



Welcome to [E-XFL.COM](#)

What is "[Embedded - Microcontrollers](#)"?

"[Embedded - Microcontrollers](#)" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "[Embedded - Microcontrollers](#)"

Details

Product Status	Active
Core Processor	dsPIC
Core Size	16-Bit
Speed	30 MIPS
Connectivity	I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, Motor Control PWM, QEI, POR, PWM, WDT
Number of I/O	30
Program Memory Size	24KB (8K x 24)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	A/D 9x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f3011-30i-ml

dsPIC30F3010/3011

High Performance, 16-Bit Digital Signal Controllers

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the “*dsPIC30F Family Reference Manual*” (DS70046). For more information on the device instruction set and programming, refer to the “*16-bit MCU and DSC Programmer’s Reference Manual*” (DS70157).

High-Performance Modified RISC CPU:

- Modified Harvard Architecture
- C Compiler Optimized Instruction Set Architecture with Flexible Addressing modes
- 83 Base Instructions
- 24-bit Wide Instructions, 16-bit Wide Data Path
- 24 Kbytes On-Chip Flash Program Space (8K instruction words)
- 1 Kbyte of On-Chip Data RAM
- 1 Kbyte of Nonvolatile Data EEPROM
- 16 x 16-bit Working Register Array
- Up to 30 MIPS Operation:
 - DC to 40 MHz external clock input
 - 4 MHz-10 MHz oscillator input with PLL active (4x, 8x, 16x)
- 29 Interrupt Sources
 - 3 external interrupt sources
 - 8 user-selectable priority levels for each interrupt source
 - 4 processor trap sources

DSP Engine Features:

- Dual Data Fetch
- Accumulator Write Back for DSP Operations
- Modulo and Bit-Reversed Addressing modes
- Two, 40-bit Wide Accumulators with Optional saturation Logic
- 17-bit x 17-bit Single-Cycle Hardware Fractional/Integer Multiplier
- All DSP Instructions Single Cycle
- ± 16 -bit Single-Cycle Shift

Peripheral Features:

- High-Current Sink/Source I/O Pins: 25 mA/25 mA
- Timer module with Programmable Prescaler:
 - Five 16-bit timers/counters; optionally pair 16-bit timers into 32-bit timer modules
- 16-bit Capture Input Functions
- 16-bit Compare/PWM Output Functions
- 3-Wire SPI modules (supports 4 Frame modes)
- I²C™ module Supports Multi-Master/Slave mode and 7-bit/10-bit Addressing
- 2 UART modules with FIFO Buffers

Motor Control PWM Module Features:

- 6 PWM Output Channels
 - Complementary or Independent Output modes
 - Edge and Center-Aligned modes
- 3 Duty Cycle Generators
- Dedicated Time Base
- Programmable Output Polarity
- Dead-Time Control for Complementary mode
- Manual Output Control
- Trigger for A/D Conversions

Quadrature Encoder Interface Module Features:

- Phase A, Phase B and Index Pulse Input
- 16-bit Up/Down Position Counter
- Count Direction Status
- Position Measurement (x2 and x4) mode
- Programmable Digital Noise Filters on Inputs
- Alternate 16-bit Timer/Counter mode
- Interrupt on Position Counter Rollover/Underflow

Analog Features:

- 10-bit Analog-to-Digital Converter (ADC) with 4 Sample and Hold (S&H) Inputs:
 - 1 Msps conversion rate
 - 9 input channels
 - Conversion available during Sleep and Idle
- Programmable Brown-out Reset

Table of Contents

1.0	Device Overview	11
2.0	CPU Architecture Overview.....	19
3.0	Memory Organization	27
4.0	Address Generator Units.....	39
5.0	Interrupts	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	91
15.0	Motor Control PWM Module.....	97
16.0	SPI Module.....	107
17.0	I2C™ Module	111
18.0	Universal Asynchronous Receiver Transmitter (UART) Module	119
19.0	10-bit High-Speed Analog-to-Digital Converter (ADC) Module	127
20.0	System Integration	139
21.0	Instruction Set Summary	155
22.0	Development Support.....	165
23.0	Electrical Characteristics	169
24.0	Packaging Information.....	209
	Index	219
	The Microchip Web Site.....	225
	Customer Change Notification Service	225
	Customer Support.....	225
	Reader Response	226
	Product Identification System	227

TO OUR VALUED CUSTOMERS

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

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

Most Current Data Sheet

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

<http://www.microchip.com>

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

Errata

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

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

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

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

Customer Notification System

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

dsPIC30F3010/3011

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

TABLE 1-2: dsPIC30F3010 I/O PIN DESCRIPTIONS

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

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

2.0 CPU ARCHITECTURE OVERVIEW

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the “dsPIC30F Family Reference Manual” (DS70046). For more information on the device instruction set and programming, refer to the “16-bit MCU and DSC Programmer’s Reference Manual” (DS70157).

