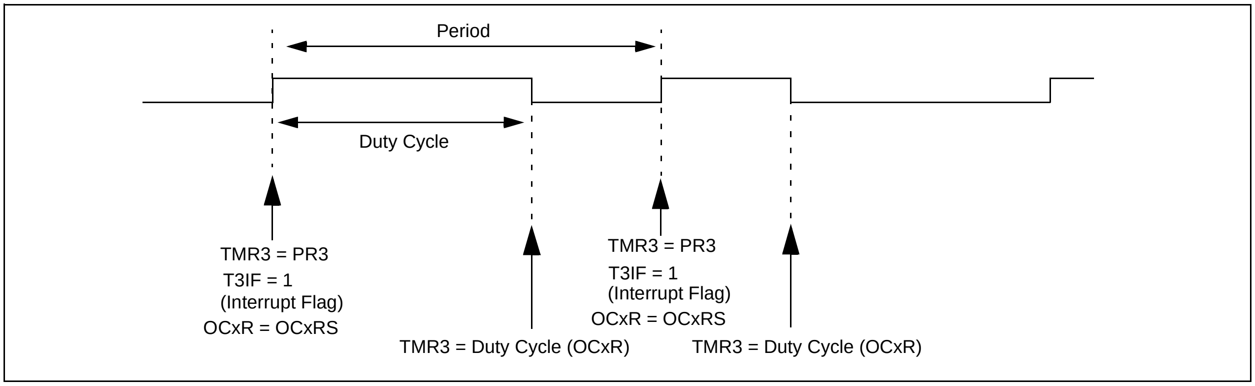
PWM frequency is defined as 1 / [PWM period].

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

See Figure 13-2 for key PWM period comparisons. Timer3 is referred to in the figure for clarity.

FIGURE 13-2: PWM OUTPUT TIMING



13.5 Output Compare Operation During CPU SLEEP Mode

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

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

13.6 Output Compare Operation During CPU IDLE Mode

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

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

The selected time-base for the output compare can be configured for:

- Internal Instruction Cycle
- External Synchronous

13.7 Output Compare Interrupts

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

For all modes except the PWM mode, when a compare event occurs, the respective interrupt flag (OCxIF) 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 13-1: OUTPUT COMPARE REGISTER MAP

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

Legend: u = uninitialized bit

14.0 QUADRATURE ENCODER INTERFACE (QEI) MODULE

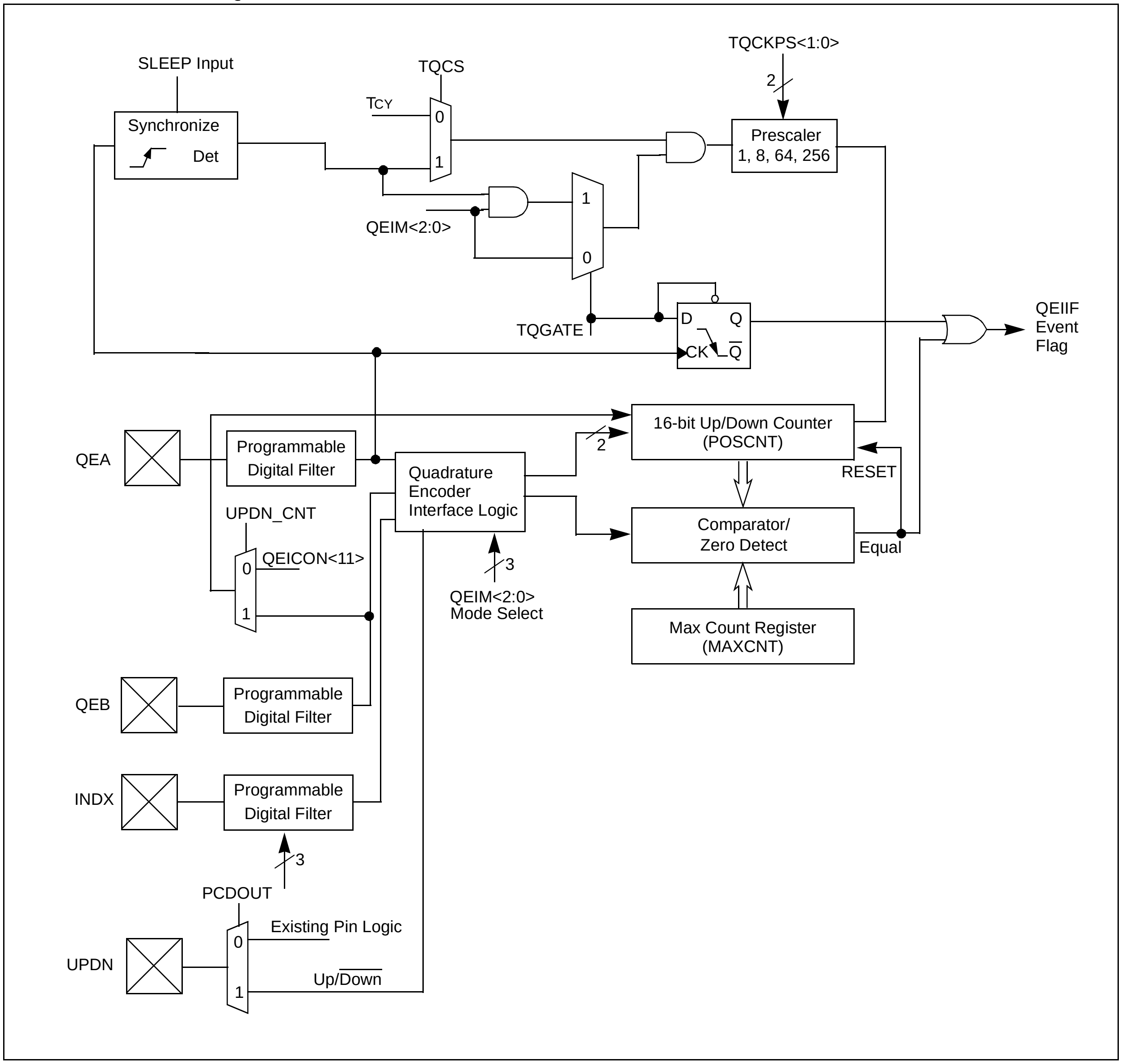
This section describes the Quadrature Encoder Interface (QEI) module and associated Operational modes. The QEI module provides the interface to incremental encoders for obtaining motor positioning data. Incremental encoders are very useful in motor control applications.

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 14-1 depicts the Quadrature Encoder Interface block diagram.

FIGURE 14-1: QUADRATURE ENCODER INTERFACE BLOCK DIAGRAM



14.1 Quadrature Encoder Interface Logic

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

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

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

14.2 16-bit Up/Down Position Counter Mode

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

14.2.1 POSITION COUNTER ERROR CHECKING

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

14.2.2 POSITION COUNTER RESET

The position counter RESET enable bit, POSRES (QEI<2>) controls whether the position counter is reset when the index pulse is detected. This bit is only applicable when QEIM<2:0> = '`100`' or '`110`'.

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

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

14.2.3 COUNT DIRECTION STATUS

As mentioned in the previous section, the QEI logic generates an UPDN signal, based upon the relationship between Phase A and Phase B. In addition to the output pin, the state of this internal UPDN signal is supplied to 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.

14.3 Position Measurement Mode

There are two Measurement modes which are supported and are termed x2 and x4. These modes are selected by the QEIM<2:0> 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.

14.4 Programmable Digital Noise Filters

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

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

QEA and QEB is programmed by bits QECK<2:0> (DFLTCON<6:4>) and is derived from the base instruction cycle T_{CY}. For the index channel the clock divide frequency for the digital filter is programmed by bits INDCK<2:0> (DFLTCON<2:0>), and is also derived from the base instruction cycle T_{CY}.

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.

14.5 Alternate 16-bit Timer/Counter

When the QEI module is not configured for the QEI mode QEIM<2:0> = `001`, the module can be configured 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 UPDN = 1, the timer will count up. When UPDN = 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 the UPDN control/status bit (QEICON<11>), or the QEB pin state. When UPDN_CNT = 1, the timer count direction is controlled from the QEB pin. Likewise, when UPDN_CNT = 0, the timer count direction is controlled by the UPDN bit.

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

14.6 QEI Module Operation During CPU SLEEP Mode

14.6.1 QEI OPERATION DURING CPU SLEEP MODE

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

14.6.2 TIMER OPERATION DURING CPU SLEEP MODE

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

14.7 QEI Module Operation During CPU IDLE Mode

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

14.7.1 QEI OPERATION DURING CPU IDLE MODE

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

14.7.2 TIMER OPERATION DURING CPU IDLE MODE

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

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

14.8 Quadrature Encoder Interface Interrupts

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

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

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

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

TABLE 14-1: QEI REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
QEICON	0122	CNTERR	—	QEISIDL	INDX	UPDN	QEIM2	QEIM1	QEIM0	SWPAB	PCDOUT	TQGATE	TQCKPS1	TQCKPS0	POSRES	TQCS	UPDN_CNT	0000 0000 0000 0000
DFLTCON	0124	—	—	—	—	—	—	—	—	QEOUT	QECK2	QECK1	QECK0	INDOUT	INDCK2	INDCK1	INDCK0	0000 0000 0000 0000
POSCNT	0126	Position Counter<15:0>																0000 0000 0000 0000
MAXCNT	0128	Maximun Count<15:0>																1111 1111 1111 1111

Legend: u = uninitialized bit

NOTES:

15.0 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

This module contains 4 duty cycle generators, numbered 1 through 4. The module has 8 PWM output pins, numbered PWM1H/PWM1L through PWM4H/PWM4L. The eight I/O pins are grouped into High/Low numbered pairs, denoted by the suffix H or L, respectively. For complementary loads, the low PWM pins are always the complement of the corresponding High I/O pin.

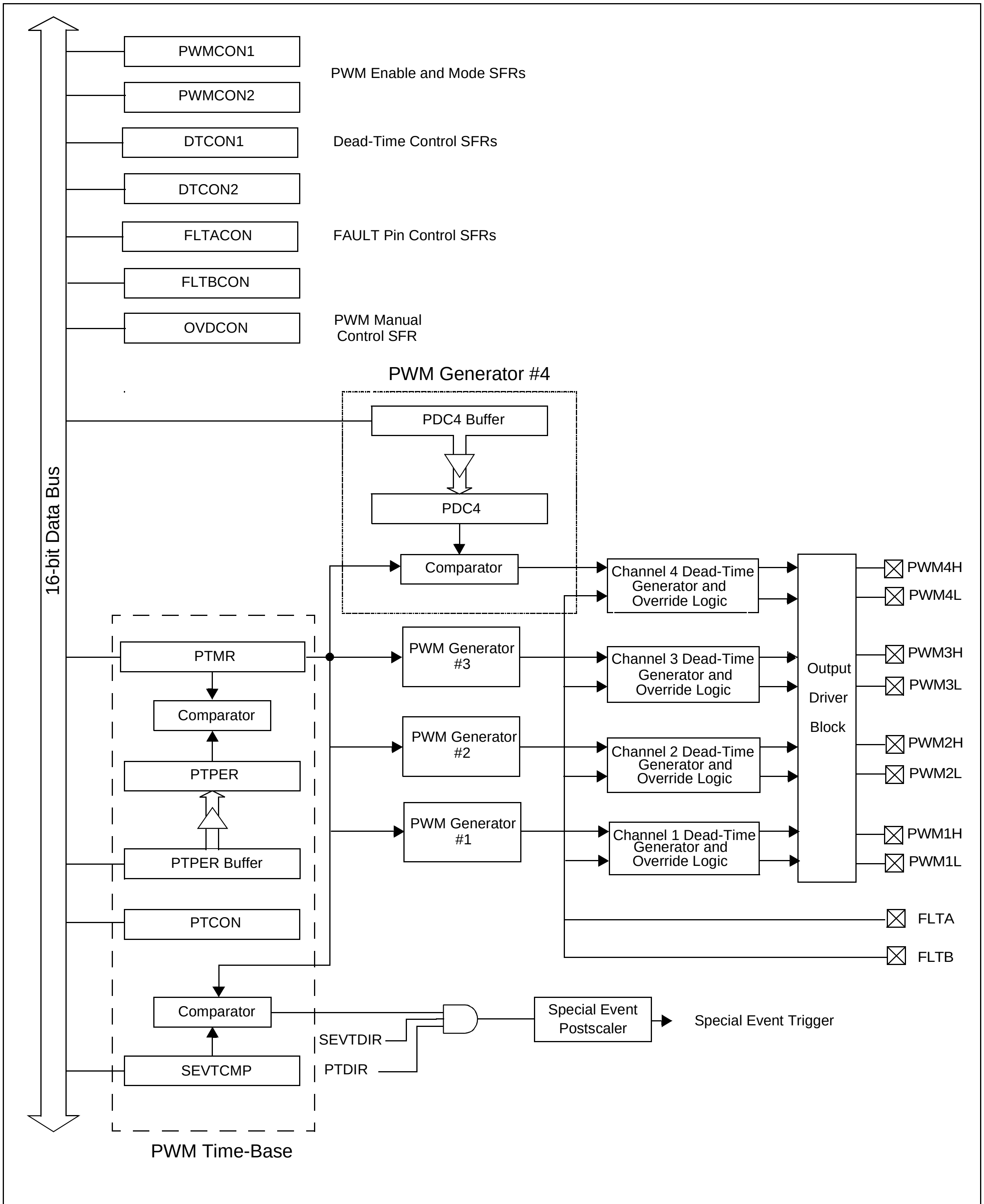
There are two versions of the PWM module depending on the particular dsPIC30F device selected. There is an 8-output module in some devices and a 6-output PWM module in others.

Simplified block diagrams of the 8-output and 6-output Motor Control PWM modules are shown in Figure 15-1 and Figure 15-2 , respectively.

TABLE 15-1: FEATURE SUMMARY: 6-OUTPUT PWM VS. 8-OUTPUT PWM

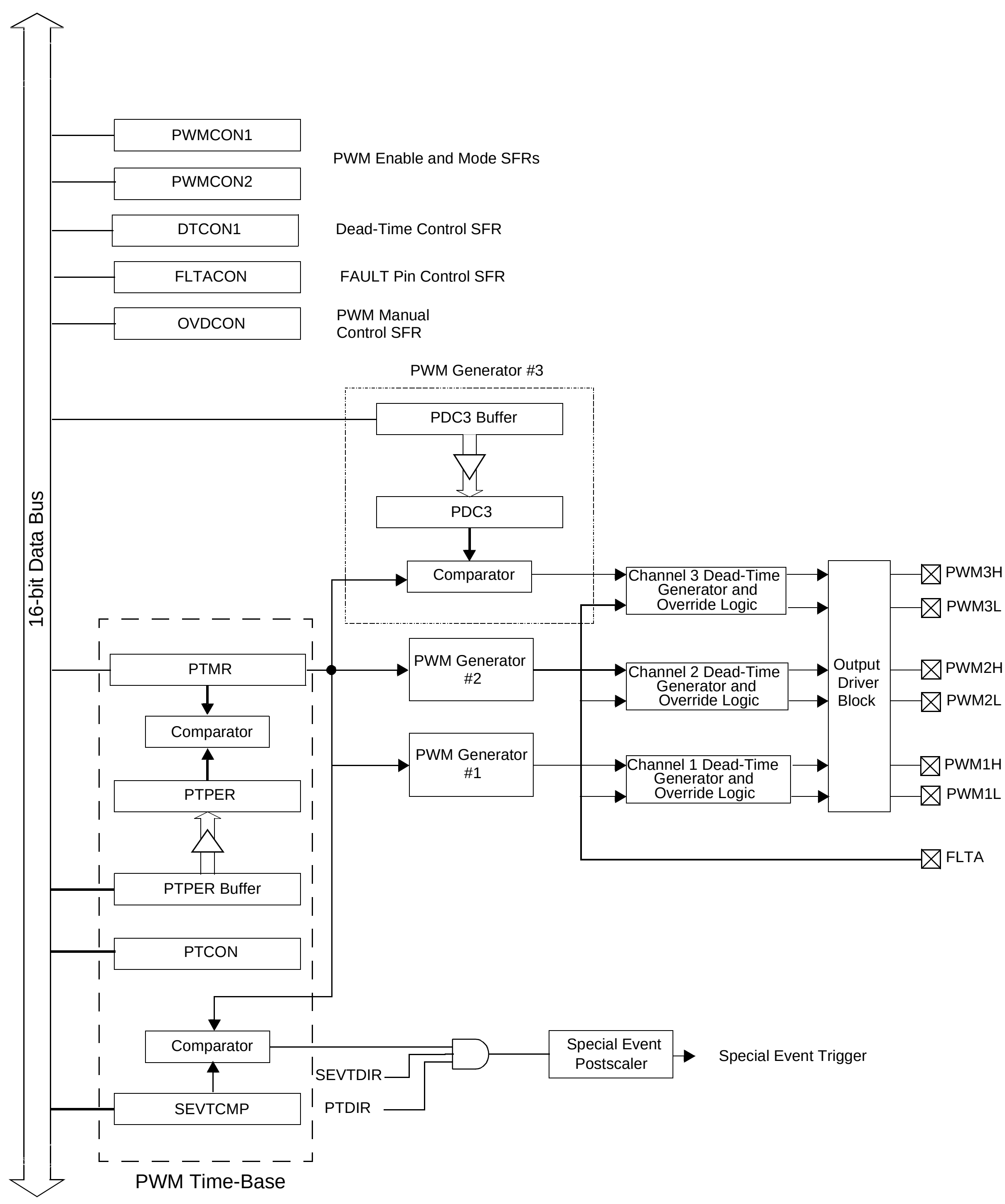
Feature	6-Output PWM Module	8-Output PWM Module
I/O Pins	6	8
PWM Generators	3	4
FAULT Input Pins	1	2
Dead-Time Generators	1	2

FIGURE 15-1: 8-OUTPUT PWM MODULE BLOCK DIAGRAM



Note: Details of PWM Generator #1, #2, and #3 not shown for clarity.

FIGURE 15-2: 6-OUTPUT PWM BLOCK DIAGRAM



Note: Details of PWM Generator #1 and #2 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.

15.1 PWM Time-Base

The PWM time-base is provided by a 15-bit timer with a prescaler and postscaler. The time-base is accessible via the PTMR SFR. PTMR<15> is a read only status bit, PTDIR, that indicates the present count direction of the PWM time-base. If PTDIR is cleared, PTMR is counting upwards, 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.

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).

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.

15.1.1 FREE RUNNING MODE

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

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

15.1.2 SINGLE SHOT MODE

In the Single Shot 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.

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

15.1.3 CONTINUOUS UP/DOWN 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.

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.

15.1.4 DOUBLE UPDATE MODE

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

The Double Update mode provides two additional functions to the user. First, the control loop bandwidth is doubled because the PWM duty cycles can be updated, twice per period. 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.

15.1.5 PWM TIME-BASE PRESCALER

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

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

The PTMR register is not cleared when PTCON is written.

15.1.6 PWM TIME-BASE POSTSCALER

The match output of PTMR can optionally be 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.

15.2 PWM Period

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

- Free Running and Single Shot modes: When the PTMR register is reset to zero after a match with the PTPER register.
- 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 15-1: PWM PERIOD

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

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 15-1.

The maximum resolution (in bits) for a given device oscillator and PWM frequency can be determined from the following formula:

EQUATION 15-2: PWM RESOLUTION

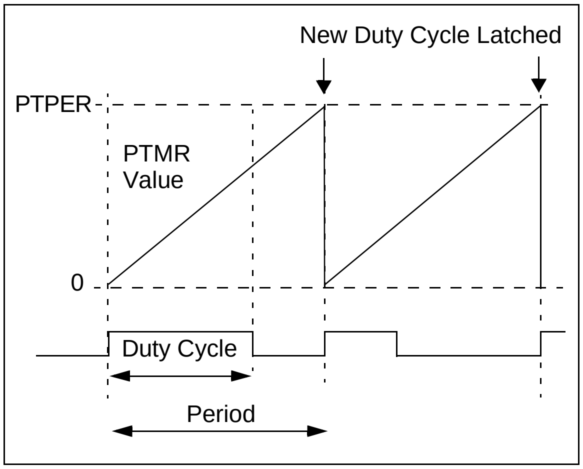
$$Resolution = \frac{\log(2 \bullet T_{PWM} / T_{CY})}{\log(2)}$$

15.3 Edge Aligned PWM

Edge aligned PWM signals are produced by the module when the PWM time-base is in the Free Running or Single Shot mode. For edge aligned PWM outputs, the output 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 15-3). 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 15-3: EDGE ALIGNED PWM



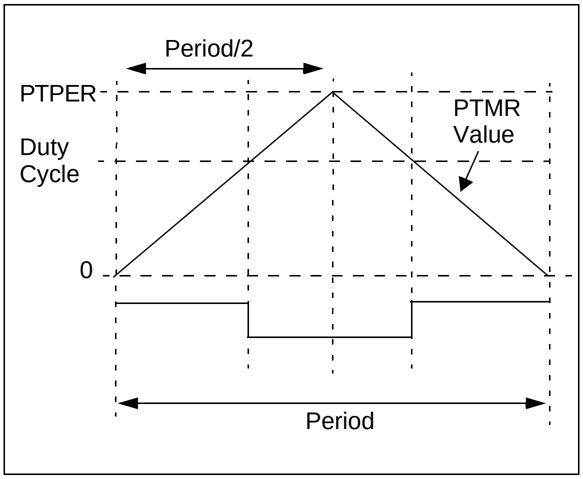
15.4 Center Aligned PWM

Center aligned PWM signals are produced by the module when the PWM time-base is configured in an Up/Down Counting mode (see Figure 15-4).

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 15-4: CENTER ALIGNED PWM



15.5 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.

15.5.1 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).

