

dsPIC30F Programmer's Reference Manual

High Performance Digital Signal Controllers

© 2003 Microchip Technology Inc.

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

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

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

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contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates.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

Trademarks

The Microchip name and logo, the Microchip logo, KEELOQ, MPLAB, PIC, PICmicro, PICSTART, PRO MATE and PowerSmart are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

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

Accuron, Application Maestro, dsPIC, dsPICDEM, dsPICDEM.net, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, PICC, PICkit, PICDEM, PICDEM.net, PowerCal, PowerInfo, PowerMate, PowerTool, rfLAB, rfPIC, Select Mode, SmartSensor, SmartShunt, SmartTel and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Serialized Quick Turn Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

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

© 2003, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.





Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999 and Mountain View, California in March 2002. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUS, KEELoQ® code hopping devices, Serial EEPROMs, microperipherals, non-volatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.



Table of Contents

	PAGE
SECTION 1. INTRODUCTION	1-1
Introduction	
Manual Objective	
Development Support	
Style and Symbol Conventions	
Instruction Set Symbols	
Related Documents	
SECTION 2. PROGRAMMER'S MODEL	2-1
dsPIC30F Overview	
Programmer's Model	2-3
SECTION 3. INSTRUCTION SET OVERVIEW	3-1
Introduction	
Instruction Set Overview	
Instruction Set Summary Tables	
SECTION 4. INSTRUCTION SET DETAILS	4-1
Data Addressing Modes	
Program Addressing Modes	
Instruction Stalls	
Byte Operations	
Word Move Operations	
Using 10-bit Literal Operands	
Software Stack Pointer and Frame Pointer	
Conditional Branch Instructions	
Z Status Bit	
Assigned Working Register Usage	
DSP Data Formats	
Accumulator Usage	
Accumulator Access	
DSP MAC Instructions	
DSP Accumulator Instructions	
Scaling Data with the FBCL Instruction Normalizing the Accumulator with the FBCL Instruction	
SECTION 5. INSTRUCTION DESCRIPTIONS	5-1
Instruction Symbols	
Instruction Encoding Field Descriptors Introduction	
Instruction Description Example	
Instruction Descriptions	
SECTION 6. REFERENCE	6-1
Data Memory Map	
Core Special Function Register Map	
Program Memory Map	
Instruction Bit Map	
Instruction Set Summary Table	

~ -

NOTES:

1



Section 1. Introduction

HIGHLIGHTS

This section of the manual contains the following topics:

1.1	Introduction	. 1-2
1.2	Manual Objective	. 1-2
1.3	Development Support	. 1-2
1.4	Style and Symbol Conventions	. 1-3
1.5	Instruction Set Symbols	. 1-4
1.6	Related Documents	. 1-5

1.1 Introduction

Microchip Technology's focus is on products that meet the needs of the embedded control market. We are a leading supplier of:

- 8-bit general purpose microcontrollers (PICmicro[®] MCUs)
- dsPIC30F 16-bit microcontrollers
- · Speciality and standard non-volatile memory devices
- Security devices (KEELOQ[®])
- · Application specific standard products

Please request a Microchip Product Line Card for a listing of all the interesting products that we have to offer. This literature can be obtained from your local sales office, or downloaded from the Microchip web site (www.microchip.com).

1.2 Manual Objective

PICmicro and dsPIC30F devices are grouped by the size of their Instruction Word and Data Path. The current device families are:

- 1. Base-Line: 12-bit Instruction Word length, 8-bit Data Path
- 2. Mid-Range: 14-bit Instruction Word length, 8-bit Data Path
- 3. High-End: 16-bit Instruction Word length, 8-bit Data Path
- 4. Enhanced: 16-bit Instruction Word length, 8-bit Data Path
- 5. dsPIC30F: 24-bit Instruction Word length, 16-bit Data Path

This manual is a software developer's reference for the dsPIC30F 16-bit MCU family of devices. This manual describes the Instruction Set in detail and also provides general information to assist the user in developing software for the dsPIC30F MCU family.

This manual does not include detailed information about the core, peripherals, system integration or device-specific information. The user should refer to the dsPIC30F MCU Family Reference Manual for information about the core, peripherals and system integration. For device specific information, the user should refer to the data sheet. The information that can be found in the data sheet includes:

- Device memory map
- Device pinout and packaging details
- Device electrical specifications
- List of peripherals included on the device.

Code examples are given throughout this manual. These examples are valid for any device in the dsPIC30F MCU family.

1.3 Development Support

Microchip offers a wide range of development tools that allow users to efficiently develop and debug application code. Microchip's development tools can be broken down into four categories:

- 1. Code generation
- 2. Hardware/Software debug
- 3. Device programmer
- 4. Product evaluation boards

Information about the latest tools, product briefs and user guides can be obtained from the Microchip web site (www.microchip.com) or from your local Microchip Sales Office.