2.1 Core Overview

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

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

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

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

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

- 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 “Address Generator Units”** for details on Modulo and Bit-Reversed addressing.

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

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

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

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

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

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, 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 saturation logic loads the maximally positive 9.31 (0x7FFFFFFF) or maximally negative 9.31 value (0x80000000) into the target accumulator. The SA or SB bit is set and remains set until cleared by the user. This is referred to as 'super saturation' and provides protection against erroneous data or unexpected algorithm problems (e.g., gain calculations).
2. Bit 31 Overflow and Saturation:
When bit 31 overflow and saturation occurs, the saturation logic then loads the maximally positive 1.31 value (0x007FFFFFFF) or maximally negative 1.31 value (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.

2.4.2.2 Accumulator 'Write Back'

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

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

2.4.2.3 Round Logic

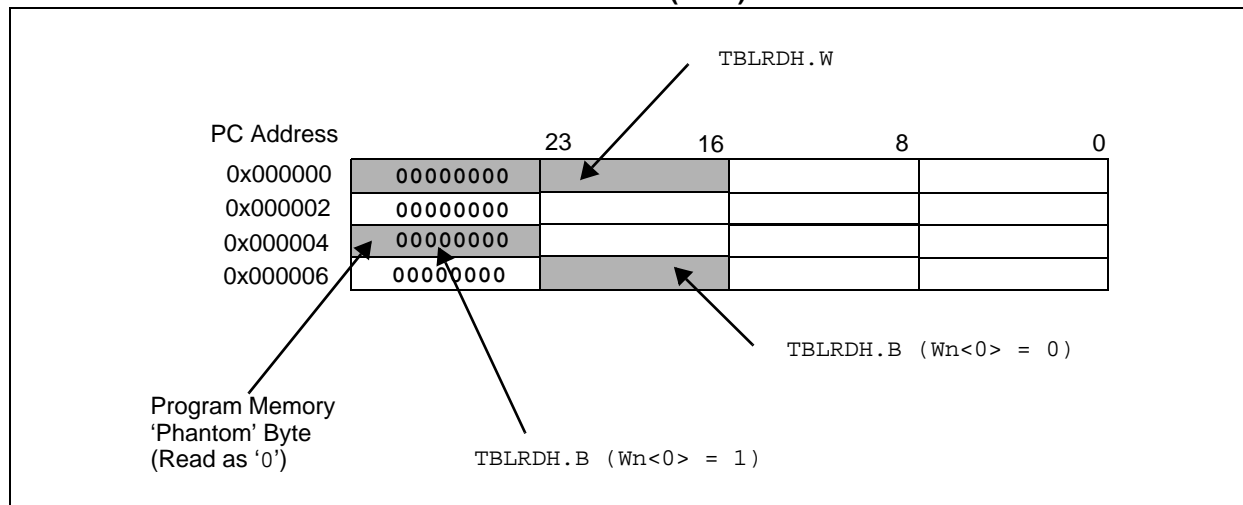
The round logic is a combinational block, which performs a conventional (biased) or convergent (unbiased) round function during an accumulator write (store). The Round mode is determined by the state of the RND bit in the CORCON register. It generates a 16-bit, 1.15 data value which is passed to the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

Conventional rounding takes bit 15 of the accumulator, zero-extends it and adds it to the ACCxH word (bits 16 through 31 of the accumulator). If the ACCxL word (bits 0 through 15 of the accumulator) is between 0x8000 and 0xFFFF (0x8000 included), ACCxH is incremented. If ACCxL is between 0x0000 and 0x7FFF, ACCxH is left unchanged. A consequence of this algorithm is that over a succession of random rounding operations, the value 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 LSb (bit 16 of the accumulator) of ACCxH is examined. If it is '1', ACCxH is incremented. If it is '0', ACCxH is not modified. Assuming that bit 16 is effectively random in nature, this scheme 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 2.4.2.4 "Data Space Write Saturation"**). 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 through the X bus. For this class of instructions, the data is always subject to rounding.

FIGURE 3-4: PROGRAM DATA TABLE ACCESS (MSB)



3.1.2 DATA ACCESS FROM PROGRAM MEMORY USING PROGRAM SPACE VISIBILITY

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

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

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

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

Although each data space address, 0x8000 and higher, maps directly into a corresponding program memory address, as shown in Figure 3-5, only the lower 16 bits of the 24-bit program word are used to contain the data. The upper 8 bits should be programmed to force an illegal instruction to maintain machine robustness. Refer to the “16-bit MCU and DSC Programmer’s Reference Manual” (DS70157) for details on instruction encoding.