15.6 Complementary PWM Operation

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

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

- PDC1 register controls PWM1H/PWM1L outputs
- PDC2 register controls PWM2H/PWM2L outputs
- PDC3 register controls PWM3H/PWM3L outputs
- PDC4 register controls PWM4H/PWM4L 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.

15.7 Dead-Time Generators

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

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

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

15.7.1 DEAD-TIME GENERATORS

Each complementary output pair for the PWM module has a 6-bit down counter that is used to produce the dead-time insertion. As shown in Figure 15-5, each dead-time unit has a rising and falling edge detector connected to the duty cycle comparison output.

15.7.2 DEAD-TIME ASSIGNMENT

The DTCON2 SFR contains control bits that allow the dead-times to be assigned to each of the complementary outputs. Table 15-2 summarizes the function of each dead-time selection control bit.

TABLE 15-2: DEAD-TIME SELECTION BITS

Bit	Function
DTS1A	Selects PWM1L/PWM1H active edge dead-time.
DTS1I	Selects PWM1L/PWM1H inactive edge dead-time.
DTS2A	Selects PWM2L/PWM2H active edge dead-time.
DTS2I	Selects PWM2L/PWM2H inactive edge dead-time.
DTS3A	Selects PWM3L/PWM3H active edge dead-time.
DTS3I	Selects PWM3L/PWM3H inactive edge dead-time.
DTS4A	Selects PWM4L/PWM4H active edge dead-time.
DTS4I	Selects PWM4L/PWM4H inactive edge dead-time.

15.7.3 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:

- T_{CY}
- $2T_{CY}$
- $4T_{CY}$
- $8T_{CY}$

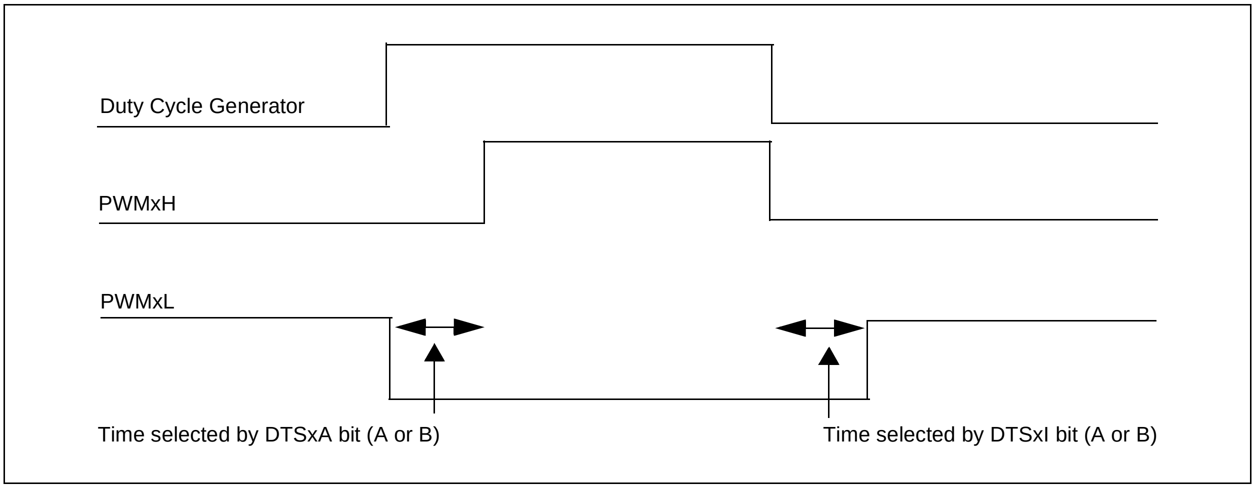
After the prescaler values are selected, the dead-time for each unit is adjusted by loading two 6-bit unsigned values into the DTCON1 special function register.

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.

FIGURE 15-5: DEAD-TIME TIMING DIAGRAM



15.8 Independent PWM Output

An independent PWM Output mode is required for driving certain types of loads. A particular PWM output pair is in the Independent Output mode when the corresponding 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. 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

15.9 Single Pulse PWM Operation

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

15.10 PWM Output Override

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

All control bits associated with the PWM output override function are contained in the OVDCON register. The upper half of the OVDCON register contains eight bits, POVDxH<4:1> and POVDxL<4:1>, that determine which PWM I/O pins will be overridden. The lower half of the OVDCON register contains eight bits, POUTxH<4:1> and POUTxL<4:1>, that determine the state of the PWM I/O pins when a particular output is overridden via the POVD bits.

15.10.1 COMPLEMENTARY OUTPUT MODE

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

15.10.2 OVERRIDE SYNCHRONIZATION

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

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

 Advance Information

15.11 PWM Output and Polarity Control

There are three device configuration bits associated with the PWM module that provide PWM output pin control.

- HPOL configuration bit
- LPOL configuration bit
- PWMPIN configuration bit

These three configuration bits in the FPORBOR configuration register (see the System Integration section) 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 the PWMxH outputs, whereas the LPOL bit specifies the polarity for the PWMxL outputs.

15.11.1 OUTPUT PIN CONTROL

The PEN<4:1>H and PEN<4:1>L control bits in the PWMCON1 SFR enable each High PWM output pin and each Low PWM output pin, respectively. If a particular PWM output pin not enabled, it is treated as a general purpose I/O pin.

15.12 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.

15.12.1 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.

15.12.2 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 PWMxH pin always has priority in the Complementary mode, so that both I/O pins cannot be driven active simultaneously.

15.12.3 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.

15.12.4 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.

15.13 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 is enabled by setting the UDIS control bit in the PWMCON2 SFR. The UDIS bit affects all duty cycle buffer registers and the PWM time-base period buffer, PTPER. No duty cycle changes or period value changes will have effect while UDIS = 1.

15.14 PWM Special Event Trigger

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

The PWM special event trigger has an SFR named SEVTCMP, and five control bits to control its operation. The PTMR value for which a special event trigger should occur is loaded into the SEVTCMP register. When the PWM time-base is in 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.

15.14.1 SPECIAL EVENT TRIGGER POSTSCALER

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

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

- Any write to the SEVTCMP register.
- Any device RESET.

15.15 PWM Operation During CPU SLEEP Mode

The FAULT A 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.

15.16 PWM Operation During CPU IDLE Mode

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

TABLE 15-3: 8-OUTPUT PWM REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
PTCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000 0000 0000 0000
PTMR	01C2	PTDIR	PWM Timer Count Value															0000 0000 0000 0000
PTPER	01C4	—	PWM Time-Base Period Register															0000 0000 0000 0000
SEVTCMP	01C6	SEVTDIR	PWM Special Event Compare Register															0000 0000 0000 0000
PWMCON1	01C8	—	—	—	—	PTMOD4	PTMOD3	PTMOD2	PTMOD1	PEN4H	PEN3H	PEN2H	PEN1H	PEN4L	PEN3L	PEN2L	PEN1L	0000 0000 1111 1111
PWMCON2	01CA	—	—	—	—	SEVOPS<3:0>				—	—	—	—	—	—	OSYNC	UDIS	0000 0000 0000 0000
DTCON1	01CC	DTBPS<1:0>		Dead-Time B Value						DTAPS<1:0>		Dead-Time A Value						0000 0000 0000 0000
DTCON2	01CE	—	—	—	—	—	—	—	—	DTS4A	DTS4I	DTS3A	DTS3I	DTS2A	DTS2I	DTS1A	DTS1I	0000 0000 0000 0000
FLTACON	01D0	FAOV4H	FAOV4L	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	FAEN4	FAEN3	FAEN2	FAEN1	0000 0000 0000 0000
FLTBCON	01D2	FBOV4H	FBOV4L	FBOV3H	FBOV3L	FBOV2H	FBOV2L	FBOV1H	FBOV1L	FLTBM	—	—	—	FBEN4	FBEN3	FBEN2	FBEN1	0000 0000 0000 0000
OVDCON	01D4	POVD4H	POVD4L	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	POUT4H	POUT4L	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	1111 1111 0000 0000
PDC1	01D6	PWM Duty Cycle #1 Register																0000 0000 0000 0000
PDC2	01D8	PWM Duty Cycle #2 Register																0000 0000 0000 0000
PDC3	01DA	PWM Duty Cycle #3 Register																0000 0000 0000 0000
PDC4	01DC	PWM Duty Cycle #4 Register																0000 0000 0000 0000

Legend: u = uninitialized bit

TABLE 15-4: 6-OUTPUT PWM REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
PTCON	01C0	PTEN	—	PTSIDL	—	—	—	—	—	PTOPS<3:0>				PTCKPS<1:0>		PTMOD<1:0>		0000 0000 0000 0000
PTMR	01C2	PTDIR	PWM Timer Count Value															0000 0000 0000 0000
PTPER	01C4	—	PWM Time-Base Period Register															0000 0000 0000 0000
SEVTCMP	01C6	SEVTDIR	PWM Special Event Compare Register															0000 0000 0000 0000
PWMCON1	01C8	—	—	—	—	—	PTMOD3	PTMOD2	PTMOD1	—	PEN3H	PEN2H	PEN1H	—	PEN3L	PEN2L	PEN1L	0000 0000 0111 0111
PWMCON2	01CA	—	—	—	—	SEVOPS<3:0>				—	—	—	—	—	—	OSYNC	UDIS	0000 0000 0000 0000
DTCON1	01CC	—	—	—	—	—	—	—	—	DTAPS<1:0>		Dead-Time A Value						0000 0000 0000 0000
FLTACON	01D0	—	—	FAOV3H	FAOV3L	FAOV2H	FAOV2L	FAOV1H	FAOV1L	FLTAM	—	—	—	—	FAEN3	FAEN2	FAEN1	0000 0000 0000 0000
OVDCON	01D4	—	—	POVD3H	POVD3L	POVD2H	POVD2L	POVD1H	POVD1L	—	—	POUT3H	POUT3L	POUT2H	POUT2L	POUT1H	POUT1L	0011 1111 0000 0000
PDC1	01D6	PWM Duty Cycle #1 Register																0000 0000 0000 0000
PDC2	01D8	PWM Duty Cycle #2 Register																0000 0000 0000 0000
PDC3	01DA	PWM Duty Cycle #3 Register																0000 0000 0000 0000

Legend: u = uninitialized bit

NOTES:

16.0 SPI MODULE

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

16.1 Operating Function Description

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.

Note:	The transmit and receive buffers are mapped to the same register address, SPIxBUF.

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.

16.1.1 WORD AND BYTE COMMUNICATION

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

The user software must disable the module prior to changing the MODE16 bit. The SPI module 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.

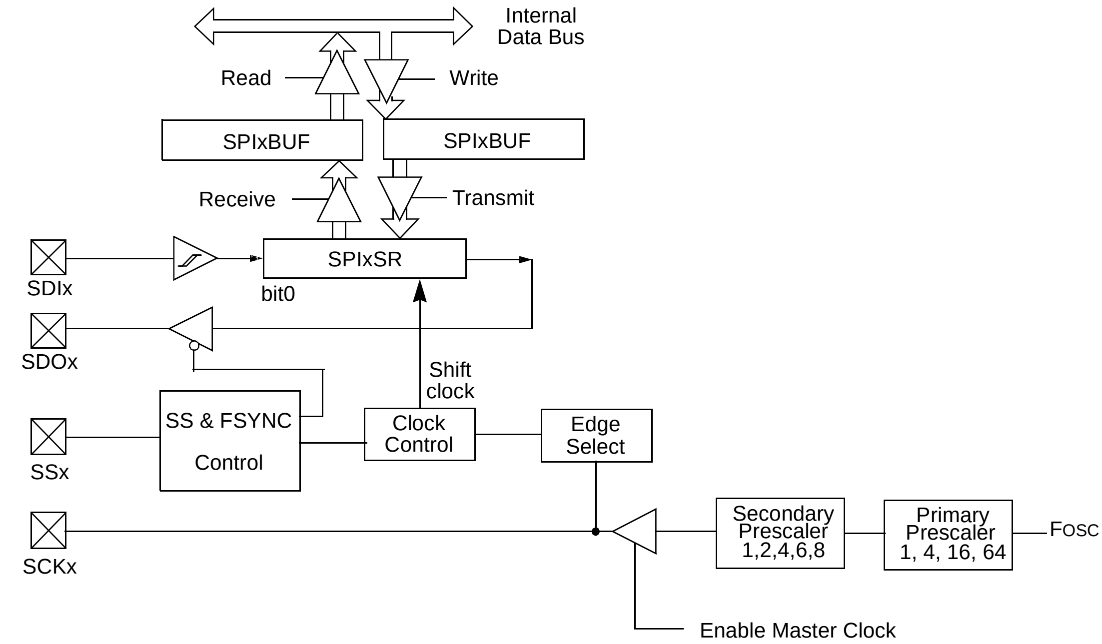
16.1.2 SDOx DISABLE

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.

16.2 Framed SPI Support

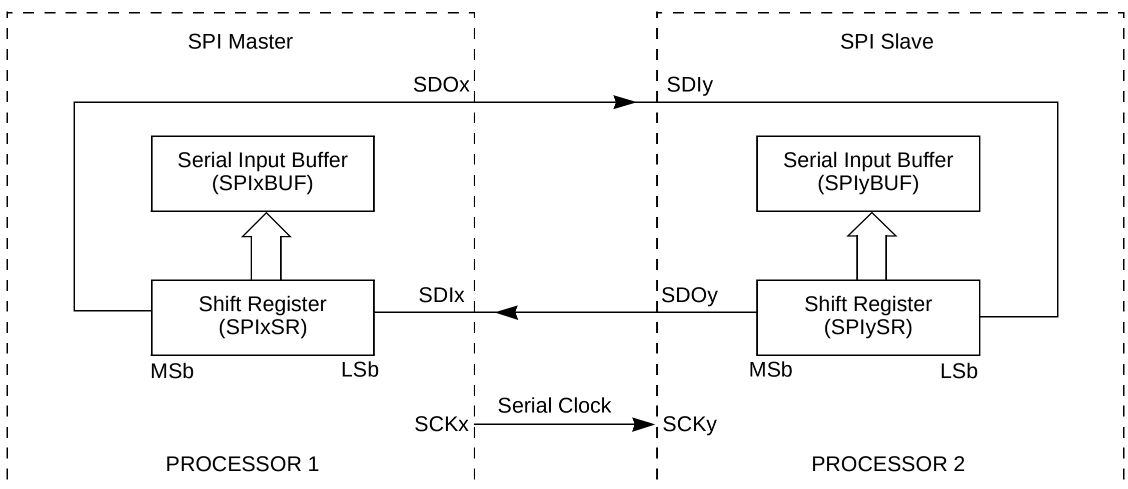
The module supports a basic framed SPI protocol in Master or Slave mode. The control bit FRMEN enables framed SPI support and causes the SSx pin to perform the frame synchronization pulse (FSYNC) function. The control bit SPIFSD determines whether the SSx pin is an input or an output, i.e., whether the 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.

FIGURE 16-1: SPI BLOCK DIAGRAM



Note: x = 1 or 2.

FIGURE 16-2: SPI MASTER/SLAVE CONNECTION



Note: x = 1 or 2, y = 1 or 2.

 Advance Information

16.3 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.

16.4 SPI Operation During CPU SLEEP Mode

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

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

16.5 SPI Operation During CPU IDLE Mode

When the device enters IDLE mode, all clock sources remain functional. The SPISIDL bit (SPIxSTAT<13>) 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 16-1: 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	—	—	—	—	SPITBF	SPIRBF	0000 0000 0000 0000
SPI1CON	0222	—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	0000 0000 0000 0000
SPI1BUF	0224	Transmit and Receive Buffer																0000 0000 0000 0000

Legend: u = uninitialized bit

TABLE 16-2: 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	—	—	—	—	SPITBF	SPIRBF	0000 0000 0000 0000
SPI2CON	0228	—	FRMEN	SPIFSD	—	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	0000 0000 0000 0000
SPI2BUF	022A	Transmit and Receive Buffer																0000 0000 0000 0000

Legend: u = uninitialized bit

17.0 I^{2}C MODULE

The Inter-Integrated Circuit™ (I^{2}C™) 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 operation.
- 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 operation; detects bus collision and will arbitrate accordingly.

17.1 Operating Function Description

The hardware fully implements all the master and slave functions of the 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.

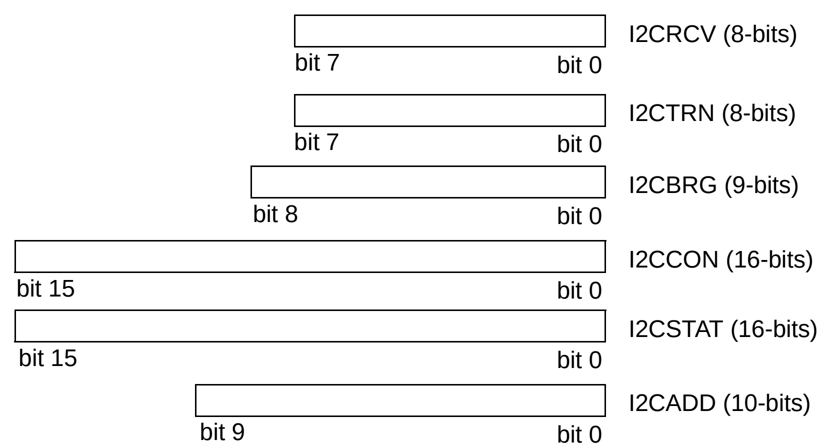
17.1.1 VARIOUS I^{2}C MODES

The following types of I^{2}C operation are supported:

- I^{2}C Slave operation with 7-bit address
- I^{2}C Slave operation with 10-bit address
- I^{2}C Master operation with 7 or 10-bit address

See the I^{2}C programmer's model in Figure 17-1.

FIGURE 17-1: PROGRAMMER'S MODEL



17.1.2 PIN CONFIGURATION IN I^{2}C MODE

I^{2}C has a 2-pin interface: pin SCL is clock and pin SDA is data.

17.1.3 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.

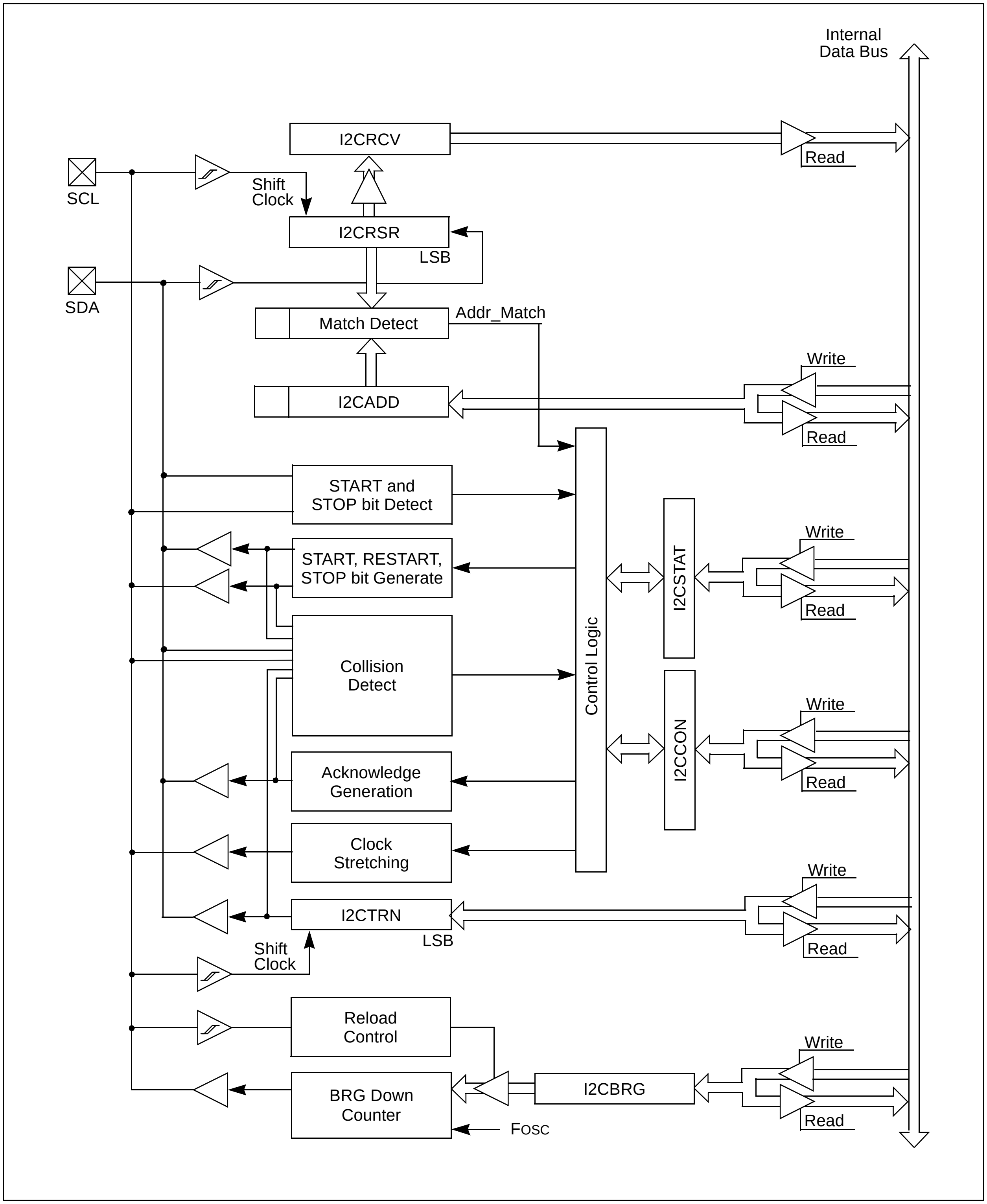
I2CRSR is the shift register used for shifting data, whereas I2CRCV is the buffer register to which data bytes are written to or read from. This register is the receive buffer, as shown in Figure 16-1. I2CTRN is the transmit register to which bytes are written during a transmit operation, as shown in 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.