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

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

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

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

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

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

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

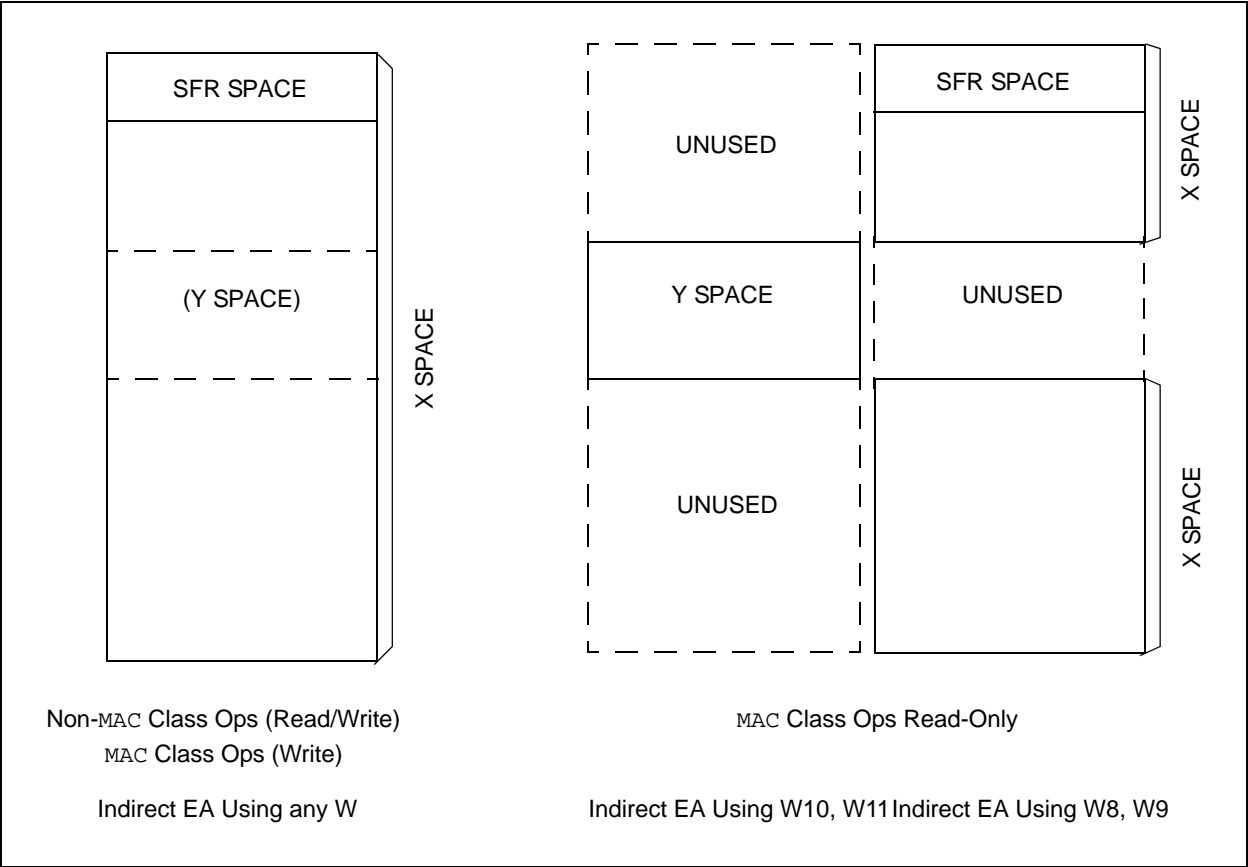


TABLE 3-3: CORE REGISTER MAP⁽¹⁾ (CONTINUED)

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
CORCON	0044	—	—	—	US	EDT	DL2	DL1	DL0	SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	IF	0000 0000 0010 0000
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; — = unimplemented bit, read as '0'

Note 1: Refer to the “dsPIC30F Family Reference Manual” (DS70046) for descriptions of register bit fields.

5.2 Reset Sequence

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

5.2.1 RESET SOURCES

There are 6 sources of error which will cause a device reset.

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

5.3 Traps

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

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

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

There are 8 fixed priority levels for traps: Level 8 through Level 15, which 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.

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 four circumstances:

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

6.4 RTSP Operation

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

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

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

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

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

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

6.5 RTSP Control Registers

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

- `NVMCON`
- `NVMADR`
- `NVMADRU`
- `NVMKEY`

6.5.1 NVMCON REGISTER

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

6.5.2 NVMADR REGISTER

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

6.5.3 NVMADRU REGISTER

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

6.5.4 NVMKEY REGISTER

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

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

7.0 DATA EEPROM MEMORY

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the “dsPIC30F Family Reference Manual” (DS70046). For more information on the device instruction set and programming, refer to the “16-bit MCU and DSC Programmer’s Reference Manual” (DS70157).

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

The four SFRs used to read and write the program Flash memory are used to access data EEPROM memory as well. As described in **Section 4.0 “Address Generator Units”**, these registers are:

- NVMCON
- NVMADR
- NVMADRU
- NVMKEY

The EEPROM data memory allows read and write of single words and 16-word blocks. When interfacing to data memory, NVMADR, in conjunction with the NVMADRU register, is used to address the EEPROM location being accessed. TBLRD and TBLWT instructions are used to read and write data EEPROM. The dsPIC30F3010/3011 devices have 1 Kbyte (512 words) of data EEPROM, with an address range from 0x7FFC00 to 0x7FFFE.