FIGURE 17-2: I²C BLOCK DIAGRAM



 Advance Information

17.2 I^2C Module Addresses

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

If the A10M bit (I2CCON<10>) is 0, the address is 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.

17.3 I^2C 7-bit Slave Mode Operation

Once enabled (I2CEN = 1), the slave module will wait for a START bit to occur (i.e., the I^2C 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 ($\overline{ACK}$ bit), the I2CIF interrupt pulse is generated. The address match does not affect the contents of the I2CRCV buffer or the RBF bit.

17.3.1 SLAVE TRANSMISSION

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

17.3.2 SLAVE RECEPTION

If the R_W bit received is a '0' during an address match, then Receive mode is initiated. Incoming bits are sampled on the rising edge of SCL. After 8 bits are received, if I2CRCV is not full or I2COV is not set, I2CRSR is transferred to I2CRCV. $\overline{ACK}$ 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 $\overline{ACK}$ 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 ($\overline{ACK}$ = 1) and the I2CRCV is updated.

17.4 I^2C 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^2C 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.

17.4.1 10-BIT MODE SLAVE TRANSMISSION

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

17.4.2 10-BIT MODE SLAVE RECEPTION

Once addressed, the master can, 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.

17.5 Automatic Clock Stretch

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

17.5.1 TRANSMIT CLOCK STRETCHING

Both 10-bit and 7-bit Transmit modes implement clock stretching by asserting the SCLREL bit after the falling edge of the ninth clock if the TBF bit is cleared, indicating the buffer is empty. 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 $\overline{ACK}$ on the falling edge of the ninth clock, and if the TBF bit is still clear, then the SCLREL bit is automatically cleared. The SCLREL being cleared to '0' will assert the SCL line low. The user's ISR must set the SCLREL bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the I2CTRN before the master device can initiate another transmit sequence.

> **Note 1:** If the user loads the contents of I2CTRN, setting the TBF bit before the falling edge of the ninth clock, the SCLREL bit will not be cleared and clock stretching will not occur.
>
> **2:** The SCLREL bit can be set in software, regardless of the state of the TBF bit.

17.5.2 RECEIVE CLOCK STRETCHING

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

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

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

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

> **Note 1:** If the user reads the contents of the I2CRCV, clearing the RBF bit before the falling edge of the ninth clock, the SCLREL bit will not be cleared and clock stretching will not occur.
>
> **2:** The SCLREL bit can be set in software, regardless of the state of the RBF bit. The user should be careful to clear the RBF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

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

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

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

17.6 Software Controlled Clock Stretching (STREN = 1)

When the STREN bit is '1', the SCLREL bit may be cleared by software to allow software to control the clock stretching. The logic will synchronize writes to the SCLREL bit with the SCL clock. Clearing the SCLREL bit will not assert the SCL output until the module detects a falling edge on the SCL output and SCL is sampled low. If the SCLREL bit is cleared by the user while the SCL line 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^2C 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.

17.7 Interrupts

The I^2C module generates two interrupt flags, I2CIF (I^2C Transfer Complete Interrupt Flag) and BCLIF (I^2C Bus Collision Interrupt Flag). The I2CIF interrupt flag is pulsed high for one T_{CY} on the falling edge of the 9th clock pulse. The BCLIF interrupt flag is pulsed high for one T_{CY} when a bus collision event is detected.

17.8 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.

17.9 IPMI Support

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

17.10 General Call Address Support

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

17.11 I^{2}C Master Support

As a Master device, six operations are supported.

- Assert a START condition on SDA and SCL.
- Assert a RESTART condition on SDA and SCL.
- Write to the I2CTRN register initiating transmission of data/address.
- Generate a STOP condition on SDA and SCL.
- Configure the I^{2}C port to receive data.
- Generate an Acknowledge condition at the end of a received byte of data.

17.12 I^{2}C Master 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.

17.12.1 I^{2}C MASTER TRANSMISSION

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

17.12.2 I^{2}C MASTER 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.

17.12.3 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. If clock arbitration is taking place, for instance, the BRG is reloaded when the SCL pin is sampled high.

As per the I^{2}C standard, FSCK may be 100 kHz, 400 kHz. However, the user can specify any baud rate up to 1 MHz. I2CBRG values of 0 or 1 are illegal.

EQUATION 17-1: SERIAL CLOCK RATE

$$FSCK = F_{CY} / I2CBRG$$

17.12.4 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 ensures that the SCL high time will always be at least one BRG rollover count, in the event that the clock is held low by an external device.

17.12.5 MULTI-MASTER COMMUNICATION, BUS COLLISION, AND BUS ARBITRATION

Multi-Master operation support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a 1 on SDA, by letting SDA float high 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^2C 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^2C 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^2C 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 a Multi-Master environment, the interrupt generation on the detection of START and STOP conditions allows the determination of when the bus is free. Control of the I^2C bus can be taken when the P bit is set in the I2CSTAT register, or the bus is IDLE and the S and P bits are cleared.

17.13 I^2C Module Operation During CPU SLEEP and IDLE Modes

17.13.1 I^2C OPERATION DURING CPU SLEEP MODE

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

17.13.2 I^2C OPERATION DURING CPU IDLE MODE

For the I^2C, the I2CSIDL bit selects if the module will stop on IDLE or continue on IDLE. If I2CSIDL = 0, the module will continue operation on assertion of the IDLE mode. If I2CSIDL = 1, the module will stop on IDLE.

TABLE 17-1: 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								0000 0000 0000 0000
I2CTRN	0202	—	—	—	—	—	—	—	—	Transmit Register								0000 0000 1111 1111
I2CBRG	0204	—	—	—	—	—	—	—	Baud Rate Generator									0000 0000 0000 0000
I2CCON	0206	I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0001 0000 0000 0000
I2CSTAT	0208	ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10	IWCOL	I2COV	D_A	P	S	R_W	RBF	TBF	0000 0000 0000 0000
I2CADD	020A	—	—	—	—	—	—	Address Register										0000 0000 0000 0000

Legend: u = uninitialized bit

NOTES:

18.0 UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER (UART) MODULE

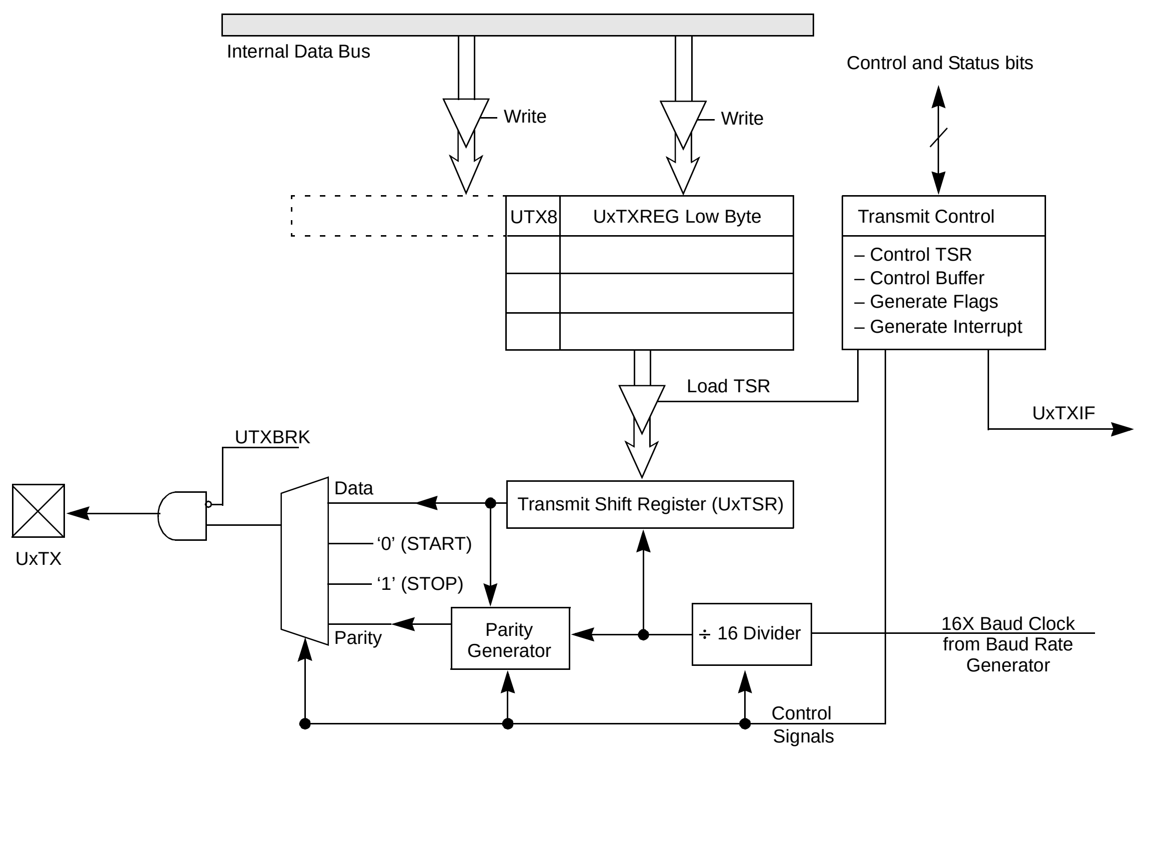
This section describes the Universal Asynchronous Receiver/Transmitter Communications module.

18.1 UART Module Overview

The key features of the UART module are:

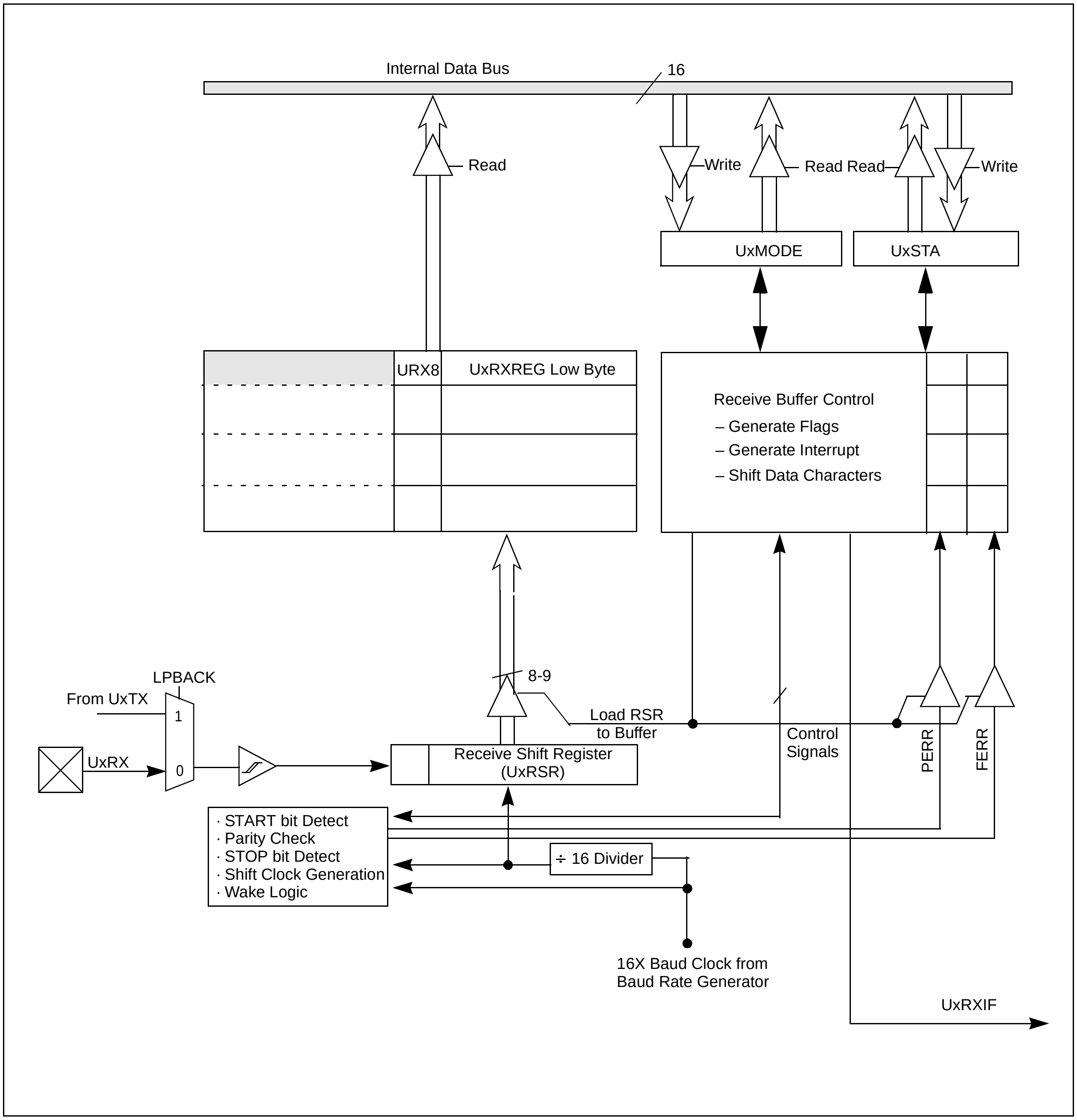
- Full-duplex, 8 or 9-bit data communication
- Even, Odd or No Parity options (for 8-bit data)
- One or two STOP bits
- Fully integrated Baud Rate Generator with 16-bit prescaler
- Baud rates range from 38 bps to 1.875 Mbps at a 30 MHz instruction rate
- 4-word deep transmit data buffer
- 4-word deep receive data buffer
- Parity, Framing and Buffer Overrun error detection
- Support for Interrupt only on Address Detect (9th bit = 1)
- Separate Transmit and Receive Interrupts
- Loopback mode for diagnostic support

FIGURE 18-1: UART TRANSMITTER BLOCK DIAGRAM



Note: x = 1 or 2.

FIGURE 18-2: UART RECEIVER BLOCK DIAGRAM



18.2 Enabling and Setting Up UART

18.2.1 ENABLING THE UART

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

18.2.2 DISABLING THE UART

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

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

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

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

18.2.3 ALTERNATE I/O

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

18.2.4 SETTING UP DATA, PARITY AND STOP BIT SELECTIONS

Control bits PDSEL<1:0> in the 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).

18.3 Transmitting Data

18.3.1 TRANSMITTING IN 8-BIT DATA MODE

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

1. Set up the UART:
   First, the data length, parity and number of STOP bits must be selected, 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.

18.3.2 TRANSMITTING IN 9-BIT DATA MODE

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

18.3.3 TRANSMIT BUFFER (UxTXB)

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

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

The FIFO is reset during any device RESET, but is not affected when the device enters a Power Saving mode, or wakes up from a Power Saving mode.

18.3.4 TRANSMIT INTERRUPT

The transmit interrupt flag (U1TXIF or U2TXIF) is located in the corresponding interrupt flag register.

The transmitter generates an edge to set the UxTXIF bit. The condition 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.

18.3.5 TRANSMIT BREAK

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

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

18.4 Receiving Data

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

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

1. Set up the UART (see Section 18.3.1).
2. Enable the UART (see Section 18.3.1).
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.

18.4.2 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.

18.4.3 RECEIVE INTERRUPT

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

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

18.5 Reception Error Handling

18.5.1 RECEIVE BUFFER OVERRUN ERROR (OERR BIT)

The OERR bit (UxSTA<1>) is set if all of the following conditions occur:

a) The receive buffer is full.
b) The receive shift register is full, but unable to transfer the character to the receive buffer.
c) The STOP bit of the character in the UxRSR is detected, indicating that the UxRSR needs to transfer the character to the buffer.

Once OERR is set, no further data is shifted in UxRSR (until the OERR bit is cleared in software or a RESET occurs). The data held in UxRSR and UxRXREG remains valid.

18.5.2 FRAMING ERROR (FERR)

The FERR bit (UxSTA<2>) is set if a ‘0’ is detected instead of a STOP bit. If two STOP bits are selected, both STOP bits must be ‘1’, otherwise FERR will be set. The read only FERR bit is buffered along with the received data. It is cleared on any RESET.

18.5.3 PARITY ERROR (PERR)

The PERR bit (UxSTA<3>) is set if the parity of the received word is incorrect. This error bit is applicable only if a Parity mode (odd or even) is selected. The read only PERR bit is buffered along with the received data bytes. It is cleared on any RESET.

18.5.4 IDLE STATUS

When the receiver is active, i.e., between the initial detection of the START bit and the completion of the STOP bit, the RIDLE bit (UxSTA<4>) is ‘0’. Between the completion of the STOP bit and detection of the next START bit, the RIDLE bit is ‘1’, indicating that the UART is IDLE.

18.5.5 RECEIVE BREAK

The receiver will count and expect a certain number of bit times based on the values programmed in the PDSEL (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.

18.6 Address Detect Mode

Setting the ADDEN bit (UxSTA<5>) enables this special mode, in which a 9th bit (URX8) value of ‘1’ identifies the received word as an address rather than data. This mode is only applicable for 9-bit data communication. The URXISEL control bit does not have any impact on interrupt generation in this mode, since an interrupt (if enabled) will be generated every time the received word has the 9th bit set.

18.7 Loopback Mode

Setting the LPBACK bit enables this special mode, in which the UxTX pin is internally connected to the UxRX pin. When configured for the loopback mode, the UxRX pin is disconnected from the internal UART receive logic. However, the UxTX pin still functions as in a normal operation.

To select this mode:

a) Configure UART for desired mode of operation.
a) Set LPBACK = 1 to enable Loopback mode.
b) Enable transmission as defined in Section 18.3.

18.8 Baud Rate Generator

The UART has a 16-bit baud rate generator to allow maximum flexibility in baud rate generation. The baud rate generator register (UxBRG) is readable and writable. The baud rate is computed as follows:

BRG = 16-bit value held in UxBRG register (0 through 65535)

F_{CY} = Instruction Clock Rate ($1/T_{CY}$)

The equation for the Baud Rate is given below.

EQUATION 18-1: BAUD RATE

$$\text{Baud Rate} = F_{CY} / (16*(BRG+1))$$

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.

18.9 Auto Baud Support

To allow the system to determine baud rates of received characters, the input can be optionally linked to a selected capture input. To enable this mode, the user must program the input capture module to detect the falling and rising edges of the START bit.

18.10 UART Operation During CPU SLEEP and IDLE Modes

18.10.1 UART OPERATION DURING CPU SLEEP MODE

When the device enters SLEEP mode, all clock sources to the module are shutdown 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.

18.10.2 UART OPERATION DURING CPU IDLE MODE

For the UART, the USIDL bit selects if the module will stop operation when the device enters IDLE mode, or whether the module will continue on IDLE. If USIDL = 0, the module will continue operation during IDLE mode. If USIDL = 1, the module will stop on IDLE.

TABLE 18-1: UART1 REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
U1MODE	020C	UARTEN	—	USIDL	—	—	ALTIO	—	—	WAKE	LPBACK	ABAUD	—	—	PDSEL1	PDSEL0	STSEL	0000 0000 0000 0000
U1STA	020E	UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0000 0001 0001 0000
U1TXREG	0210	—	—	—	—	—	—	—	UTX8	Transmit Register								0000 000u uuuu uuuu
U1RXREG	0212	—	—	—	—	—	—	—	URX8	Receive Register								0000 0000 0000 0000
U1BRG	0214	Baud Rate Generator Prescaler																0000 0000 0000 0000

Legend: u = uninitialized bit

TABLE 18-2: 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	—	—	—	—	—	WAKE	LPBACK	ABAUD	—	—	PDSEL1	PDSEL0	STSEL	0000 0000 0000 0000
U2STA	0218	UTXISEL	—	—	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0000 0001 0001 0000
U2TXREG	021A	—	—	—	—	—	—	—	UTX8	Transmit Register								0000 000u uuuu uuuu
U2RXREG	021C	—	—	—	—	—	—	—	URX8	Receive Register								0000 0000 0000 0000
U2BRG	021E	Baud Rate Generator Prescaler																0000 0000 0000 0000

Legend: u = uninitialized bit

NOTES:

19.0 CAN MODULE

19.1 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 (each buffer may contain up to 8 bytes of data)
- 6 full (standard/extended identifier) acceptance filters, 2 associated with the high priority receive buffer, and 4 associated with the low priority receive buffer
- 2 full acceptance filter masks, one each associated with the high and low priority receive buffers
- Three transmit buffers with application specified prioritization and abort capability (each buffer may contain up to 8 bytes of data)
- Programmable wake-up functionality with integrated low pass filter
- Programmable Loopback mode supports self-test operation
- Signaling via interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to timer module for time-stamping and network synchronization
- Low power SLEEP mode

19.1.1 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.