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

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

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

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

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

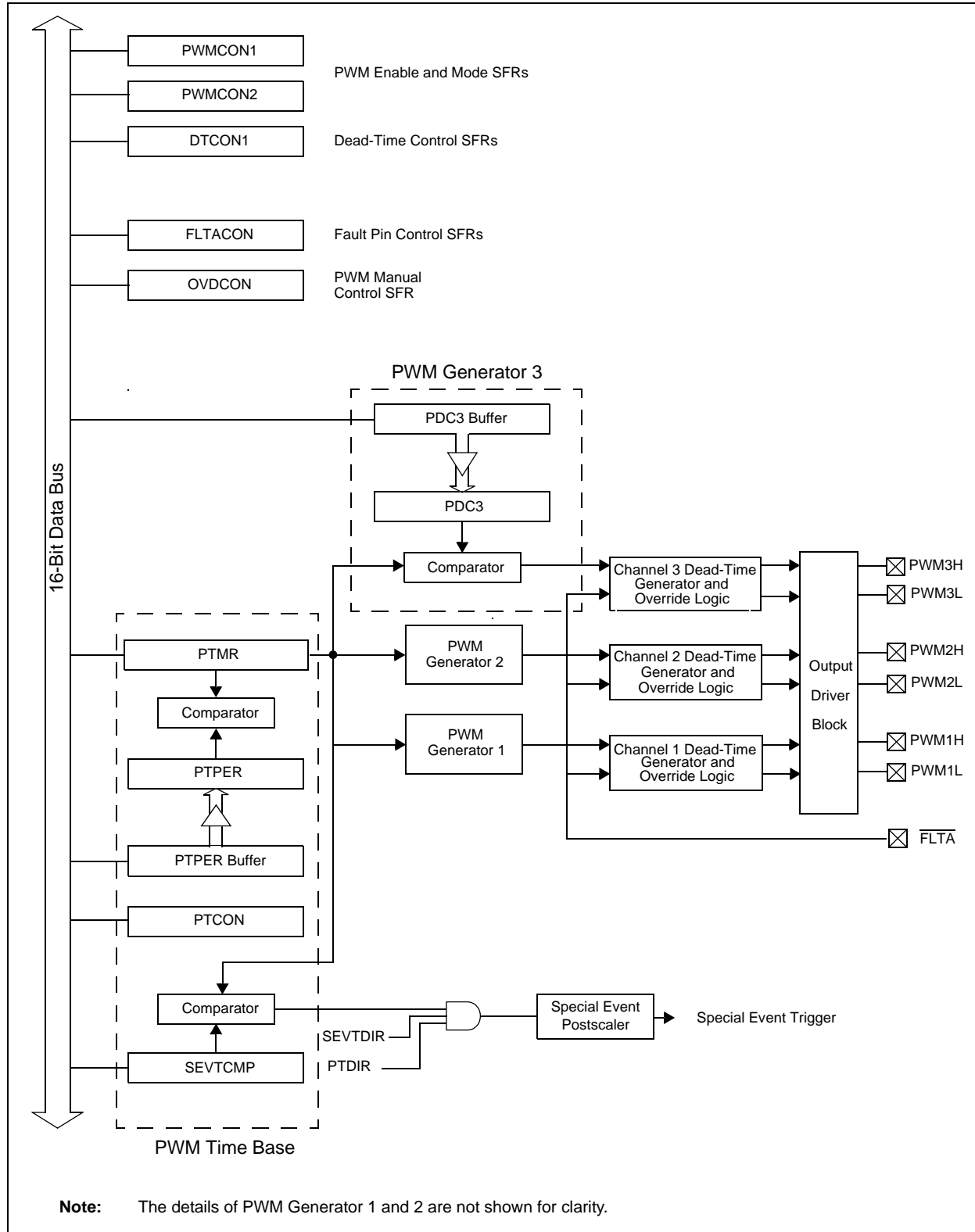
7.1 Reading the Data EEPROM

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

EXAMPLE 7-1: DATA EEPROM READ

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

FIGURE 15-1: PWM MODULE BLOCK DIAGRAM



15.1 PWM Time Base

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

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

Note: If the Period register is set to 0x0000, the timer will stop counting, and the interrupt and 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 PTCN SFR. The Continuous Up/Down Count modes support center-aligned PWM generation. The Single-Shot mode allows the PWM module to support pulse control of certain Electronically Commutative Motors (ECMs).

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

15.1.1 FREE-RUNNING MODE