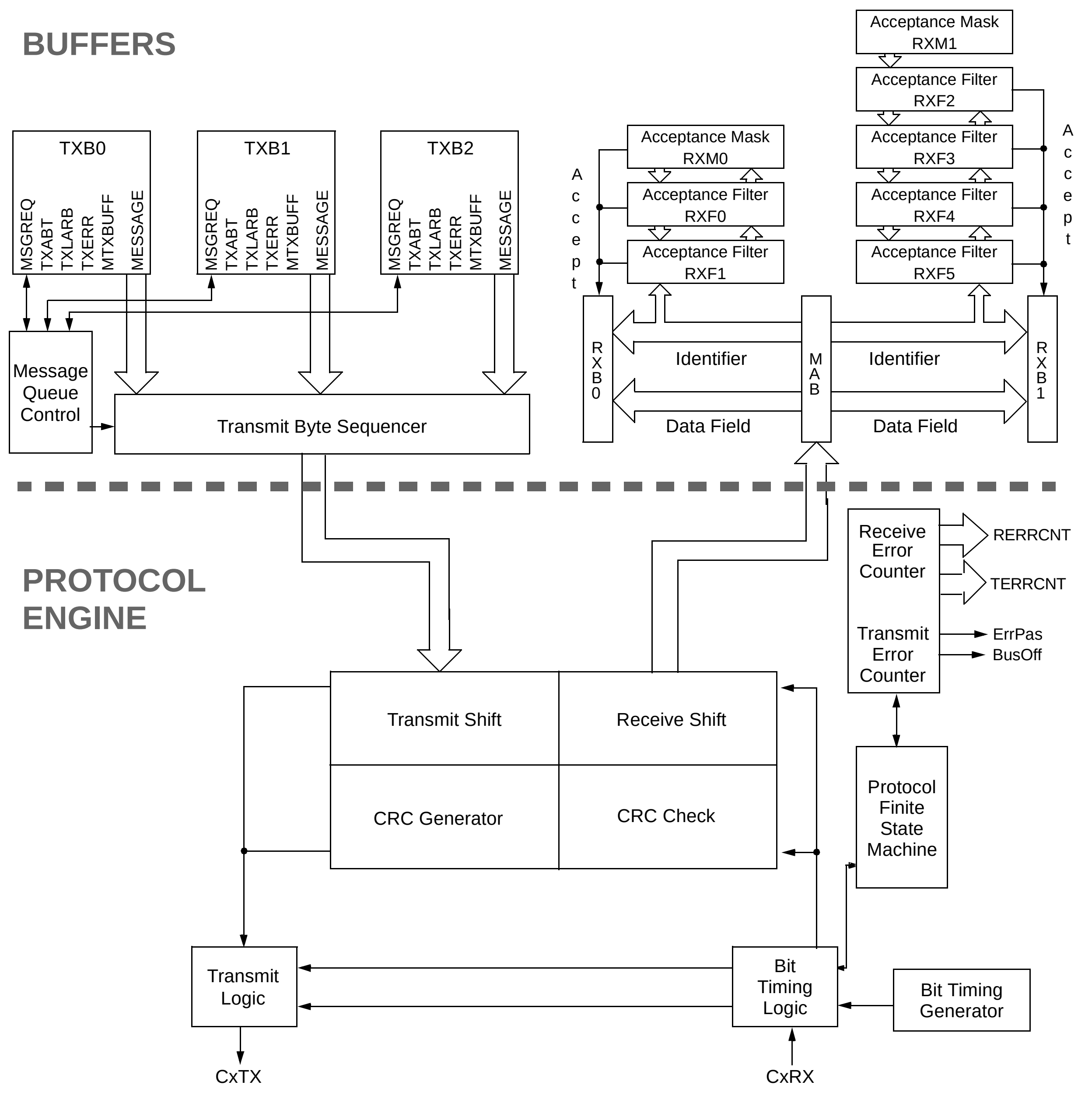
The CAN Module supports the following Frame types:

- Standard Data Frame
- Extended Data Frame
- Remote Frame
- Error Frame
- Overload Frame Reception
- Interframe Space

19.1.2 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 19-1 is a block diagram of these buffers and their connection to the protocol engine.

FIGURE 19-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM



Note: x = 1 or 2.

TABLE 19-1: CAN1 REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
C1RXF0SID	0300	Receive Acceptance Filter 0 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C1RXF0EID	0302	Receive Acceptance Filter 0 Extended Identifier																uuuu uuuu uuuu uuuu
C1RXF1SID	0304	Receive Acceptance Filter 1 Standard Identifier Filter											—	EXIDEN	—	EDI17	EID16	uuuu uuuu uuu0 u0uu
C1RXF1EID	0306	Receive Acceptance Filter 1 Extended Identifier																uuuu uuuu uuuu uuuu
C1RXF2SID	0308	Receive Acceptance Filter 2 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C1RXF2EID	030A	Receive Acceptance Filter 2 Extended Identifier																uuuu uuuu uuuu uuuu
C1RXF3SID	030C	Receive Acceptance Filter 3 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C1RXF3EID	030E	Receive Acceptance Filter 3 Extended Identifier																uuuu uuuu uuuu uuuu
C1RXF4SID	0310	Receive Acceptance Filter 4 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C1RXF4EID	0312	Receive Acceptance Filter 4 Extended Identifier																uuuu uuuu uuuu uuuu
C1RXF5SID	0314	Receive Acceptance Filter 5 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C1RXF5EID	0316	Receive Acceptance Filter 5 Extended Identifier																uuuu uuuu uuuu uuuu
C1RXM0SID	0318	Receive Acceptance Mask 0 Standard Identifier Mask											—	—	—	EID17	EID16	uuuu uuuu uuu0 00uu
C1RXM0EID	031A	Receive Acceptance Mask 0 Extended Identifier Mask																uuuu uuuu uuuu uuuu
C1RXM1SID	031C	Receive Acceptance Mask 1 Standard Identifier Mask											—	—	—	EID17	EID16	uuuu uuuu uuu0 00uu
C1RXM1EID	031E	Receive Acceptance Mask 1 Extended Identifier Mask																uuuu uuuu uuuu uuuu
C1TX2CON	0320	—	TXRTR	—	—	DLC<3:0>				—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>		0000 0000 0000 0000
C1TX2SID	0322	Transmit Buffer 2 Standard Identifier											—	EXIDE	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C1TX2EID	0324	Transmit Buffer 2 Extended Identifier																uuuu uuuu uuuu uuuu
C1TX2B1	0326	Transmit Buffer 2 Byte 1								Transmit Buffer 2 Byte 0								uuuu uuuu uuuu uuuu
C1TX2B2	0328	Transmit Buffer 2 Byte 3								Transmit Buffer 2 Byte 2								uuuu uuuu uuuu uuuu
C1TX2B3	032A	Transmit Buffer 2 Byte 5								Transmit Buffer 2 Byte 4								uuuu uuuu uuuu uuuu
C1TX2B4	032C	Transmit Buffer 2 Byte 7								Transmit Buffer 2 Byte 6								uuuu uuuu uuuu uuuu
C1TX1CON	0330	—	TXRTR	—	—	DLC<3:0>				—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>		0000 0000 0000 0000
C1TX1SID	0332	Transmit Buffer12 Standard Identifier											—	EXIDE	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C1TX1EID	0334	Transmit Buffer 1 Extended Identifier																uuuu uuuu uuuu uuuu
C1TX1B1	0336	Transmit Buffer 1 Byte 1								Transmit Buffer 1 Byte 0								uuuu uuuu uuuu uuuu
C1TX1B2	0338	Transmit Buffer 1 Byte 3								Transmit Buffer 1 Byte 2								uuuu uuuu uuuu uuuu
C1TX1B3	033A	Transmit Buffer 1 Byte 5								Transmit Buffer 1 Byte 4								uuuu uuuu uuuu uuuu
C1TX1B4	033C	Transmit Buffer 1 Byte 7								Transmit Buffer 1 Byte 6								uuuu uuuu uuuu uuuu
C1TX0CON	0340	—	TXRTR	—	—	DLC<3:0>				—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>		0000 0000 0000 0000
C1TX0SID	0342	Transmit Buffer STARTD Identifer											—	EXIDE	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C1TX0EID	0344	Transmit Buffer Extended Identifier																uuuu uuuu uuuu uuuu
C1TX0B1	0346	Transmit Buffer 0 Byte 1								Transmit Buffer 0 Byte 0								uuuu uuuu uuuu uuuu
C1TX0B2	0348	Transmit Buffer 0 Byte 3								Transmit Buffer 0 Byte 2								uuuu uuuu uuuu uuuu
C1TX0B3	034A	Transmit Buffer 0 Byte 5								Transmit Buffer 0 Byte 4								uuuu uuuu uuuu uuuu

Legend: u = uninitialized bit

TABLE 19-1: CAN1 REGISTER MAP (CONTINUED)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
C1TX0B4	034C	Transmit Buffer 0 Byte 7								Transmit Buffer 0 Byte 6								uuuu uuuu uuuu uuuu
C1RX1CON	0350	—	RXRTR	RB1	RB0	DLC<3:0>				RXFUL	RXM<1:0>		—	RXRTRO	FILHIT<2:0>			0000 0000 0000 0000
C1RX1SID	0352	Receive Buffer 1 Standard Identifier											SRR	EXID	—	EID17	EDI16	uuuu uuuu uuuu u0uu
C1RX1EID	0354	Receive Buffer 1 Extended Identifer																uuuu uuuu uuuu uuuu
C1RX1B1	0356	Receive Buffer 1 Byte 1								Receive Buffer 1 Byte 0								uuuu uuuu uuuu uuuu
C1RX1B2	0358	Receive Buffer 1 Byte 3								Receive Buffer 1 Byte 2								uuuu uuuu uuuu uuuu
C1RX1B3	035A	Receive Buffer 1 Byte 5								Receive Buffer 1 Byte 4								uuuu uuuu uuuu uuuu
C1RX1B4	035C	Receive Buffer 1 Byte 7								Receive Buffer 1 Byte 6								uuuu uuuu uuuu uuuu
C1RX0CON	0360	—	RXRTR	RB1	RB0	DLC<3:0>				RXFUL	RXM<1:0>		—	RXRTRO	DBEN	JTOFF	FILHIT0	0000 0000 0000 0000
C1RX0SID	0362	Receive Buffer 0 Standard Identifier											SRR	EXID	—	EID17	EID16	uuuu uuuu uuuu u0uu
C1RX0EID	0364	Receive Buffer 0 Extended Identifer																uuuu uuuu uuuu uuuu
C1RX0B1	0366	Receive Buffer 0 Byte 1								Receive Buffer 0 Byte 0								uuuu uuuu uuuu uuuu
C1RX0B2	0368	Receive Buffer 0 Byte 3								Receive Buffer 0 Byte 2								uuuu uuuu uuuu uuuu
C1RX0B3	036A	Receive Buffer 0 Byte 5								Receive Buffer 0 Byte 4								uuuu uuuu uuuu uuuu
C1RX0B4	036C	Receive Buffer 0 Byte 7								Receive Buffer 0 Byte 6								uuuu uuuu uuuu uuuu
C1CTRL	036E	—	—	CSIDLE	ABAT	—	REQOP<2:0>			OPMODE<2:0>			—	ICODE<2:0>			—	0000 0100 1000 0000
C1CFG1	0370	—	—	—	—	—	—	—	—	SJWS<1:0>		BRP<5:0>						0000 0000 0000 0000
C1CFG2	0372	CANCAP	WAKFIL	—	—	—	SEG2PH<2:0>			BTLMODE	SAM	SEG1PH<2:0>			PRSEG<2:0>			uu00 0uuu uuuu uuuu
C1INTF	0374	RX0OVR	RX1OVR	TXBO	TXEP	RXEP	TXWAR	RXWAR	EWARN	IVRIF	WAKIF	ERRIF	TX2IF	TX1IF	TX0IF	RX1IF	RX0IF	0000 0000 0000 0000
C1INTE	0376	—	—	—	—	—	—	—	—	IVRIE	WAKIE	ERRIE	TX2IE	TX1IE	TX0IE	RX1E	RX0IE	0000 0000 0000 0000
C1EC	0378	Transmit Error Count Register								Receive Error Count Register								0000 0000 0000 0000

Legend: u = uninitialized bit

TABLE 19-2: CAN2 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
C2RXF0SID	0380	Receive Acceptance Filter 0 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C2RXF0EID	0382	Receive Acceptance Filter 0 Extended Identifier																uuuu uuuu uuuu uuuu
C2RXF1SID	0384	Receive Acceptance Filter 1 Standard Identifier Filter											—	EXIDEN	—	EDI17	EID16	uuuu uuuu uuu0 u0uu
C2RXF1EID	0386	Receive Acceptance Filter 1 Extended Identifier																uuuu uuuu uuuu uuuu
C2RXF2SID	0388	Receive Acceptance Filter 2 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C2RXF2EID	038A	Receive Acceptance Filter 2 Extended Identifier																uuuu uuuu uuuu uuuu
C2RXF3SID	038C	Receive Acceptance Filter 3 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C2RXF3EID	038E	Receive Acceptance Filter 3 Extended Identifier																uuuu uuuu uuuu uuuu
C2RXF4SID	0390	Receive Acceptance Filter 4 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C2RXF4EID	0392	Receive Acceptance Filter 4 Extended Identifier																uuuu uuuu uuuu uuuu
C2RXF5SID	0394	Receive Acceptance Filter 5 Standard Identifier Filter											—	EXIDEN	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C2RXF5EID	0396	Receive Acceptance Filter 5 Extended Identifier																uuuu uuuu uuuu uuuu
C2RXM0SID	0398	Receive Acceptance Mask 0 Standard Identifier Mask											—	—	—	EID17	EID16	uuuu uuuu uuu0 00uu
C2RXM0EID	039A	Receive Acceptance Mask 0 Extended Identifier Mask																uuuu uuuu uuuu uuuu
C2RXM1SID	039C	Receive Acceptance Mask 1 Standard Identifier Mask											—	—	—	EID17	EID16	uuuu uuuu uuu0 00uu
C2RXM1EID	039E	Receive Acceptance Mask 1 Extended Identifier Mask																uuuu uuuu uuuu uuuu
C2TX2CON	03A0	—	TXRTR	—	—	DLC<3:0>				—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>		0000 0000 0000 0000
C2TX2SID	03A2	Transmit Buffer 2 Standard Identifier											—	EXIDE	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C2TX2EID	03A4	Transmit Buffer 2 Extended Identifier																uuuu uuuu uuuu uuuu
C2TX2B1	03A6	Transmit Buffer 2 Byte 1								Transmit Buffer 2 Byte 0								uuuu uuuu uuuu uuuu
C2TX2B2	03A8	Transmit Buffer 2 Byte 3								Transmit Buffer 2 Byte 2								uuuu uuuu uuuu uuuu
C2TX2B3	03AA	Transmit Buffer 2 Byte 5								Transmit Buffer 2 Byte 4								uuuu uuuu uuuu uuuu
C2TX2B4	03AC	Transmit Buffer 2 Byte 7								Transmit Buffer 2 Byte 6								uuuu uuuu uuuu uuuu
C2TX1CON	03B0	—	TXRTR	—	—	DLC<3:0>				—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>		0000 0000 0000 0000
C2TX1SID	03B2	Transmit Buffer12 Standard Identifier											—	EXIDE	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C2TX1EID	03B4	Transmit Buffer 1 Extended Identifier																uuuu uuuu uuuu uuuu
C2TX1B1	03B6	Transmit Buffer 1 Byte 1								Transmit Buffer 1 Byte 0								uuuu uuuu uuuu uuuu
C2TX1B2	03B8	Transmit Buffer 1 Byte 3								Transmit Buffer 1 Byte 2								uuuu uuuu uuuu uuuu
C2TX1B3	03BA	Transmit Buffer 1 Byte 5								Transmit Buffer 1 Byte 4								uuuu uuuu uuuu uuuu
C2TX1B4	03BC	Transmit Buffer 1 Byte 7								Transmit Buffer 1 Byte 6								uuuu uuuu uuuu uuuu
C2TX0CON	03C0	—	TXRTR	—	—	DLC<3:0>				—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI<1:0>		0000 0000 0000 0000
C2TX0SID	03C2	Transmit Buffer STARTD Identifer											—	EXIDE	—	EID17	EID16	uuuu uuuu uuu0 u0uu
C2TX0EID	03C4	Transmit Buffer Extended Identifier																uuuu uuuu uuuu uuuu
C2TX0B1	03C6	Transmit Buffer 0 Byte 1								Transmit Buffer 0 Byte 0								uuuu uuuu uuuu uuuu
C2TX0B2	03C8	Transmit Buffer 0 Byte 3								Transmit Buffer 0 Byte 2								uuuu uuuu uuuu uuuu

Legend: u = uninitialized bit

TABLE 19-2: CAN2 REGISTER MAP (CONTINUED)

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
C2TX0B3	03CA	Transmit Buffer 0 Byte 5								Transmit Buffer 0 Byte 4								uuuu uuuu uuuu uuuu
C2TX0B4	03CC	Transmit Buffer 0 Byte 7								Transmit Buffer 0 Byte 6								uuuu uuuu uuuu uuuu
C2RX1CON	03D0	—	RXRTR	RB1	RB0	DLC<3:0>				RXFUL	RXM<1:0>		-—	RXRTRO	FILHIT<2:0>			0000 0000 0000 0000
C2RX1SID	03D2	Receive Buffer 1 Standard Identifier											SRR	EXID	—	EID17	EDI16	uuuu uuuu uuuu u0uu
C2RX1EID	03D4	Receive Buffer 1 Extended Identifer																uuuu uuuu uuuu uuuu
C2RX1B1	03D6	Receive Buffer 1 Byte 1								Receive Buffer 1 Byte 0								uuuu uuuu uuuu uuuu
C2RX1B2	03D8	Receive Buffer 1 Byte 3								Receive Buffer 1 Byte 2								uuuu uuuu uuuu uuuu
C2RX1B3	03DA	Receive Buffer 1 Byte 5								Receive Buffer 1 Byte 4								uuuu uuuu uuuu uuuu
C2RX1B4	03DC	Receive Buffer 1 Byte 7								Receive Buffer 1 Byte 6								uuuu uuuu uuuu uuuu
C2RX0CON	03E0	—	RXRTR	RB1	RB0	DLC<3:0>				RXFUL	RXM<1:0>		—	RXRTRO	DBEN	JTOFF	FILHIT0	0000 0000 0000 0000
C2RX0SID	03E2													EXID	—	EID17	EID16	uuuu uuuu uuuu u0uu
C2RX0EID	03E4	Receive Buffer 0 Extended Identifer																uuuu uuuu uuuu uuuu
C2RX0B1	03E6	Receive Buffer 0 Byte 1								Receive Buffer 0 Byte 0								uuuu uuuu uuuu uuuu
C2RX0B2	03E8	Receive Buffer 0 Byte 3								Receive Buffer 0 Byte 2								uuuu uuuu uuuu uuuu
C2RX0B3	03EA	Receive Buffer 0 Byte 5								Receive Buffer 0 Byte 4								uuuu uuuu uuuu uuuu
C2RX0B4	03EC	Receive Buffer 0 Byte 7								Receive Buffer 0 Byte 6								uuuu uuuu uuuu uuuu
C2CTRL	03EE	—	—	CSIDLE	ABAT	—	REQOP<2:0>			OPMODE<2:0>			—	ICODE<2:0>			—	0000 0100 1000 0000
C2CFG1	03F0	—	—	—	—	—	—	—	—	SJWS<1:0>		BRP<5:0>						0000 0000 0000 0000
C2CFG2	03F2	CANCAP	WAKFIL	—	—	—	SEG2PH<2:0>			BTLMODE	SAM	SEG1PH<2:0>			PRSEG<2:0>			uu00 0uuu uuuu uuuu
C2INTF	03F4	RX0OVR	RX1OVR	TXBO	TXEP	RXEP	TXWAR	RXWAR	EWARN	IVRIF	WAKIF	ERRIF	TX2IF	TX1IF	TX0IF	RX1IF	RX0IF	0000 0000 0000 0000
C2INTE	03F6	—	—	—	—	—	—	—	—	IVRIE	WAKIE	ERRIE	TX2IE	TX1IE	TX0IE	RX1E	RX0IE	0000 0000 0000 0000
C2EC	03F8	Transmit Error Count Register								Receive Error Count Register								0000 0000 0000 0000

Legend: u = uninitialized bit

20.0 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 voltages are software selectable to either the device supply voltage (AV_{DD}/AV_{SS}) or the voltage level on the ($V_{REF}+$/$V_{REF}-$) 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.

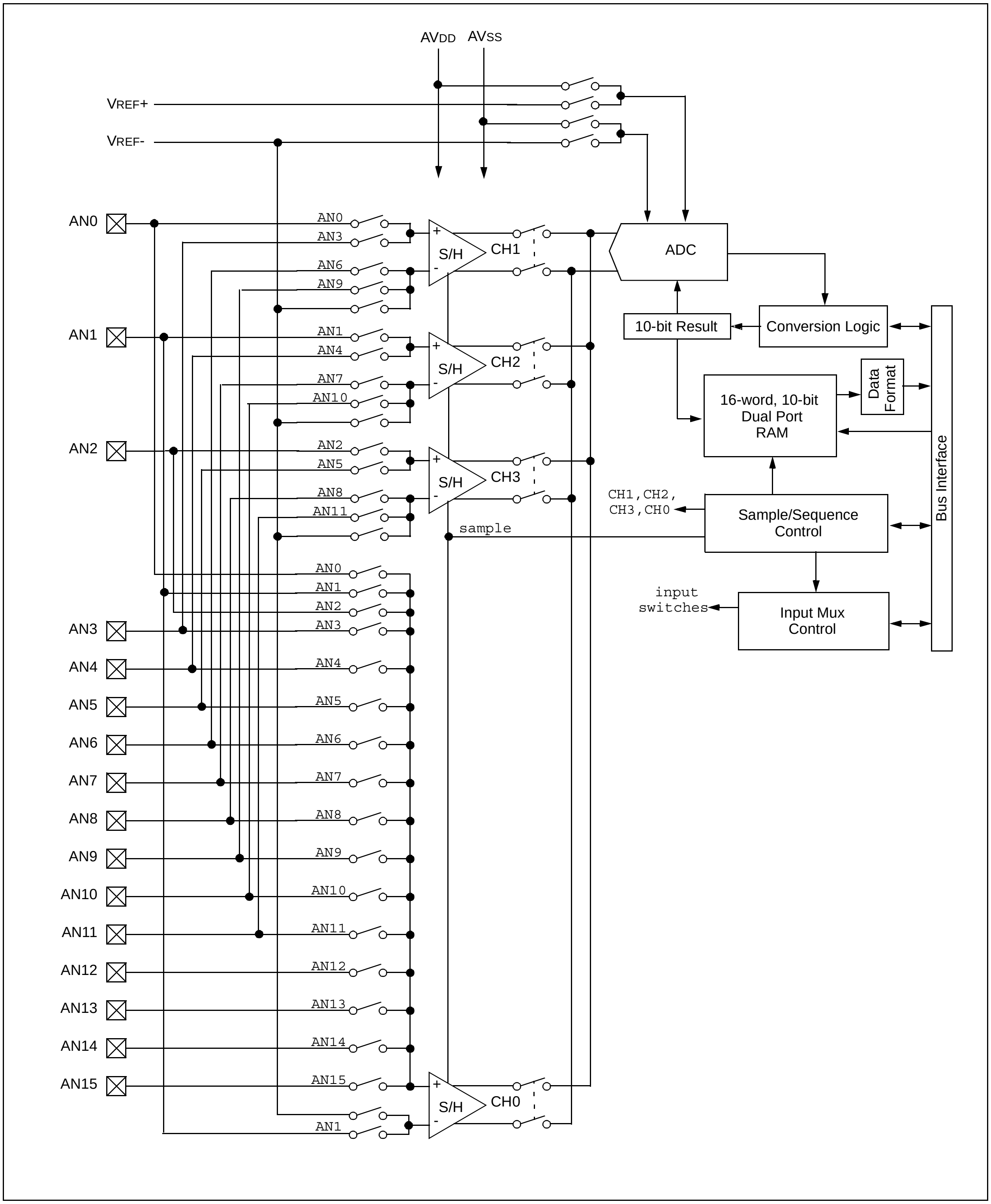
- A/D Control Register1 (ADCON1)
- A/D Control Register2 (ADCON2)
- A/D Control Register3 (ADCON3)
- A/D Input Select Register (ADCHS)
- A/D Port Configuration Register (ADPCFG)
- A/D Input Scan Selection Register (ADCSSL)

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 20-1.

FIGURE 20-1: 10-BIT HIGH SPEED A/D FUNCTIONAL BLOCK DIAGRAM



20.1 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.

20.2 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.

20.3 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 CH1 channels will be sampled and converted. If CHPS selects 4 channels, the CH0, CH1, CH2 and CH3 channels will be sampled and converted.

The SMPI bits will select the number of values loaded into the buffer 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. The input multiplexer has two sets of sample inputs: SAMPLE A and SAMPLE B. 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 alternately scanned across a selected number of analog inputs for the SAMPLE A group. The inputs are selected by the ADCSSL register. If a particular bit in the ADCSSL register is ‘1’, the corresponding input is selected. The inputs are always scanned from lower to higher numbered inputs, starting after each interrupt occurs. If the number of inputs selected is greater than the number of samples taken per interrupt, the higher numbered inputs are unused.

20.4 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` (Auto Start mode), 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, Motor Control PWM module, or external interrupts.

The SSRC bits provide for up to 5 alternate sources of sample trigger.

20.5 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 2 T_{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 multi-channel group conversion sequence.

20.6 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}.

EQUATION 20-1: A/D CONVERSION CLOCK

$$T_{AD} = T_{CY} * (0.5*(ADCS<5:0> +1))$$

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 20-1 shows the resultant T_{AD} times derived from the device operating frequencies and the A/D clock source selected.

TABLE 20-1: TYPICAL T_{AD} VS. DEVICE OPERATING FREQUENCIES

A/D Clock Period (T_{AD} Values)							
A/D Clock Source Select			Device F_{CY}				
A/D Clock	ADRC	ADCS<5:0>	30 MHz	25 MHz	12.5 MHz	6.25 MHz	1 MHz
$T_{CY}/2$	0	000000	16.67 ns[2]	20 ns[2]	40 ns[2]	80 ns[2]	500 ns
T_{CY}	0	000001	33.33 ns[2]	40 ns[2]	80 ns[2]	160 ns	1.0 µs
2 T_{CY}	0	000011	66.66 ns[2]	80 ns[2]	160 ns	320 ns	2.0 µs[3]
4 T_{CY}	0	000111	133.32 ns[2]	160 ns	320 ns	640 ns[3]	4.0 µs[3]
8 T_{CY}	0	001111	266.64 ns	320 ns	640 ns[3]	1.28 µs[3]	8.0 µs[3]
16 T_{CY}	0	011111	533.28 ns[3]	640 ns[3]	1.28 µs[3]	2.56 µs[3]	16.0 µs[3]
32 T_{CY}	0	111111	1066.56 ns[3]	1280 ns[3]	2.56 µs[3]	5.12 µs[3]	32.0 µs[3]
RC	1	xxxxxx	200 - 400 ns[1,4]	200 - 400 ns[1,4]	200 - 400 ns[1,4]	200 - 400 ns[1,4]	200 - 400 ns[1]

Note 1: The RC source has a typical T_{AD} time of 300 ns for V_{DD} > 3.0V.
2: These values violate the minimum required T_{AD} 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 F_{OSC} > 20 MHz

20.7 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 20-2. 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 20-2. 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 20-2 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 C_{HOLD} is assumed to be 5 pF for the A/D converter.

EQUATION 20-2: A/D SAMPLING TIME EQUATIONS

$$\Delta V_O = \Delta V_I \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC}+R_{SS}+R_S))})$$

$$1 - (\Delta V_O / \Delta V_I) = e^{(-T_C/C_{HOLD}(R_{IC}+R_{SS}+R_S))}$$

$$\Delta V_I = V_{IN} - V_{REF-}$$

$$\Delta V_O = n \bullet LSB - 1/2\ LSB$$

$$\Delta V_O / \Delta V_I = (n \bullet LSB - 1/2\ LSB) / n \bullet LSB$$

$$1 - (\Delta V_O / \Delta V_I) = 1 / 2n$$

$$1 / 2n = e^{(-T_C/C_{HOLD}(R_{IC}+R_{SS}+R_S))}$$

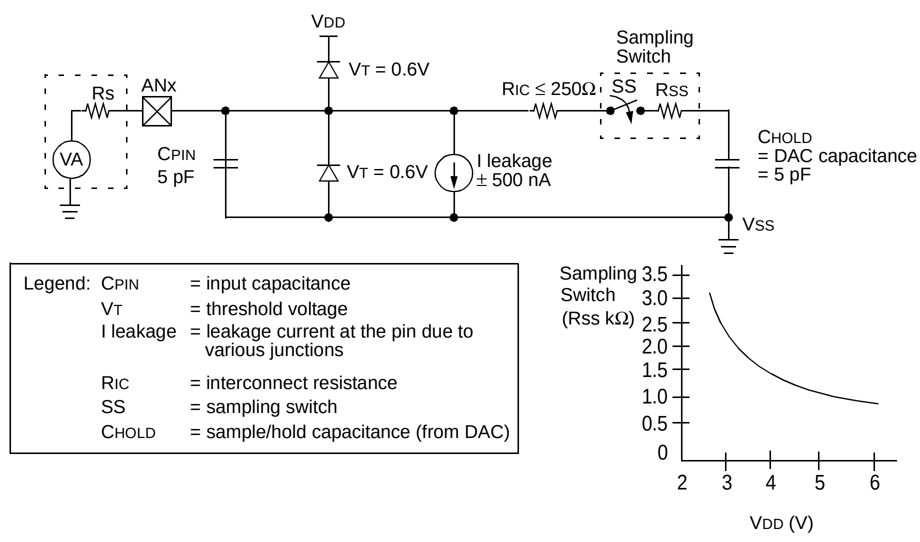
$$T_C = C_{HOLD} \bullet (R_{IC}+R_{SS}+R_S) \bullet -\ln(1/2 \bullet n)$$

$$T_{SMP} = \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time } (T_C) + \text{Temperature Coefficient } \dagger$$

† The temperature coefficient is only required for temperatures > 25°C.

$$T_{SMP} = 0.5\text{ ms} + C_{HOLD} \bullet (R_{IC}+R_{SS}+R_S) \bullet -\ln(1/2 \bullet n) + [(\text{Temp} - 25°C)(0.05\text{ ms}/°C)]$$

FIGURE 20-2: ANALOG INPUT MODEL



Note: Values shown here are untested typical values, for reference only. Exact electrical specifications are to be determined.

20.8 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.

20.9 A/D Operation During CPU SLEEP and IDLE Modes

20.9.1 A/D OPERATION DURING CPU SLEEP MODE

When the device enters SLEEP mode, all clock sources to the module are shutdown 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.

20.9.2 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.

20.10 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.

20.11 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 in right justified (integer) format.

FIGURE 20-3: 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

20.12 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 CH0SA<3:0>/CH0SB<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 not convert an analog input. Analog levels on any pin that is defined as a digital input (including the ANx pins), may cause the input buffer to consume current that exceeds the device specifications.

20.13 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 20-2: ADC REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
ADCBUF0	0280	—	—	—	—	—	—	ADC Data Buffer 0										uuuu uuuu uuuu uuuu
ADCBUF1	0282	—	—	—	—	—	—	ADC Data Buffer 1										uuuu uuuu uuuu uuuu
ADCBUF2	0284	—	—	—	—	—	—	ADC Data Buffer 2										uuuu uuuu uuuu uuuu
ADCBUF3	0286	—	—	—	—	—	—	ADC Data Buffer 3										uuuu uuuu uuuu uuuu
ADCBUF4	0288	—	—	—	—	—	—	ADC Data Buffer 4										uuuu uuuu uuuu uuuu
ADCBUF5	028A	—	—	—	—	—	—	ADC Data Buffer 5										uuuu uuuu uuuu uuuu
ADCBUF6	028C	—	—	—	—	—	—	ADC Data Buffer 6										uuuu uuuu uuuu uuuu
ADCBUF7	028E	—	—	—	—	—	—	ADC Data Buffer 7										uuuu uuuu uuuu uuuu
ADCBUF8	0290	—	—	—	—	—	—	ADC Data Buffer 8										uuuu uuuu uuuu uuuu
ADCBUF9	0292	—	—	—	—	—	—	ADC Data Buffer 9										uuuu uuuu uuuu uuuu
ADCBUFA	0294	—	—	—	—	—	—	ADC Data Buffer 10										uuuu uuuu uuuu uuuu
ADCBUFB	0296	—	—	—	—	—	—	ADC Data Buffer 11										uuuu uuuu uuuu uuuu
ADCBUFC	0298	—	—	—	—	—	—	ADC Data Buffer 12										uuuu uuuu uuuu uuuu
ADCBUFD	029A	—	—	—	—	—	—	ADC Data Buffer 13										uuuu uuuu uuuu uuuu
ADCBUFE	029C	—	—	—	—	—	—	ADC Data Buffer 14										uuuu uuuu uuuu uuuu
ADCBUFF	029E	—	—	—	—	—	—	ADC Data Buffer 15										uuuu uuuu uuuu uuuu
ADCON1	02A0	ADON	—	ADSIDL	ADSTBY	—	—	FORM<1:0>		SSRC<2:0>			—	SIMSAM	ASAM	SAMP	CONV	0000 0000 0000 0000
ADCON2	02A2	VCFG<2:0>			OFFCAL	—	CSCNA	CHPS<1:0>		BUFS	—	SMPI<3:0>				BUFM	ALTS	0000 0000 0000 0000
ADCON3	02A4	—	—	—	SAMC<4:0>					ADRC	—	ADCS<5:0>						0000 0000 0000 0000
ADCHS	02A6	CH123NB<1:0>		CH123SB	CH0NB	CH0SB<3:0>				CH123NA<1:0>		CH123SA	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

Legend: u = uninitialized bit

21.0 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

21.1 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 postscaler for system power savings
- A Fail-Safe Clock Monitor (FSCM) that detects clock failure and takes fail-safe measures
- Clock Control Register OSCCON
- Configuration bits for main oscillator selection

Table 21-1 provides a summary of the dsPIC30F Oscillator Operating modes. A simplified diagram of the oscillator system is shown in Figure 21-1.

TABLE 21-1: OSCILLATOR OPERATING MODES

Oscillator Mode	Description
XTL	200 kHz - 4 MHz crystal on OSC1:OSC2.
XT	4 MHz - 10 MHz crystal on OSC1:OSC2.
XT w/ PLL 4x	4 MHz - 10 MHz crystal on OSC1:OSC2. 4x PLL enabled.
XT w/ PLL 8x	4 MHz - 10 MHz crystal on OSC1:OSC2. 8x PLL enabled.
XT w/ PLL 16x	4 MHz - 10 MHz crystal on OSC1:OSC2. 16x PLL enabled(1).
LP	32 kHz crystal on 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.

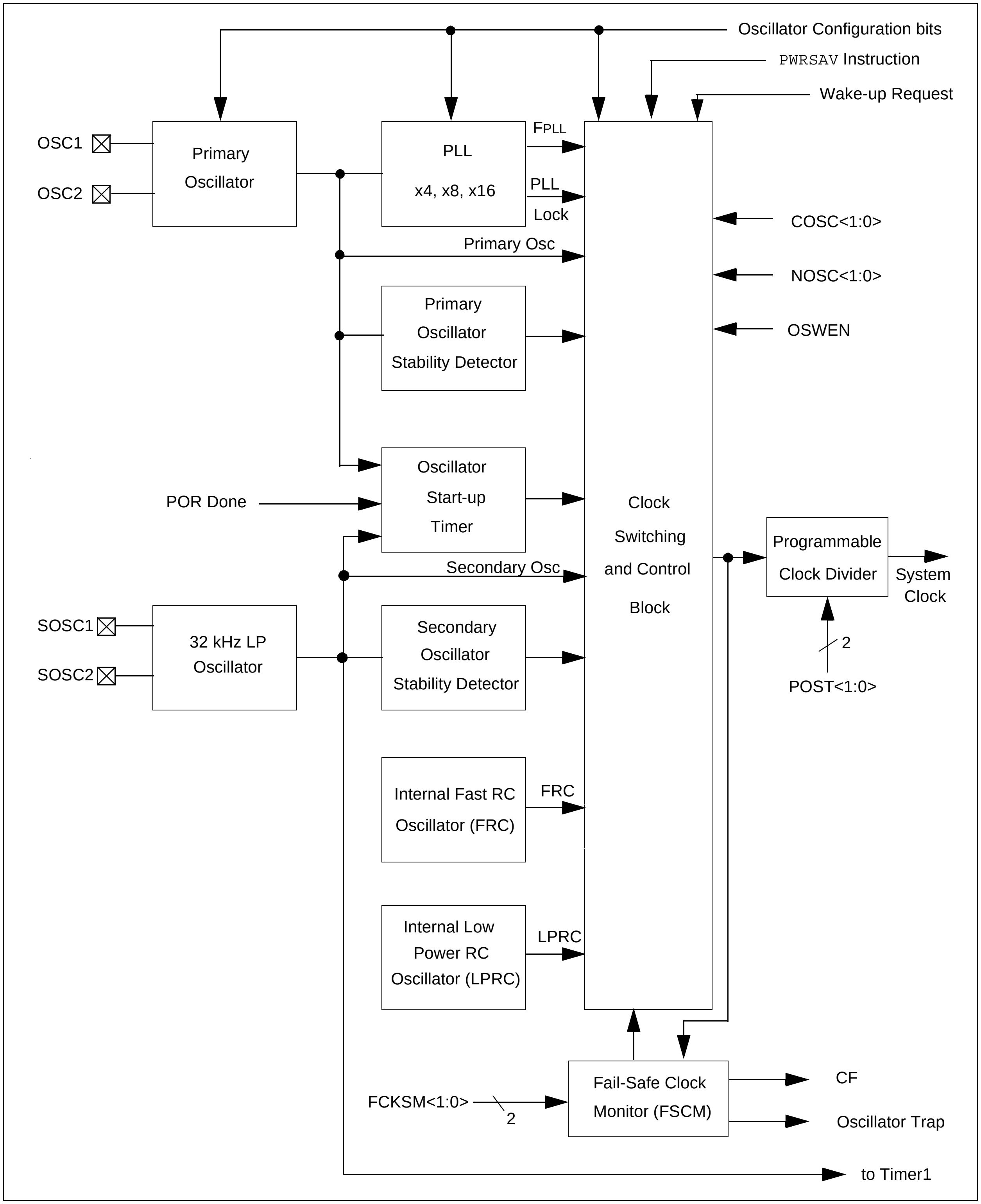
Note 1: dsPIC30F maximum operating frequency of 120 MHz must be met.

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.

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 21-1: OSCILLATOR SYSTEM BLOCK DIAGRAM



21.2 Oscillator Configurations

21.2.1 INITIAL CLOCK SOURCE SELECTION

While coming out of Power-on Reset or Brown-out Reset, the device selects its clock source based on:

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 21-2.

TABLE 21-2: CONFIGURATION BIT VALUES FOR CLOCK SELECTION

Oscillator Mode	Oscillator Source	FOS1	FOS0	FPR3	FPR2	FPR1	FPR0	OSC2 Function
EC	Primary	1	1	1	0	1	1	CLKO
ECIO	Primary	1	1	1	1	0	0	I/O
EC w/ PLL 4x	Primary	1	1	1	1	0	1	I/O
EC w/ PLL 8x	Primary	1	1	1	1	1	0	I/O
EC w/ PLL 16x	Primary	1	1	1	1	1	1	I/O
ERC	Primary	1	1	1	0	0	1	CLKO
ERCIO	Primary	1	1	1	0	0	0	I/O
XT	Primary	1	1	0	1	0	0	OSC2
XT w/ PLL 4x	Primary	1	1	0	1	0	1	OSC2
XT w/ PLL 8x	Primary	1	1	0	1	1	0	OSC2
XT w/ PLL 16x	Primary	1	1	0	1	1	1	OSC2
XTL	Primary	1	1	0	0	0	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.

21.2.2 OSCILLATOR START-UP TIMER (OST)

In order to ensure that a crystal oscillator (or ceramic resonator) has started and stabilized, an oscillator start-up timer is included. It is a simple 10-bit counter that counts 1024 T_{OSC} cycles before releasing the oscillator clock to the rest of the system. The time-out period is designated as T_{OST}. The T_{OST} time is involved every time the oscillator has to restart, i.e., on POR, BOR and wake-up from SLEEP. The oscillator start-up timer is applied to the LP Oscillator, XT, XTL, and HS modes (upon wake-up from SLEEP, POR and BOR) for the primary oscillator.

21.2.3 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

21.2.4 PHASE LOCKED LOOP (PLL)

The PLL multiplies the clock which is generated by the primary oscillator. The PLL is selectable to have either gains of x4, x8, and x16. Input and output frequency ranges are summarized in the table below.

TABLE 21-3: PLL FREQUENCY RANGE

F_{IN}	PLL Multiplier	F_{OUT}
4 MHz - 10 MHz	x4	16 MHz - 40 MHz
4 MHz - 10 MHz	x8	32 MHz - 80 MHz
4 MHz - 7.5 MHz	x16	64 MHz - 160 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.

21.2.5 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’.

21.2.6 LOW POWER RC OSCILLATOR (LPRC)

The LPRC oscillator is a component of the Watchdog Timer (WDT) and oscillates at a nominal frequency of 512 kHz. The LPRC oscillator is the clock source for the Power-up Timer (PWRT) circuit, WDT, and clock monitor circuits. It may also be used to provide a low frequency clock source option for applications where power consumption is critical, and timing accuracy is not required

The LPRC oscillator is always enabled at a Power-on Reset, because it is the clock source for the PWRT. After the PWRT expires, the LPRC oscillator 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.

21.2.7 FAIL-SAFE CLOCK MONITOR

The Fail-Safe Clock Monitor (FSCM) allows the device to continue to operate even in the event of an oscillator failure. The FSCM function is enabled by appropriately programming the FCKSM 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 shutdown. The user may decide to treat the Trap as a warm RESET by simply loading the RESET address into the oscillator fail trap vector. In this event, the CF (Clock Fail) status bit (OSCCON<3>) is also set whenever a clock failure is recognized.

In the event of a clock failure, the WDT is unaffected and continues to run on the LPRC clock.

If the oscillator has a very slow start-up time coming out of POR, BOR or SLEEP, it is possible that the PWRT timer will expire before the oscillator has started. In such cases, the FSCM 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.

21.2.8 PROTECTION AGAINST ACCIDENTAL WRITES TO OSCCON

A write to the OSCCON register is intentionally made difficult, because it controls clock switching and clock scaling.

To write to the OSCCON low byte, the following code sequence must be executed without any other instructions in between:

- `Byte Write "0x46" to OSCCON low`
- `Byte Write "0x57" to OSCCON low`

Byte Write is allowed for one instruction cycle. Write desired value or use bit manipulation instruction.

To write to the OSCCON high byte, the following instructions must be executed without any other instructions in between:

- `Byte Write "0x78" to OSCCON high`
- `Byte Write "0x9A" to OSCCON high`

Byte Write is allowed for one instruction cycle. Write "Desired Value" or use bit manipulation instruction.

21.3 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 (BOR)
f) `RESET` Instruction
g) Reset cause by trap lockup (TRAPR)
h) Reset caused by illegal opcode, or by using an uninitialized W register as an address pointer (IOPUWR)

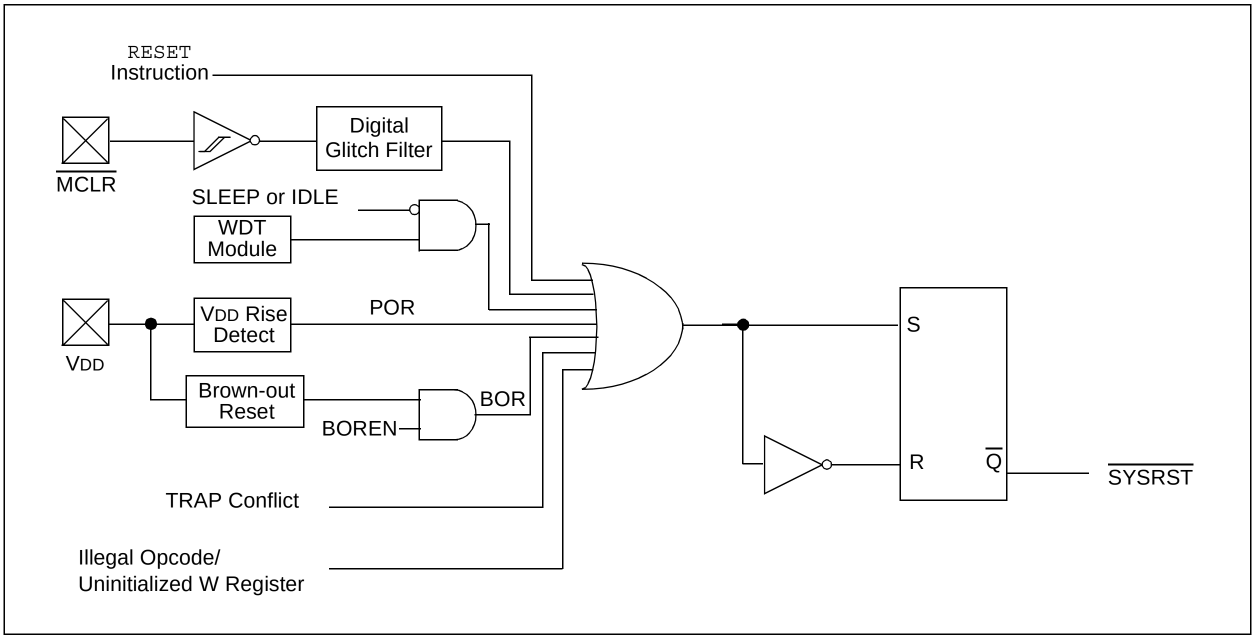
Different registers are affected in different ways by various RESET conditions. Most registers are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. Status bits from the RCON register are set or cleared differently in different RESET situations, as indicated in Table 21-4. 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 21-2.