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

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

15.1.2 SINGLE-SHOT MODE

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

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

15.1.3 CONTINUOUS UP/DOWN COUNT MODES

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

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

16.2 Framed SPI Support

The module supports a basic framed SPI protocol in Master or Slave mode. The control bit, `FRMEN`, enables framed SPI support and causes the `SS1` pin to perform the Frame Synchronization (FSYNC) pulse function. The control bit, `SPIFSD`, determines whether

the `SS1` pin is an input or an output (i.e., whether the module receives or generates the frame synchronization pulse). The frame pulse is an active-high pulse for a single SPI clock cycle. When frame synchronization is enabled, the data transmission starts only on the subsequent transmit edge of the SPI clock.

FIGURE 16-1: SPI BLOCK DIAGRAM

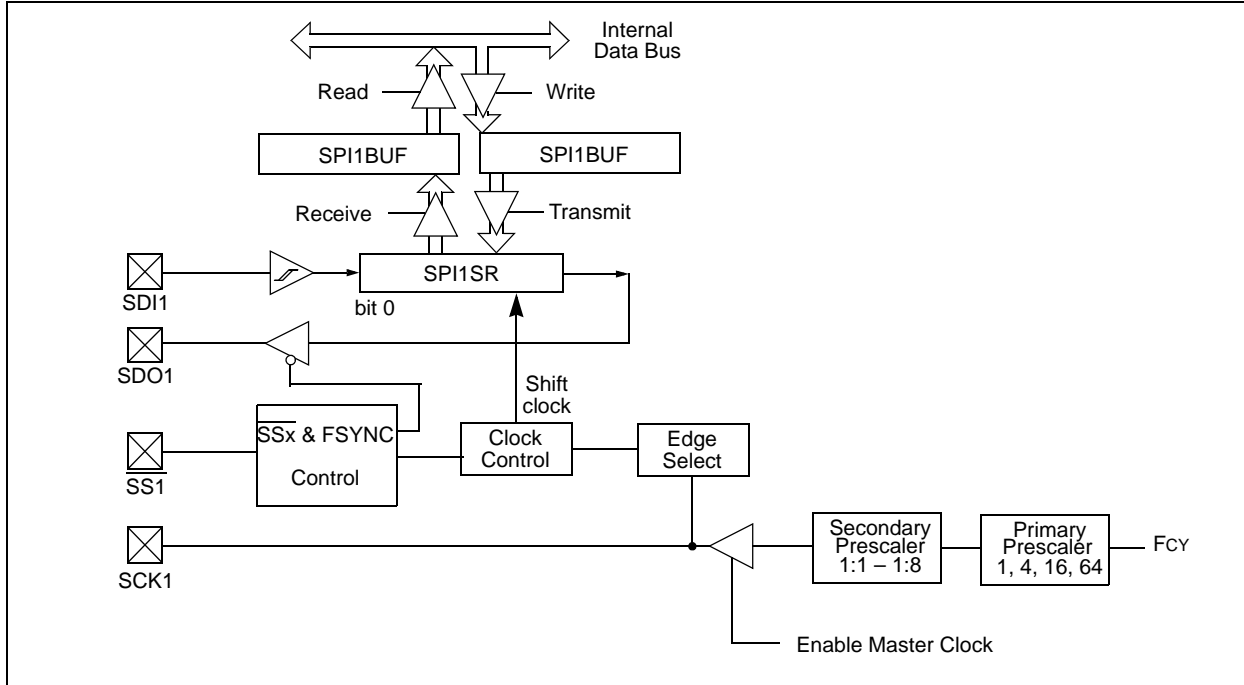
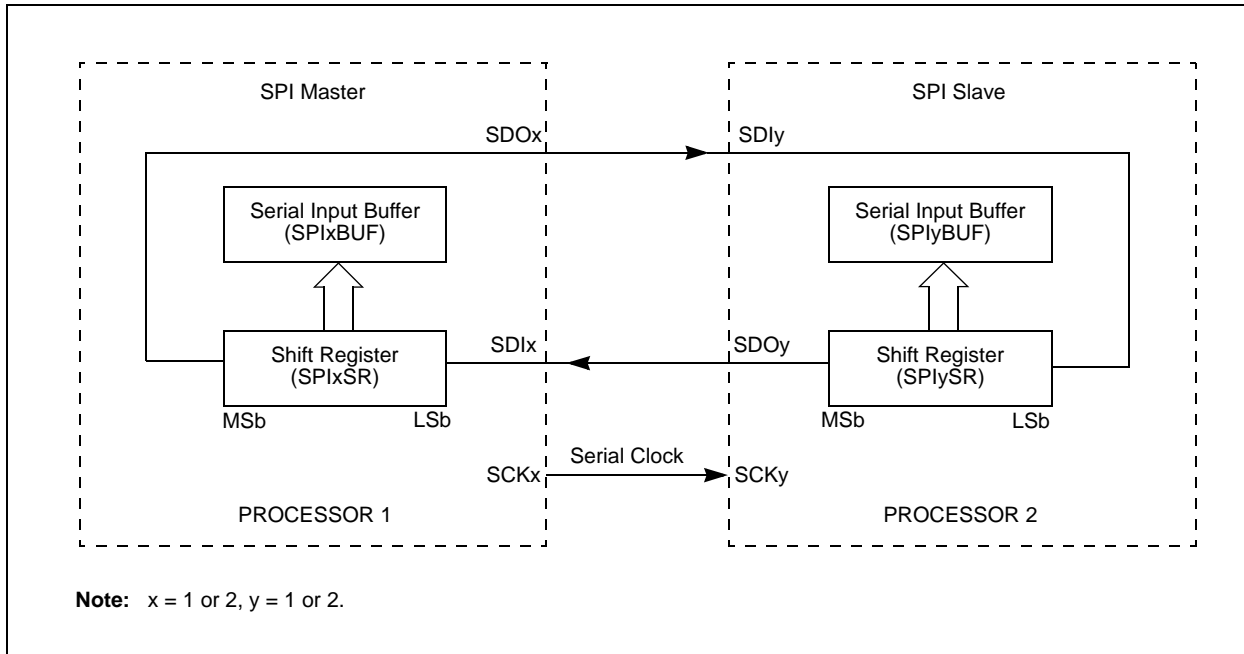