A $\overline{\text{MCLR}}$ noise filter is provided in the $\overline{\text{MCLR}}$ Reset path. The filter detects and ignores small pulses.

Internally generated RESETS do not drive $\overline{\text{MCLR}}$ pin low.

FIGURE 21-2: RESET SYSTEM BLOCK DIAGRAM



21.3.1 POR: POWER-ON RESET

A power-on event will generate an internal POR 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). 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 10 μs 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 21-3 through Figure 21-5.

FIGURE 21-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD)

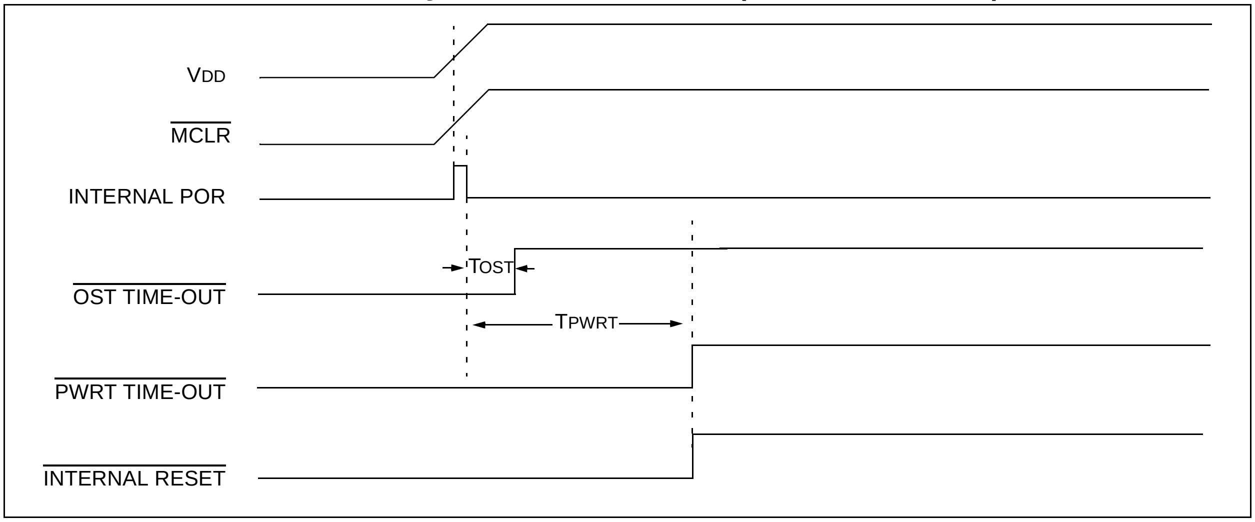


FIGURE 21-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

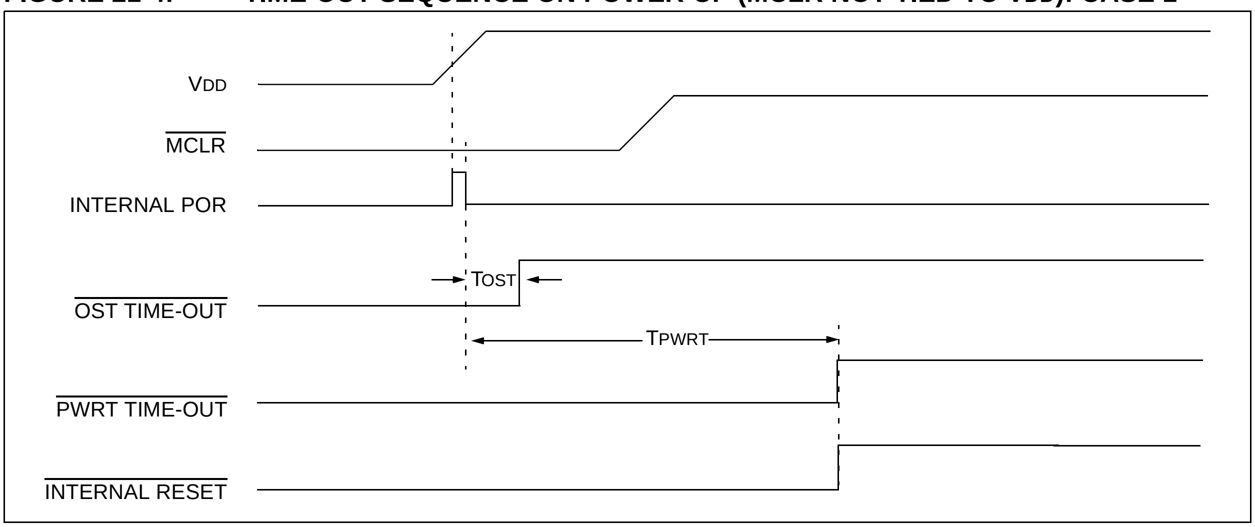
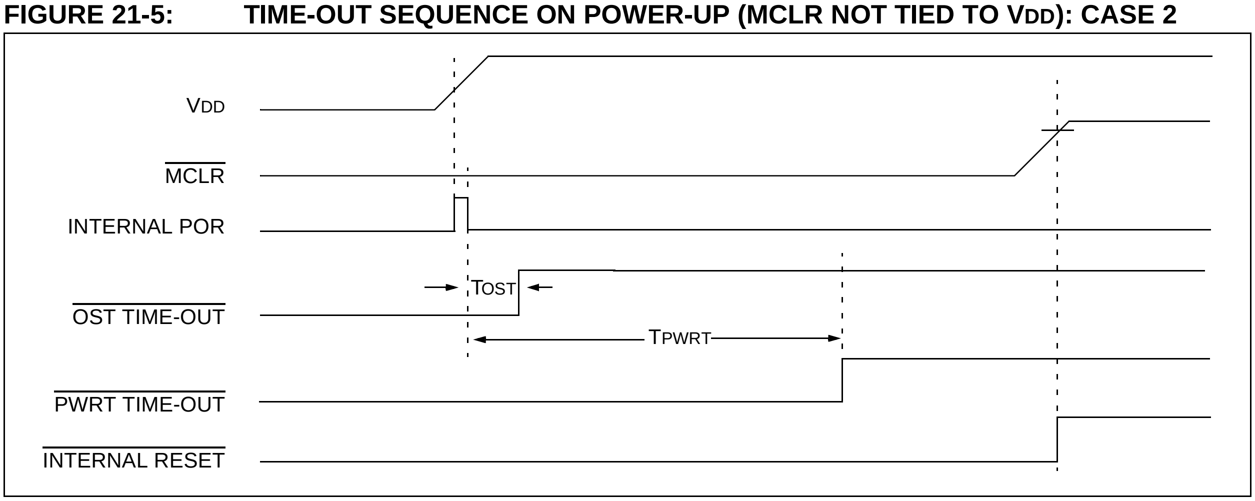


FIGURE 21-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2



21.3.1.1 POR with Long Crystal Start-up Time (with FSCM Enabled)

The oscillator start-up circuitry is not linked to the POR circuitry. Some crystal circuits (especially low frequency crystals) 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.

21.3.1.2 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.

21.3.2 BOR: PROGRAMMABLE BROWN-OUT RESET

The BOR (Brown-out Reset) module is based on an internal voltage reference circuit. The main purpose of the BOR module is to generate a device RESET when a brown-out condition occurs. Brown-out conditions are generally caused by glitches on the AC mains, i.e., missing 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

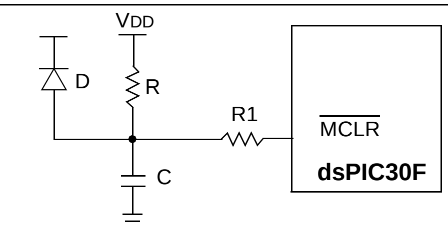
Note:	The BOR voltage trip points indicated here are nominal values provided for design guidance only. Refer to the Electrical Specifications in the specific device data sheet for BOR voltage limit specifications.

A BOR will generate a RESET pulse which will reset the device. The BOR 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 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 time-out (T_{POR}) and the PWRT time-out (T_{PWRT}) will be applied before the internal RESET is released. If $T_{PWRT} = 0$ and a crystal oscillator is being used, then a nominal delay of $T_{FSCM} = 100\ \mu s$ is applied. The total delay in this case is ($T_{POR} + T_{FSCM}$).

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 21-6: 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 should be suitably chosen so as to make sure that the voltage drop across R does not violate the device's electrical specification.

3: R1 should be suitably chosen so as to limit any current flowing into $\overline{MCLR}$ from external capacitor C, in the event of $\overline{MCLR}/V_{PP}$ 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 21-4 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 21-4: INITIALIZATION CONDITION FOR RCON REGISTER CASE 1

Condition	Program Counter	TRAPR	IOPUWR	EXTR	SWR	WDTO	IDLE	SLEEP	POR	BOR
Power-on Reset	0x000000	0	0	0	0	0	0	0	1	1
Brown-out Reset	0x000000	0	0	0	0	0	0	0	0	1
$\overline{\text{MCLR}}$ Reset during normal operation	0x000000	0	0	1	0	0	0	0	0	0
Software Reset during normal operation	0x000000	0	0	0	1	0	0	0	0	0
$\overline{\text{MCLR}}$ Reset during SLEEP	0x000000	0	0	1	0	0	0	1	0	0
$\overline{\text{MCLR}}$ Reset during IDLE	0x000000	0	0	1	0	0	1	0	0	0
WDT Time-out Reset	0x000000	0	0	0	0	1	0	0	0	0
WDT Wake-up	PC + 2	0	0	0	0	1	0	1	0	0
Interrupt Wake-up from SLEEP	PC + 2(1)	0	0	0	0	0	0	1	0	0
Clock Failure Trap	0x000004	0	0	0	0	0	0	0	0	0
Trap Reset	0x000000	1	0	0	0	0	0	0	0	0
Illegal Operation Trap	0x000000	0	1	0	0	0	0	0	0	0

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 21-5 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 21-5: INITIALIZATION CONDITION FOR RCON REGISTER CASE 2

Condition	Program Counter	TRAPR	IOPUWR	EXTR	SWR	WDTO	IDLE	SLEEP	POR	BOR
Power-on Reset	0x000000	0	0	0	0	0	0	0	1	1
Brown-out Reset	0x000000	u	u	u	u	u	u	u	0	1
$\overline{\text{MCLR}}$ Reset during normal operation	0x000000	u	u	1	0	0	0	0	u	u
Software Reset during normal operation	0x000000	u	u	0	1	0	0	0	u	u
$\overline{\text{MCLR}}$ Reset during SLEEP	0x000000	u	u	1	u	0	0	1	u	u
$\overline{\text{MCLR}}$ Reset during IDLE	0x000000	u	u	1	u	0	1	0	u	u
WDT Time-out Reset	0x000000	u	u	0	0	1	0	0	u	u
WDT Wake-up	PC + 2	u	u	u	u	1	u	1	u	u
Interrupt Wake-up from SLEEP	PC + 2(1)	u	u	u	u	u	u	1	u	u
Clock Failure Trap	0x000004	u	u	u	u	u	u	u	u	u
Trap Reset	0x000000	1	u	u	u	u	u	u	u	u
Illegal Operation Reset	0x000000	u	1	u	u	u	u	u	u	u

Legend: u = unchanged, 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.

21.4 Watchdog Timer (WDT)

21.4.1 WATCHDOG TIMER OPERATION

The primary function of the Watchdog Timer (WDT) is to reset the processor in the event of a software malfunction. The WDT is a free running timer, 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.

21.4.2 ENABLING AND DISABLING THE WDT

The Watchdog Timer can be “Enabled” or “Disabled” only through a configuration bit (FWDTEN) in the configuration register FWDT.

Setting FWDTEN = 1 enables the Watchdog Timer. The enabling is done when programming the device. By default, after chip-erase, FWDTEN bit = 1. Any device programmer capable of programming dsPIC30F devices allows programming of this and other configuration bits 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.

21.5 Power Saving Modes

There are two power saving states that can be entered through the execution of a special instruction, `PWRSAV`.

These are: SLEEP and IDLE.

The format of the `PWRSAV` instruction is as follows:

`PWRSAV <parameter>`, where ‘`parameter`’ defines IDLE or SLEEP mode.

21.5.1 SLEEP MODE

In SLEEP mode, the clock to the CPU and the peripherals is shutdown. If an on-chip oscillator is being used, it is shutdown.

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 $\overline{MCLR}$)
- 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 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, T_{POR}, T_{LOCK} and T_{PWRT} delays are applied.

If EC, FRC, LPRC or EXTRC oscillators are used, then a delay of T_{POR} (~ 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 T_{POR}. 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. The processor will process the interrupt and branch to the ISR. The SLEEP status bit in RCON register is set upon wake-up.

Note:	In spite of various delays applied (T_{POR}, T_{LOCK} and T_{PWRT}), 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, 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.

21.5.2 IDLE MODE

In IDLE mode, the clock to the CPU is shutdown 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. The processor will process the interrupt and branch to the ISR. 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, 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.

21.6 Device Configuration Registers

The configuration bits in each device configuration register specify some of the Device modes and are programmed by a device programmer, or by using the In-Circuit Serial Programming (ICSP) feature of the device. Each device configuration register is a 24-bit register, but only the lower 16 bits of each register are used to hold configuration data. There are four device configuration registers available to the user:

1. FOSC (0xF80000): Oscillator Configuration Register
2. FWDT (0xF80002): Watchdog Timer Configuration Register
3. FBORPOR (0xF80004): BOR and POR Configuration Register
4. FGS (0xF8000A): General Code Segment Configuration Register

The placement of the configuration bits is automatically handled when you select the device in your device programmer. The desired state of the configuration bits may be specified in the source code (dependent on the language tool used), or through the programming interface. After the device has been programmed, the application software may read the configuration bit values through the table read instructions. For additional information, please refer to the Programming Specifications of the device.

21.7 Peripheral Module Disable (PMD) Registers

The peripheral module disable (PMD) registers provide a method to disable a peripheral module by stopping all clock sources supplied to that module. When a peripheral is disabled via the appropriate PMD control bit, the peripheral is in a minimum power consumption state. The control and status registers associated with the peripheral will also be disabled, so writes to those registers will have no effect and read values will be invalid.

A peripheral module will only be enabled if both the associated bit in the the PMD register is cleared, and the peripheral is supported by the specific dsPIC variant. If the peripheral is present in the device, it is enabled in the PMD register by default.

TABLE 21-6: SYSTEM INTEGRATION REGISTER MAP

SFR Name	Addr.	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
RCON	0740	TRAPR	IOPUWR	BGST	LVDEN	LVDL<3:0>				EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR	Depends on type of RESET.
OSCCON	0742	—	—	COSC<1:0>		—	—	NOSC<1:0>		POST<1:0>		LOCK	—	CF	—	LPOSCEN	OSWEN	Depends on configuration bits.
PMD1	0770	T5MD	T4MD	T3MD	T2MD	T1MD	QEIMD	PWMMD	—	I2CMD	U2MD	U1MD	SPI2MD	SPI1MD	C2MD	C1MD	—	0000 0000 0000 0000
PMD2	0772	IC8MD	IC7MD	IC6MD	IC5MD	IC4MD	IC3MD	IC2MD	IC1MD	OC8MD	OC7MD	OC6MD	OC5MD	OC4MD	OC3MD	OC2MD	OC1MD	0000 0000 0000 0000

Legend: u = uninitialized bit

TABLE 21-7: DEVICE CONFIGURATION REGISTER MAP

File Name	Addr.	Bits 23-16	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FOSC	F80000	—	FSCKM<1:0>		—	—	—	—	FGS<1:0>		—	—	—	—	FG1<3:0>			
FWDT	F80002	—	FWDTEN	—	—	—	—	—	—	—	—	—	FWPSA<1:0>		FWPSB<3:0>			
FBORPOR	F80004	—	MCLREN	—	—	—	—	PWMPIN	HPOL	LPOL	BOREN	—	BORV<1:0>		—	—	FPWRT<1:0>	
FGS	F8000A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	GCP	GWRP

22.0 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 22-1 shows the general symbols used in describing the instructions.

The dsPIC30F instruction set summary in Table 22-2 lists all the instructions along with the status flags affected by each instruction.

Most word or byte-oriented W register instructions (including barrel shift instructions) have three operands:

- The first source operand, which is typically a register ’Wb’ without any address modifier
- The second source operand, which is typically a register ’Ws’ with or without an address modifier
- The destination of the result, which is typically a register ’Wd’ with or without an address modifier

However, word or byte-oriented file register instructions have two operands:

- The file register specified by the value ’f’
- The destination, which could either be the file register ’f’ or the W0 register, which is denoted as ’WREG’

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 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 22-1: 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, Z	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}
Wb	Base W register ∈ {W0..W15}
Wd	Destination W register ∈ { Wd, [Wd], [Wd++], [Wd--], [++Wd], [--Wd] }

TABLE 22-1: SYMBOLS USED IN ROADRUNNER OPCODE DESCRIPTIONS (CONTINUED)

Field	Description
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 22-2: 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,Z
		ADD f,WREG	WREG = f + WREG	1	1	C,DC,N,OV,Z
		ADD #lit10,Wn	Wd = lit10 + Wd	1	1	C,DC,N,OV,Z
		ADD Wb,Ws,Wd	Wd = Wb + Ws	1	1	C,DC,N,OV,Z
		ADD Wb,#lit5,Wd	Wd = Wb + lit5	1	1	C,DC,N,OV,Z
		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,Z
		ADDC f,WREG	WREG = f + WREG + (C)	1	1	C,DC,N,OV,Z
		ADDC #lit10,Wn	Wd = lit10 + Wd + (C)	1	1	C,DC,N,OV,Z
		ADDC Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1	C,DC,N,OV,Z
		ADDC Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1	C,DC,N,OV,Z
3	AND	AND f	f = f .AND. WREG	1	1	N,Z
		AND f,WREG	WREG = f .AND. WREG	1	1	N,Z
		AND #lit10,Wn	Wd = lit10 .AND. Wd	1	1	N,Z
		AND Wb,Ws,Wd	Wd = Wb .AND. Ws	1	1	N,Z
		AND Wb,#lit5,Wd	Wd = Wb .AND. lit5	1	1	N,Z
4	ASR	ASR f	f = Arithmetic Right Shift f	1	1	C,N,OV,Z
		ASR f,WREG	WREG = Arithmetic Right Shift f	1	1	C,N,OV,Z
		ASR Ws,Wd	Wd = Arithmetic Right Shift Ws	1	1	C,N,OV,Z
		ASR Wb,Wns,Wnd	Wnd = Arithmetic Right Shift Wb by Wns	1	1	N,Z
		ASR Wb,#lit5,Wnd	Wnd = Arithmetic Right Shift Wb by lit5	1	1	N,Z
5	BCLR	BCLR f,#bit4	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 f,#bit4	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 Z bit to Ws<Wb>	1	1	None
9	BTG	BTG f,#bit4	Bit Toggle f	1	1	None
		BTG Ws,#bit4	Bit Toggle Ws	1	1	None
10	BTSC	BTSC f,#bit4	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 22-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax		Description	# of words	# of cycles	Status Flags Affected
11	BTSS	BTSS	f,#bit4	Bit Test f, Skip if Set	1	1 (2 or 3)	None
		BTSS	Ws,#bit4	Bit Test Ws, Skip if Set	1	1 (2 or 3)	None
12	BTST	BTST	f,#bit4	Bit Test f	1	1	Z
		BTST.C	Ws,#bit4	Bit Test Ws to C	1	1	C
		BTST.Z	Ws,#bit4	Bit Test Ws to Z	1	1	Z
		BTST.C	Ws,Wb	Bit Test Ws<Wb> to C	1	1	C
		BTST.Z	Ws,Wb	Bit Test Ws<Wb> to Z	1	1	Z
13	BTSTS	BTSTS	f,#bit4	Bit Test then Set f	1	1	Z
		BTSTS.C	Ws,#bit4	Bit Test Ws to C, then Set	1	1	C
		BTSTS.Z	Ws,#bit4	Bit Test Ws to Z, then Set	1	1	Z
14	CALL	CALL	lit23	Call subroutine	2	2	None
		CALL	Wn	Call indirect subroutine	1	2	None
15	CLR	CLR	f	f = 0x0000	1	1	None
		CLR	WREG	WREG = 0x0000	1	1	None
		CLR	Ws	Ws = 0x0000	1	1	None
		CLR	Acc,Wx,Wxd,Wy,Wyd,AWB	Clear Accumulator	1	1	OA,OB,SA,SB
16	CLRWDT	CLRWDT		Clear Watchdog Timer	1	1	WDTO,SLEEP
17	COM	COM	f	f = $\overline{f}$	1	1	N,Z
		COM	f,WREG	WREG = $\overline{f}$	1	1	N,Z
		COM	Ws,Wd	Wd = $\overline{Ws}$	1	1	N,Z
18	CP	CP	f	Compare f with WREG	1	1	C,DC,N,OV,Z
		CP	Wb,#lit5	Compare Wb with lit5	1	1	C,DC,N,OV,Z
		CP	Wb,Ws	Compare Wb with Ws (Wb - Ws)	1	1	C,DC,N,OV,Z
19	CP0	CP0	f	Compare f with 0x0000	1	1	C,DC,N,OV,Z
		CP0	Ws	Compare Ws with 0x0000	1	1	C,DC,N,OV,Z
20	CP1	CP1	f	Compare f with 0xFFFF	1	1	C,DC,N,OV,Z
		CP1	Ws	Compare Ws with 0xFFFF	1	1	C,DC,N,OV,Z
21	CPB	CPB	f	Compare f with WREG, with Borrow	1	1	C,DC,N,OV,Z
		CPB	Wb,#lit5	Compare Wb with lit5, with Borrow	1	1	C,DC,N,OV,Z
		CPB	Wb,Ws	Compare Wb with Ws, with Borrow (Wb - Ws - $\overline{C}$)	1	1	C,DC,N,OV,Z
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	DAW	Wn	Wn = decimal adjust Wn	1	1	C
27	DEC	DEC	f	f = f -1	1	1	C,DC,N,OV,Z
		DEC	f,WREG	WREG = f -1	1	1	C,DC,N,OV,Z
		DEC	Ws,Wd	Wd = Ws - 1	1	1	C,DC,N,OV,Z
28	DEC2	DEC2	f	f = f -2	1	1	C,DC,N,OV,Z
		DEC2	f,WREG	WREG = f -2	1	1	C,DC,N,OV,Z
		DEC2	Ws,Wd	Wd = Ws - 2	1	1	C,DC,N,OV,Z
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 22-2: 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,Z,C
		DIV.SD	Wm,Wn	Signed 32/16-bit Integer Divide	1	18	N,Z,C
		DIV.U	Wm,Wn	Unsigned 16/16-bit Integer Divide	1	18	N,Z,C
		DIV.UD	Wm,Wn	Unsigned 32/16-bit Integer Divide	1	18	N,Z,C
33	DIVF	DIVF	Wm,Wn	Signed 16/16-bit Fractional Divide	1	18	N,Z,C
34	DO	DO	#lit14,Expr	Do code to PC+Expr, lit14+1 times	2	2	None
		DO	Wn,Expr	Do code to PC+Expr, (Wn)+1 times	2	2	None
35	ED	ED	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean Distance	1	1	OA,OB,OAB, SA,SB,SAB
36	EDAC	EDAC	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean Distance Accumulate	1	1	OA,OB,OAB, SA,SB,SAB
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,Z
		INC	f,WREG	WREG = f + 1	1	1	C,DC,N,OV,Z
		INC	Ws,Wd	Wd = Ws + 1	1	1	C,DC,N,OV,Z
43	INC2	INC2	f	f = f + 2	1	1	C,DC,N,OV,Z
		INC2	f,WREG	WREG = f + 2	1	1	C,DC,N,OV,Z
		INC2	Ws,Wd	Wd = Ws + 2	1	1	C,DC,N,OV,Z
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,Z
		IOR	f,WREG	WREG = f .IOR. WREG	1	1	N,Z
		IOR	#lit10,Wn	Wd = lit10 .IOR. Wd	1	1	N,Z
		IOR	Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1	N,Z
		IOR	Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1	N,Z
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,Z
		LSR	f,WREG	WREG = Logical Right Shift f	1	1	C,N,OV,Z
		LSR	Ws,Wd	Wd = Logical Right Shift Ws	1	1	C,N,OV,Z
		LSR	Wb,Wns,Wnd	Wnd = Logical Right Shift Wb by Wns	1	1	N,Z
		LSR	Wb,#lit5,Wnd	Wnd = Logical Right Shift Wb by lit5	1	1	N,Z
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 22-2: 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,Z
		MOV f,WREG	Move f to WREG	1	1	N,Z
		MOV #lit16,Wn	Move 16-bit literal to Wn	1	1	None
		MOV.b #lit8,Wn	Move 8-bit literal to Wn	1	1	None
		MOV Wn,f	Move Wn to f	1	1	None
		MOV Wso,Wdo	Move Ws to Wd	1	1	None
		MOV WREG,f	Move WREG to f	1	1	N,Z
		MOV.D Wns,Wd	Move Double from W(ns):W(ns+1) to Wd	1	2	None
		MOV.D Ws,Wnd	Move Double from Ws to W(nd+1):W(nd)	1	2	None
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, SA,SB,SAB
		MPY Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square Wm to Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
54	MPY.N	MPY.N Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wm by Wn) to Accumulator	1	1	None
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,Z
		NEG f,WREG	WREG = $\overline{f}$ + 1	1	1	C,DC,N,OV,Z
		NEG Ws,Wd	Wd = $\overline{Ws}$ + 1	1	1	C,DC,N,OV,Z
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	None
		REPEAT Wn	Repeat Next Instruction (Wn)+1 times	1	2	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,Z
		RLC f,WREG	WREG = Rotate Left through Carry f	1	1	C,N,Z
		RLC Ws,Wd	Wd = Rotate Left through Carry Ws	1	1	C,N,Z

TABLE 22-2: 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,Z
		RLNC	f,WREG	WREG = Rotate Left (No Carry) f	1	1	N,Z
		RLNC	Ws,Wd	Wd = Rotate Left (No Carry) Ws	1	1	N,Z
70	RRC	RRC	f	f = Rotate Right through Carry f	1	1	C,N,Z
		RRC	f,WREG	WREG = Rotate Right through Carry f	1	1	C,N,Z
		RRC	Ws,Wd	Wd = Rotate Right through Carry Ws	1	1	C,N,Z
71	RRNC	RRNC	f	f = Rotate Right (No Carry) f	1	1	N,Z
		RRNC	f,WREG	WREG = Rotate Right (No Carry) f	1	1	N,Z
		RRNC	Ws,Wd	Wd = Rotate Right (No Carry) Ws	1	1	N,Z
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,Z
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,Z
		SL	f,WREG	WREG = Left Shift f	1	1	C,N,OV,Z
		SL	Ws,Wd	Wd = Left Shift Ws	1	1	C,N,OV,Z
		SL	Wb,Wns,Wnd	Wnd = Left Shift Wb by Wns	1	1	N,Z
		SL	Wb,#lit5,Wnd	Wnd = Left Shift Wb by lit5	1	1	N,Z
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,Z
		SUB	f,WREG	WREG = f - WREG	1	1	C,DC,N,OV,Z
		SUB	#lit10,Wn	Wn = Wn - lit10	1	1	C,DC,N,OV,Z
		SUB	Wb,Ws,Wd	Wd = Wb - Ws	1	1	C,DC,N,OV,Z
		SUB	Wb,#lit5,Wd	Wd = Wb - lit5	1	1	C,DC,N,OV,Z
78	SUBB	SUBB	f	f = f - WREG - ($\overline{C}$)	1	1	C,DC,N,OV,Z
		SUBB	f,WREG	WREG = f - WREG - ($\overline{C}$)	1	1	C,DC,N,OV,Z
		SUBB	#lit10,Wn	Wn = Wn - lit10 - ($\overline{C}$)	1	1	C,DC,N,OV,Z
		SUBB	Wb,Ws,Wd	Wd = Wb - Ws - ($\overline{C}$)	1	1	C,DC,N,OV,Z
		SUBB	Wb,#lit5,Wd	Wd = Wb - lit5 - ($\overline{C}$)	1	1	C,DC,N,OV,Z
79	SUBR	SUBR	f	f = WREG - f	1	1	C,DC,N,OV,Z
		SUBR	f,WREG	WREG = WREG - f	1	1	C,DC,N,OV,Z
		SUBR	Wb,Ws,Wd	Wd = Ws - Wb	1	1	C,DC,N,OV,Z
		SUBR	Wb,#lit5,Wd	Wd = lit5 - Wb	1	1	C,DC,N,OV,Z
80	SUBBR	SUBBR	f	f = WREG - f - ($\overline{C}$)	1	1	C,DC,N,OV,Z
		SUBBR	f,WREG	WREG = WREG -f - ($\overline{C}$)	1	1	C,DC,N,OV,Z
		SUBBR	Wb,Ws,Wd	Wd = Ws - Wb - ($\overline{C}$)	1	1	C,DC,N,OV,Z
		SUBBR	Wb,#lit5,Wd	Wd = lit5 - Wb - ($\overline{C}$)	1	1	C,DC,N,OV,Z
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	ULNK	ULNK		Unlink frame pointer	1	1	None
87	XOR	XOR	f	f = f .XOR. WREG	1	1	N,Z
		XOR	f,WREG	WREG = f .XOR. WREG	1	1	N,Z
		XOR	#lit10,Wn	Wd = lit10 .XOR. Wd	1	1	N,Z
		XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1	N,Z
		XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1	N,Z
88	ZE	ZE	Ws,Wnd	Wnd = Zero-Extend Ws	1	1	None

23.0 DEVELOPMENT SUPPORT

Microchip offers a comprehensive package of development tools and libraries to support the dsPIC architecture. In addition, the company is partnering with many third party tools manufacturers for additional dsPIC device support. The Microchip tools will include:

- MPLAB® 6.00 Integrated Development Environment (IDE)
- dsPIC 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

23.1 MPLAB V6.00 Integrated Development Environment Software

The MPLAB Integrated Development Environment is available at no cost. The MPLAB IDE gives users the flexibility to edit, compile and emulate, all from a single user interface. Engineers can design and develop code for the dsPIC devices 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, etc.)
- 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 dsPIC 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.

23.2 dsPIC Language Suite

The Microchip Technology MPLAB C30 C compiler is a fully ANSI compliant product with standard libraries for the dsPIC architecture. It is highly optimizing and takes advantage of many dsPIC 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 C30.

23.3 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC-hosted environment, by simulating the dsPIC device 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.

23.4 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 dsPIC devices. Software control of the emulator is provided by MPLAB ICE, 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 50MHz bus speed, or 200MHz external clock speed
- Low-voltage emulation down to 1.8 volts
- Configured with 2Mb program emulation memory, additional modular memory up to 16Mb
- 64K x 136-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

23.5 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™ protocol, offers cost-effective in-circuit debugging from the graphical user interface of MPLAB. 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 in-expensive 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)

23.6 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. Interchangeable socket modules all package types.

In Stand Alone mode, the PRO MATE II device programmer can read, verify, or program PICmicro and dsPIC30F devices. It can also set code protection in this mode. PRO MATE II features include:

- 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℠ 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)

24.0 ELECTRICAL CHARACTERISTICS

Electrical characteristics are not available at this time.

NOTES:

25.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

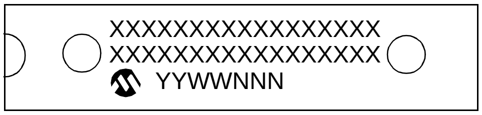
DC and AC Characteristics Graphs and Tables are not available at this time.

NOTES:

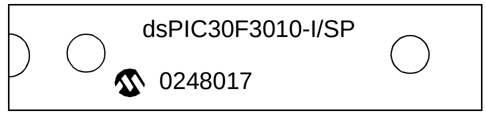
26.0 PACKAGING INFORMATION

26.1 Package Marking Information

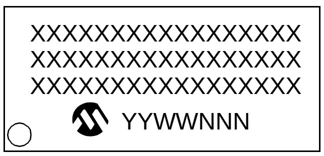
28-Lead PDIP (Skinny DIP)



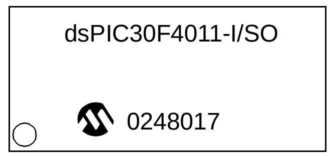
Example



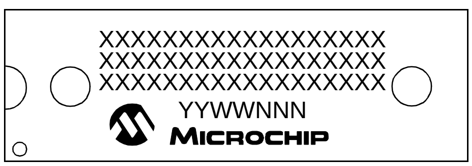
28-Lead SOIC



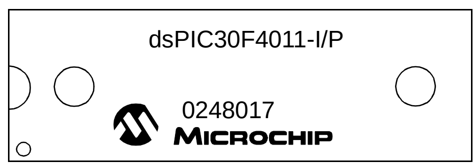
Example



40-Lead PDIP



Example



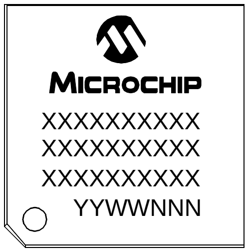
Legend:	XX...X	Customer specific information*
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week '01')
	NNN	Alphanumeric traceability code

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.

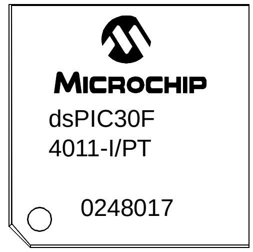
* Standard PICmicro device marking consists of Microchip part number, year code, week code, and traceability code. For PICmicro device marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

26.2 Package Marking Information (Continued)

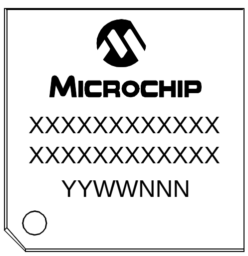
44-Lead TQFP



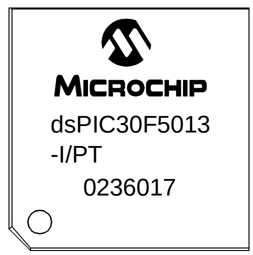
Example



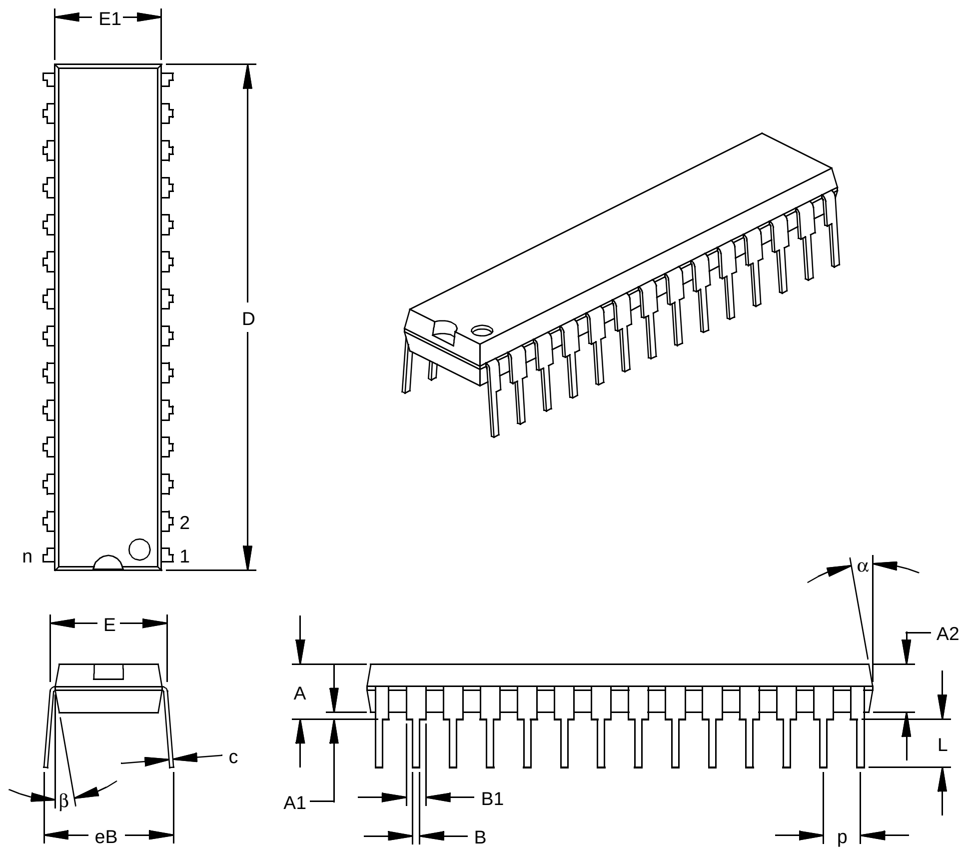
80-Lead TQFP



Example



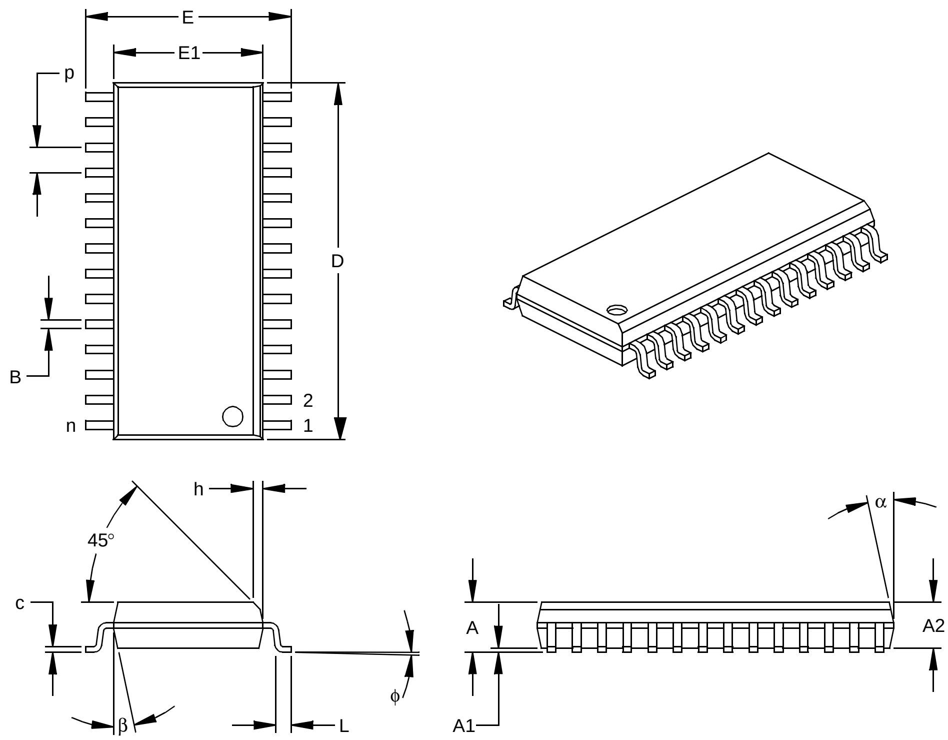
28-Lead Skinny Plastic Dual In-line (SP) – 300 mil (PDIP)



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)



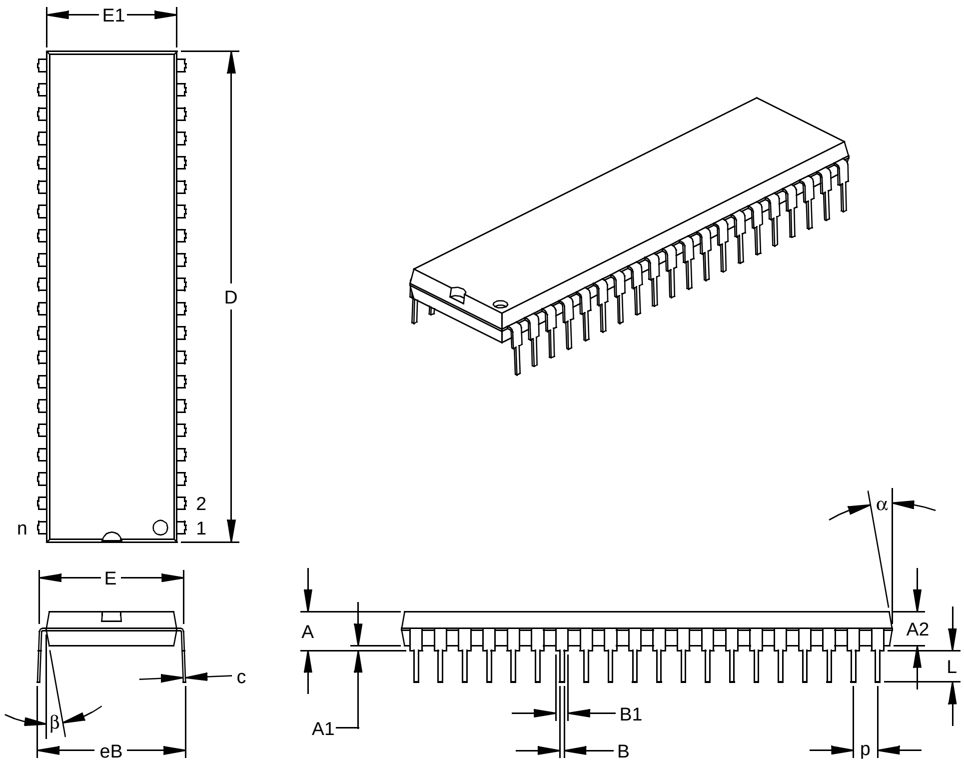
Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.050			1.27	
Overall Height	A	.093	.099	.104	2.36	2.50	2.64
Molded Package Thickness	A2	.088	.091	.094	2.24	2.31	2.39
Standoff §	A1	.004	.008	.012	0.10	0.20	0.30
Overall Width	E	.394	.407	.420	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

 Advance Information

40-Lead Plastic Dual In-line (P) – 600 mil (PDIP)