FIGURE 16-2: SPI MASTER/S�AVE CONNECTION



18.2 Enabling and Setting Up UART

18.2.1 ENABLING THE UART

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

18.2.2 DISABLING THE UART

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

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

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

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

18.2.3 ALTERNATE I/O

The alternate I/O function is enabled by setting the ALTIO bit (U1MODE<10>). If ALTIO = 1, the UxATX and UxARX pins (alternate transmit and alternate receive pins, respectively) are used by the UART module instead of the UxTX and UxRX pins. If ALTIO = 0, the UxTX and UxRX pins are used by the UART module.

18.2.4 SETTING UP DATA, PARITY AND STOP BIT SELECTIONS

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

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

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

18.3 Transmitting Data

18.3.1 TRANSMITTING IN 8-BIT DATA MODE

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

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

18.3.2 TRANSMITTING IN 9-BIT DATA MODE

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

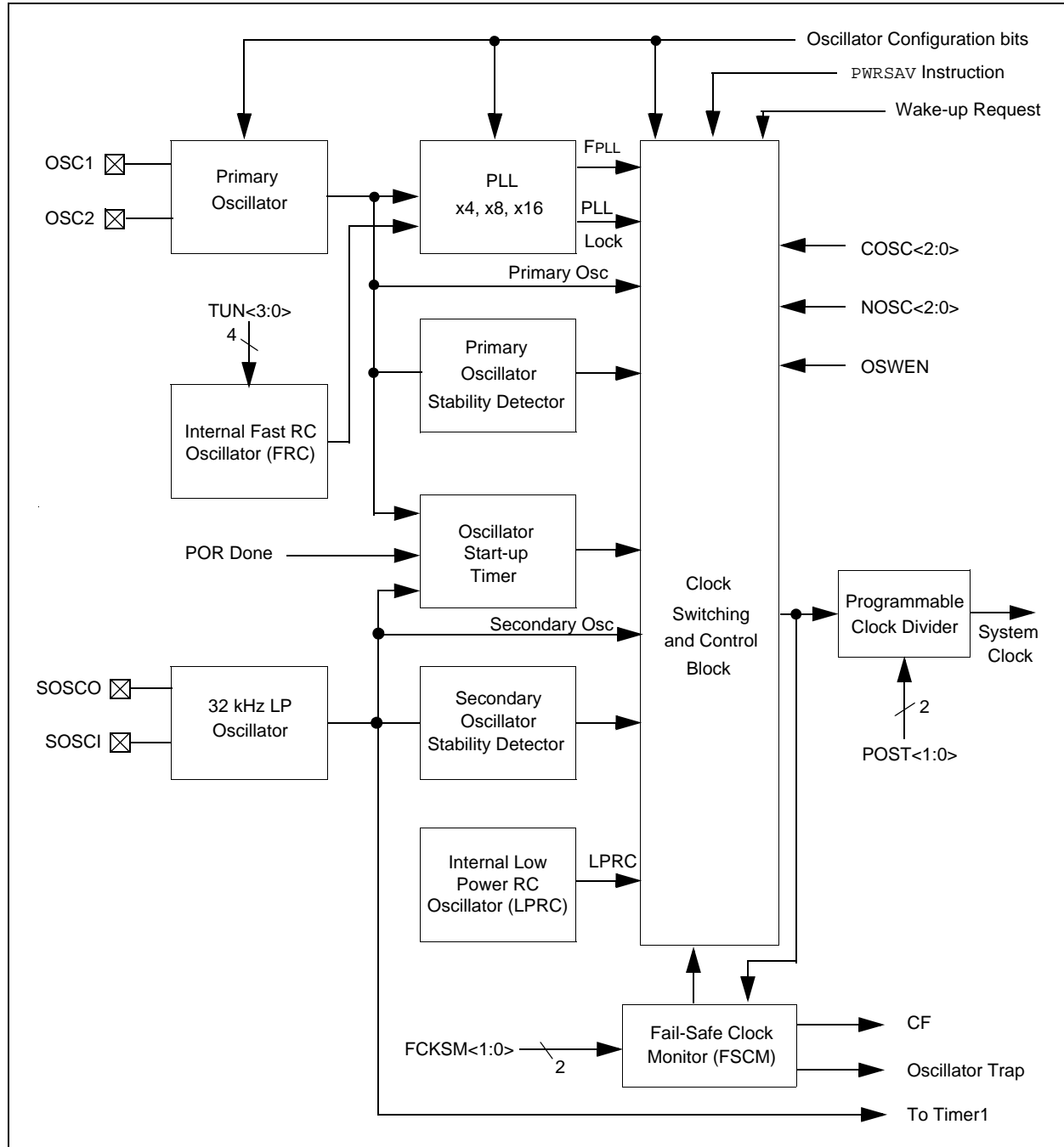
18.3.3 TRANSMIT BUFFER (UxTXB)

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

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

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

FIGURE 20-1: OSCILLATOR SYSTEM BLOCK DIAGRAM



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 (TPOR, TLOCK and TPWRT), the crystal oscillator (and PLL) may not be active at the end of the time-out (e.g., for low-frequency crystals). In such cases, if FSCM is enabled, then the device will detect this as a clock failure and process the clock failure trap, the FRC oscillator will be enabled, and the user will have to re-enable the crystal oscillator. If FSCM is not enabled, then the device will simply suspend execution of code until the clock is stable, and will remain in Sleep until the oscillator clock has started.

All Resets will wake-up the processor from Sleep mode. Any Reset, other than POR, will set the SLEEP status bit. In a POR, the SLEEP bit is cleared.

If the Watchdog Timer is enabled, then the processor will wake-up from Sleep mode upon WDT time-out. The SLEEP and WDTO status bits are both set.

20.5.2 IDLE MODE

In Idle mode, the clock to the CPU is shut down while peripherals keep running. Unlike Sleep mode, the clock source remains active.

Several peripherals have a control bit in each module that allows them to operate during Idle.

The 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 (i.e., bit is '1') and meets the required priority level
- On any Reset (POR, BOR, $\overline{\text{MCLR}}$)
- On a 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 the 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 the 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 the Watchdog Timer is enabled, then the processor will wake-up from Idle mode upon WDT time-out. The IDLE and WDTO status bits are both set.

Unlike wake-up from Sleep, there are no time delays involved in wake-up from Idle.

20.6 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 five 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
5. FICD (0xF8000C): Debug Configuration Register

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

Note: If the code protection Configuration bits (FGS<GCP> and FGS<GWRP>) have been programmed, an erase of the entire code-protected device is only possible at voltages $V_{DD} \geq 4.5V$.

21.0 INSTRUCTION SET SUMMARY

Note: This data sheet summarizes features of this group of dsPIC30F devices and is not intended to be a complete reference source. For more information on the CPU, peripherals, register descriptions and general device functionality, refer to the “dsPIC30F Family Reference Manual” (DS70046). For more information on the device instruction set and programming, refer to the “16-bit MCU and DSC Programmer’s Reference Manual” (DS70157).

The dsPIC30F instruction set adds many enhancements to the previous PIC® microcontroller (MCU) instruction sets, while maintaining an easy migration from PIC MCU instruction sets.

Most instructions are a single program memory word (24 bits). Only three instructions require two program memory locations.

Each single-word instruction is a 24-bit word divided into an 8-bit opcode which specifies the instruction type, and one or more operands which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into five basic categories:

- Word or byte-oriented operations
- Bit-oriented operations
- Literal operations
- DSP operations
- Control operations

Table 21-1 shows the general symbols used in describing the instructions.

The dsPIC30F instruction set summary in Table 21-2 lists all the instructions along with the status flags affected by each instruction.

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

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

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

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

Most bit-oriented instructions (including simple rotate/shift instructions) have two operands:

- The W register (with or without an address modifier) or File register (specified by the value of ‘Ws’ or ‘f’)
- The bit in the W register or File register (specified by a literal value, or indirectly by the contents of register ‘Wb’)

The literal instructions that involve data movement may use some of the following operands:

- A literal value to be loaded into a W register or File register (specified by the value of ‘k’)
- The W register or File register where the literal value is to be loaded (specified by ‘Wb’ or ‘f’)

However, literal instructions that involve arithmetic or logical operations use some of the following operands:

- The first source operand, which is a register ‘Wb’ without any address modifier
- The second source operand, which is a literal value
- The destination of the result (only if not the same as the first source operand), which is typically a register ‘Wd’ with or without an address modifier

The MAC class of DSP instructions may use some of the following operands:

- The accumulator (A or B) to be used (required operand)
- The W registers to be used as the two operands
- The X and Y address space prefetch operations
- The X and Y address space prefetch destinations
- The accumulator write-back destination

The other DSP instructions do not involve any multiplication, and may include:

- The accumulator to be used (required)
- The source or destination operand (designated as Wso or Wdo, respectively) with or without an address modifier
- The amount of shift, specified by a W register ‘Wn’ or a literal value

The control instructions may use some of the following operands:

- A program memory address
- The mode of the table read and table write instructions

All instructions are a single word, except for certain double-word instructions, which were made double-word instructions so that all the required information is available in these 48 bits. In the second word, the 8 MSBs are ‘0’s. If this second word is executed as an instruction (by itself), it will execute as a NOP.

dsPIC30F3010/3011

Revision F (November 2010)

This revision includes minor typographical and formatting changes throughout the data sheet text.

The major changes are referenced by their respective section in Table A-1.

TABLE A-1: MAJOR SECTION UPDATES

Section Name	Update Description
“High Performance, 16-Bit Digital Signal Controllers”	Added Note 1 to all QFN pin diagrams (see “Pin Diagrams”).
Section 1.0 “Device Overview”	Updated the Pinout I/O Descriptions for AVDD and AVSS (see Table 1-1 and Table 1-2).
Section 15.0 “Motor Control PWM Module”	Added the IUE bit (PWMCON2<2>) to the PWM Register Map (see Table 15-1). Updated the PWM Period equations (see Equation 15-1 and Equation 15-2).
Section 20.0 “System Integration”	Added a shaded note on OSCTUN functionality in Section 20.2.5 “Fast RC Oscillator (FRC)” . Updated Notes 1 and 2 in the Configuration Bit Values for Clock Selection (see Table 20-2).
Section 24.0 “Packaging Information”	Removed the 44-Lead QFN package definitions.
Section 23.0 “Electrical Characteristics”	Updated the typical and maximum values for parameter DC16 in the DC Temperature and Voltage Specifications (see Table 23-4). Updated the maximum value for parameter DI19 and the minimum value for parameter DI29 in the I/O Pin Input Specifications (see Table 23-8). Removed parameter D136 and updated the minimum, typical, maximum, and conditions for parameters D122 and D134 in the Program and EEPROM specifications (see Table 23-11). Updated Note 1 in the Internal FRC Accuracy specifications (see Table 23-17).
“Product Identification System”	Updated the “ML” package definition.