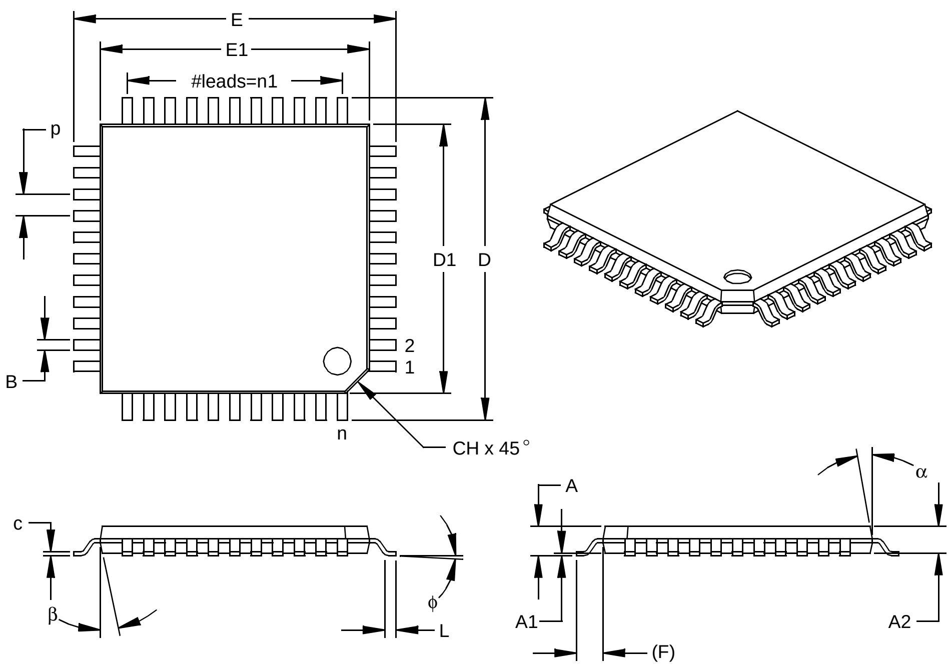


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		40			40	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.160	.175	.190	4.06	4.45	4.83
Molded Package Thickness	A2	.140	.150	.160	3.56	3.81	4.06
Base to Seating Plane	A1	.015			0.38		
Shoulder to Shoulder Width	E	.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)

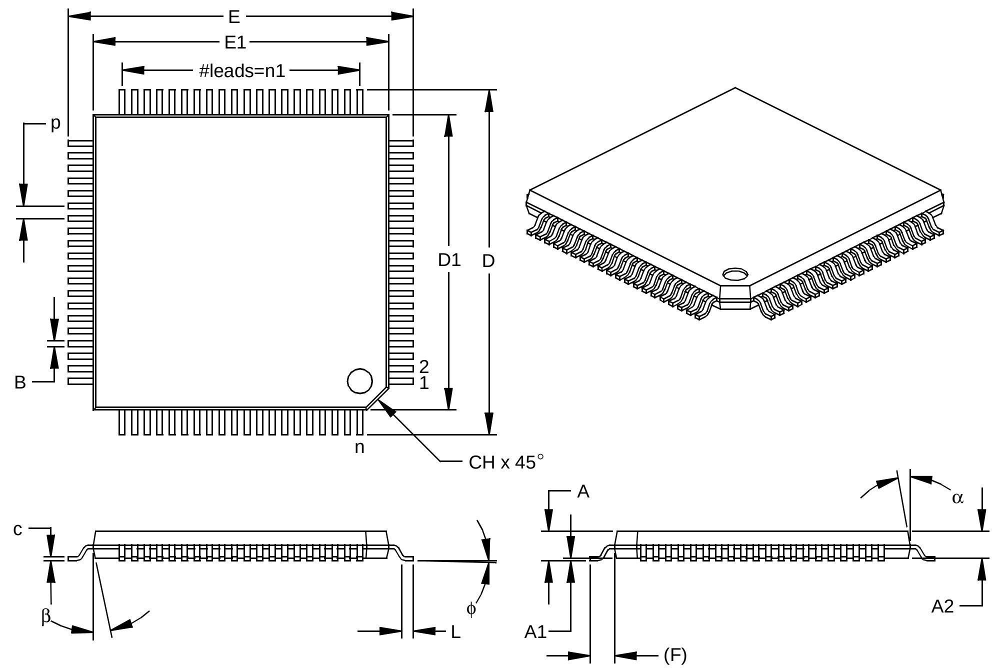


Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		44			44	
Pitch	p		.031			0.80	
Pins per Side	n1		11			11	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.004	.006	0.05	0.10	0.15
Foot 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

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

NOTES:

INDEX

A

Address Generator Units 37
Alternate Vector Table 49
Alternate 16-bit Timer/Counter 91
Automatic Clock Stretch 114
- During 10-bit Addressing (STREN = 1) 114
- During 7-bit Addressing (STREN = 1) 114
- Receive Mode 114
- Transmit Mode 114

A/D 133

B

Barrel Shifter 23
Bit-Reversed Addressing 43
- Example 43
- Implementation 43
- Modifier Values (table) 44
- Sequence Table (16-Entry) 44

Block Diagrams
- CAN Buffers and Protocol Engine 128
- Dedicated Port Structure 61
- DSP Engine 20
- dsPIC30F6010 10
- External Power-on Reset Circuit 148
- Input Capture Mode 81
- I^{2}C 112
- Oscillator System 142
- Output Compare Mode 85
- Quadrature Encoder Interface 89
- RESET System 146
- Shared Port Structure 63
- SPI 108
- SPI Master/Slave Connection 108
- UART Receiver 120
- UART Transmitter 119
- 10-bit High Speed A/D Functional 134
- 16-bit Timer1 Module 67
- 16-bit Timer4 78
- 16-bit Timer5 78
- 32-bit Timer4/5 77
- 8-Output PWM Module 96

BOR. *See* Brown-out Reset
Brown-out Reset (BOR) 141

C

CAN Module 127
- CAN1 Register Map 129
- CAN2 Register Map 131
- Overview 127
- Overview of the Module 127
- Transmit/Receive Buffers 127

Center Aligned PWM 100
Code Examples
- Data EEPROM Block Erase 58
- Data EEPROM Block Write 60
- Data EEPROM Read 57
- Data EEPROM Word Erase 58
- Data EEPROM Word Write 59
- Erasing a Row of Program Memory 53
- Initiating a Programming Sequence 55
- Loading Write Latches 54
- Program Space Data Read Through Data
  - Space Within a REPEAT Loop 28

Code Protection 141
Complementary PWM Operation 100
Configuring Analog Port Pins 63
Control Registers 52
- NVMADR 52
- NVMADRU 52
- NVMCON 52
- NVMKEY 52

Core Architecture
- Overview 13

Core Register Map 34

D

Data Accumulators and Adder/Subtractor 22, 23
Data Address Space 29
- Access RAM 31
- Alignment 30
- Alignment (Figure) 30
- Effect of Invalid Memory Accesses 30
- MCU and DSP (MAC Class) Instructions
  - Example 33
- Memory Map 30
- Memory Map Example 32
- Spaces 29
- Width 30

Data EEPROM Memory 57
- Erasing 58
- Erasing, Block 58
- Erasing, Word 58
- Protection Against Spurious Write 60
- Reading 57
- Write Verify 60
- Writing 59
- Writing, Block 60
- Writing, Word 59

Data Space Organization 37
DC and AC Characteristics Graphs and Tables 165
Dead-Time Generators 101
- Assignment 101
- Ranges 101
- Selection Bits 101

Device Configuration
- Register Map 152

Device Configuration Registers 151
- FBORPOR 151
- FGS 151
- FOSC 151
- FWDT 151

Device Overview 9
Divide Support 18
DSP Engine 19
- Multiplier 21

Dual Output Compare Match Mode 86
- Continuous Pulse Mode 86
- Single Pulse Mode 86

E

Edge Aligned PWM 99
Electrical Characteristics 163
Equations
  A/D Conversion Clock 136
  A/D Sampling Time 137
  Baud Rate 123
  PWM Period 99
  PWM Resolution 99
  Serial Clock Rate 115
Errata 7
Exception Processing 45
  Interrupt Priority 45
  Natural Order Priority (table) 46
Exception Sequence
  Trap Sources 47
External Interrupt Requests 49

F

Fast Context Saving 49
Firmware Instructions 153
FLASH Program Memory 51
  In-Circuit Serial Programming (ICSP) 51
  Run Time Self-Programming (RTSP) 51
  Table Instruction Operation Summary 51

I

In-Circuit Serial Programming (ICSP) 141
Independent PWM Output 102
Initialization Condition for RCON Register,
  Case 1 149
Initialization Condition for RCON Register,
  Case 2 149
Input Capture Interrupts 82
  Register Map 83
Input Capture Module 81
  In CPU SLEEP Mode 82
  Simple Capture Event Mode 81
Input Change Notification Module 66
  Register Map (bit 15-8) 66
  Register Map (bits 15-8) 66
  Register Map (bits 7-0) 66
Instruction Addressing Modes 37
  File Register Instructions 38
  Fundamental Modes Supported 37
  MAC Instructions 38
  MCU Instructions 38
  Move and Accumulator Instructions 38
  Other Instructions 38
Instruction Flow 16
  Pipeline - 1-Word, 1-Cycle (Figure) 16
  Pipeline - 1-Word, 2-Cycle MOV.D
    Operations (Figure) 16
  Pipeline - 1-Word, 2-Cycle Table
    Operations (Figure) 17
  Pipeline - 1-Word, 2-Cycle with Instruction
    Stall (Figure) 18
  Pipeline - 1-Word, 2-Cycle (Figure) 16
  Pipeline - 2-Word, 2-Cycle
    DO, DOW (Figure) 17
  Pipeline - 2-Word, 2-Cycle GOTO,
    CALL (Figure) 17
Instruction Set 153
Instruction Set Overview 156
Instruction Stalls 39
  Introduction 39
  Raw Dependency Detection 39
Inter-Integrated Circuit. *See* I^2C
Interrupt Controller
  Register Map 50
Interrupt Priority
  Traps 47
Interrupt Sequence 48
  Interrupt Stack Frame 48
I^2C 111
I^2C Master Mode
  Baud Rate Generator 115
  Clock Arbitration 116
  Multi-Master Communication, Bus Collision,
    and Bus Arbitration 116
  Reception 115
  Transmission 115
I^2C Module 111
  Addresses 113
  General Call Address Support 115
  Interrupts 114
  IPMI Support 115
  Master Operation 115
  Master Support 115
  Operating Function Description 111
  Operation During CPU SLEEP and
    IDLE Modes 116
  Pin Configuration 111
  Register Map 117
  Registers 111
  Slope Control 115
  Software Controlled Clock Stretching
    (STREN = 1) 114
  Various Modes 111
I^2C Module
  Programmer's Model 111
I^2C 10-bit Slave Mode Operation 113
  Reception 113
  Transmission 113
I^2C 7-bit Slave Mode Operation 113
I^2C 7-bit Slave Mode Operation
  Reception 113
  Transmission 113
I/O Ports 61
  Parallel I/O (PIO) 61

M

Memory Organization 25
Modulo Addressing 40
  Applicability 42
  Decrementing Buffer Operation Example 42
  Incrementing Buffer Operation Example 41
  Restrictions 43
  Start and End Address 40
  W Address Register Selection 40
Motor Control PWM Module 95

 Advance Information

O

Oscillator Configurations 143
Fail-Safe Clock Monitor 145
Fast RC (FRC) 144
Initial Clock Source Selection 143
Low Power RC (LPRC) 144
LP Oscillator Control 144
Phase Locked Loop (PLL) 144
Start-up Timer (OST) 144
Oscillator Operating Modes Table 141
Oscillator Selection 141
Output Compare Interrupts 87
Output Compare Mode
Register Map 88
Output Compare Module 85
Output Compare Operation During
CPU IDLE Mode 87
Output Compare SLEEP Mode Operation 87

P

Packaging Information 167
Marking 167
Peripheral Module Display (PMD) Registers 151
Pinout Descriptions 11
PORTA
Register Map 62
PORTB
Register Map 64
PORTC
Register Map 64
PORTD
Register Map 64
PORTE
Register Map 64
PORTF
Register Map 65
PORTG
Register Map 65
POR. *See* Power-on Reset
Position Measurement Mode 90
Power Saving Modes 150
IDLE 151
SLEEP 150
Power Saving Modes (SLEEP and IDLE) 141
Power-on Reset (POR) 141
Oscillator Start-up Timer (OST) 141
Power-up Timer (PWRT) 141
Product Identification System 183
Program Address Space 25
Alignment and Data Access Using
Table Instructions 26
Construction 25
Data Access from, Address Generation 25
Memory Map 29
Table Instructions
TBLRDH 26
TBLRDL 26
TBLWTH 26
TBLWTL 26
Program Counter 14
Program Data Table Access 27
Program Space Visibility
Window into Program Space Operation 28
Program Space Visibility from Data Space 27
Data Pre-fetch from Program Space Within a
REPEAT Loop 27
Programmable 141
Programmable Digital Noise Filters 91
Programmer's Model 14
Diagram 15
Programming Operations 52
Algorithm for Program FLASH 53
Erasing a Row of Program Memory 53
Initiating the Programming Sequence 55
Loading Write Latches 54
Programming, Device Instructions 153
Protection Against Accidental Writes
to OSCCON 145
PWM Duty Cycle Comparison Units 100
Duty Cycle Register Buffers 100
PWM FAULT Pins 103
Enable Bits 103
FAULT States 103
Modes 103
Cycle-by-Cycle 103
Latched 103
Priority 103
PWM Operation During CPU IDLE Mode 104
PWM Operation During CPU SLEEP Mode 104
PWM Output and Polarity Control 103
Output Pin Control 103
PWM Output Override 102
Complementary Output Mode 102
Synchronization 102
PWM Period 99
PWM Special Event Trigger 104
Postscaler 104
PWM Time-Base 98
Continuous Up/Down Counting Modes 98
Double Update Mode 99
Free Running Mode 98
Postscaler 99
Prescaler 99
Single Shot Mode 98
PWM Update Lockout 104

Q

QEI Module
Operation During CPU IDLE Mode 91
Operation During CPU SLEEP Mode 91
Register Map 93
Timer Operation During CPU IDLE Mode 92
Timer Operation During CPU SLEEP Mode 91
Quadrature Encoder Interface Interrupts 92
Quadrature Encoder Interface Logic 90
Quadrature Encoder Interface (QEI) Module 89

R

RESET 141, 146
RESET Sequence 47
RESET Sources 47
RESETS
BOR, Programmable 148
POR 146
Operating without FSCM and PWRT 148
POR with Long Crystal Start-up Time 148
RTSP Operation 52

S

Sales and Support 183
Serial Peripheral Interface. *See* SPI
Simple Capture Event Mode
  Capture Buffer Operation 82
  Capture Prescaler 81
  Hall Sensor Mode 82
  Input Capture in CPU IDLE Mode 82
  Timer2 and Timer3 Selection Mode 82
Simple Output Compare Match Mode 86
Simple PWM Mode 86
  Input Pin FAULT Protection 86
  Period 87
Single Pulse PWM Operation 102
Software Stack Pointer, Frame Pointer 14
  CALL Stack Frame 31
SPI 107
SPI Mode
  Slave Select Synchronization 109
  SPI1 Register Map 110
  SPI2 Register Map 110
SPI Module 107
  Framed SPI Support 107
  Operating Function Description 107
  SDOx Disable 107
  Word and Byte Communication 107
SPI Operation During CPU IDLE Mode 109
SPI Operation During CPU SLEEP Mode 109
STATUS Register 14
  Sticky Z (SZ) Status Bit 14
Subtractor 22
  Data Space Write Saturation 23
  Overflow and Saturation 22
  Round Logic 23
  Write Back 23
Symbols used in Roadrunner
  Opcode Descriptions 154
System Integration 141
  Overview 141
  Register Map 152

T

Timer1 Module 67
  Gate Operation 68
  Interrupt 68
  Operation During SLEEP Mode 68
  Prescaler 68
  Real-Time Clock 68
    RTC Interrupts 68
    RTC Oscillator Operation 68
  Register Map 69
  16-bit Asynchronous Counter Mode 67
  16-bit Synchronous Counter Mode 67
  16-bit Timer Mode 67
Timer2 and Timer 3 Selection Mode 86
Timer2/3 Module 71
  ADC Event Trigger 74
  Gate Operation 74
  Interrupt 74
  Operation During SLEEP Mode 74
  Register Map 75
  Timer Prescaler 74
  32-bit Synchronous Counter Mode 71
  32-bit Timer Mode 71
Timer4/5 Module 77
  Register Map 79
Timing Diagrams
  Center Aligned PWM 100
  Dead-Time 102
  Edge Aligned PWM 99
  PWM Output 87
  Time-out Sequence on Power-up (MCLR Not
    Tied to VDD), Case 1 147
  Time-out Sequence on Power-up (MCLR Not
    Tied to VDD), Case 2 147
  Time-out Sequence on Power-up (MCLR
    Tied to VDD) 147

U

UART
  Address Detect Mode 123
  Auto Baud Support 123
  Baud Rate Generator 123
  Enabling and Setting Up UART 121
    Alternate I/O 121
    Disabling 121
    Enabling 121
    Setting Up Data, Parity and STOP
      Bit Selections 121
  Loopback Mode 123
  Module Overview 119
  Operation During CPU SLEEP and IDLE Modes 124
  Receiving Data 122
    In 8-bit or 9-bit Data Mode 122
    Interrupt 122
    Receive Buffer (UxRCB) 122
  Reception Error Handling 122
    Framing Error (FERR) 123
    IDLE Status 123
    Parity Error (PERR) 123
    Receive Break 123
    Receive Buffer Overrun Error
      (OERR Bit) 122
  Transmitting Data
    In 8-bit Data Mode 121
  Transmitting Data 121
    In 9-bit Data Mode 121
    Interrupt 122
    Transmit Buffer (UxTXB) 121
  UART1 Register Map 125
  UART2 Register Map 125
Unit ID Locations 141
Universal Asynchronous Receiver Transmitter. *See* UART.

W

Wake-up from SLEEP 141
Wake-up from SLEEP and IDLE 49
Watchdog Timer (WDT) 141, 150
  Enabling and Disabling 150
  Operation 150
WWW, On-Line Support 7

 Advance Information

Z

10-bit High Speed Analog-to-Digital Converter. *See* A/D
10-bit High Speed A/D
- Aborting a Conversion 136
- ADCHS 133
- ADCON1 133
- ADCON2 133
- ADCON3 133
- ADCSSL 133
- ADPCFG 133
- Analog Input Model 137
- Configuring Analog Port Pins 139
- Connection Considerations 139
- Conversion Operation 135
- Effects of a RESET 138
- Operation During CPU IDLE Mode 138
- Operation During CPU SLEEP Mode 138
- Output Formats 139
- Power-down Modes 138
- Programming the Start of Conversion Trigger 136
- Register Map 140
- Result Buffer 135
- Sampling Requirements 137
- Selecting the Conversion Clock 136
- Selecting the Conversion Sequence 135
- TAD vs. Device Operating Frequencies Table 136

16-bit Integer and Fractional Modes Example 21
16-bit Up/Down Position Counter Mode 90
- Count Direction Status 90
- Error Checking 90

6-Output PWM
- Register Map 105

6-Output PWM vs. 8-Output PWM
- Feature Summary 95

8-Output PWM
- Register Map 105

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: **DS70082A**

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?

____

____

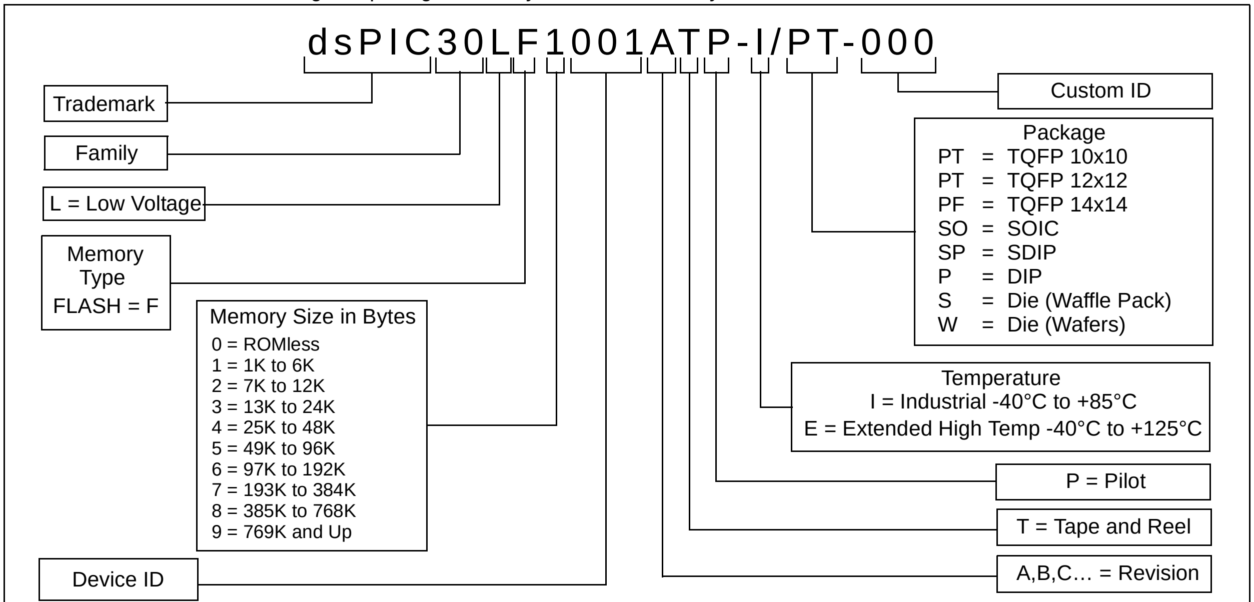
8. How would you improve our software, systems, and silicon products?

____

____

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



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.

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: 480-792-4338

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: 86-28-86766200 Fax: 86-28-86766599

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

China - Hong Kong SAR
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: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Microchip Technology (Barbados) Inc.,
Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Austria
Microchip Technology Austria GmbH
Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

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
Microchip Ltd.
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44 118 921 5869 Fax: 44-118 921-5820

05/16/02