Microchip offers other reference tools to speed the development cycle. These include:

- Application Notes
- Reference Designs
- Microchip web site
- Local Sales Offices with Field Application Support
- Corporate Support Line

The Microchip web site lists other sites that may be useful references.

1.4 Style and Symbol Conventions

Throughout this document, certain style and font format conventions are used. Most format conventions imply a distinction should be made for the emphasized text. The MCU industry has many symbols and non-conventional word definitions/abbreviations. Table 1-1 provides a description for many of the conventions contained in this document.

Table 1-1:	Document Conventions

Symbol or Term	Description
set	To force a bit/register to a value of logic '1'.
clear	To force a bit/register to a value of logic '0'.
RESET	 To force a register/bit to its default state. A condition in which the device places itself after a device RESET occurs. Some bits will be forced to '0' (such as interrupt enable bits), while others will be forced to '1' (such as the I/O data direction bits).
0xnnnn	Designates the number 'nnnn' in the hexadecimal number system. These conventions are used in the code examples. For example, 0x013F or 0xA800.
: (colon)	Used to specify a range or the concatenation of registers/bits/pins. One example is ACCAU:ACCAH:ACCAL, which is the concatenation of three registers to form the 40-bit accumulator. Concatenation order (left-right) usually specifies a positional relationship (MSb to LSb, higher to lower).
<>	Specifies bit(s) locations in a particular register. One example is SR <ipl2:ipl0> (or IPL<2:0>), which specifies the register and associated bits or bit positions.</ipl2:ipl0>
MSb, MSbit, LSb, LSbit	Indicates the Least Significant or Most Significant bit in a field.
MSByte, MSWord, LSByte, LSWord	Indicates the Least/Most Significant Byte or Word in a field of bits.
Courier Font	Used for code examples, binary numbers and for Instruction Mnemonics in the text.
Times Font	Used for equations and variables.
Times, Bold Font, Italics	Used in explanatory text for items called out from a graphic/equation/example.
Note:	A Note presents information that we wish to re-emphasize, either to help you avoid a common pitfall, or make you aware of operating differences between some device family members. In most instances, a Note is used in a shaded box (as illustrated below), however when referenced to a table, a Note will stand-alone and immediately follow the associated table (as illustrated below Table 1-2).
	Note: This is a Note in a shaded note box.

1.5 Instruction Set Symbols

The Summary Tables in Section 3-2 and Section 6.5, and the instruction descriptions in Section 5.4 utilize the symbols shown in Table 1-2.

Table 1-2:Symbols Used in Instruction Summary Tables and Descriptions

Symbol	Description
{}	Optional field or operation
[text]	The location addressed by text
(text)	The contents of text
#text	The literal defined by text
$a\in [b,c,d]$	"a" must be in the set of [b, c, d]
<n:m></n:m>	Register bit field
{label:}	Optional label name
Acc	Accumulator A or Accumulator B
AWB	Accumulator Write Back
bit4	4-bit wide bit position (0:7 in Byte mode, 0:15 in Word mode)
Expr	Absolute address, label or expression (resolved by the linker)
f	File register address
lit1	1-bit literal (0:1)
lit4	4-bit literal (0:15)
lit5	5-bit literal (0:31)
lit8	8-bit literal (0:255)
lit10	10-bit literal (0:255 in Byte mode, 0:1023 in Word mode)
lit14	14-bit literal (0:16383)
lit16	16-bit literal (0:65535)
lit23	23-bit literal (0:8388607)
Slit4	Signed 4-bit literal (-8:7)
Slit6	Signed 6-bit literal (-32:31) (range is limited to -16:16)
Slit10	Signed 10-bit literal (-512:511)
Slit16	Signed 16-bit literal (-32768:32767)
TOS	Top-of-Stack
Wb	Base working register
Wd	Destination working register (direct and indirect addressing)
Wm, Wn	Working register divide pair (dividend, divisor)
Wm*Wm	Working register multiplier pair (same source register)
Wm*Wn	Working register multiplier pair (different source registers)
Wn	Both source and destination working register (direct addressing)
Wnd	Destination working register (direct addressing)
Wns	Source working register (direct addressing)
WREG	Default working register (assigned to W0)
Ws	Source working register (direct and indirect addressing)
Wx	Source Addressing mode and working register for X data bus pre-fetch
Wxd	Destination working register for X data bus pre-fetch
Wy	Source Addressing mode and working register for Y data bus pre-fetch
Wyd	Destination working register for Y data bus pre-fetch
Note: Th	he range of each symbol is instruction dependent. Refer to Section 5, "Instruction

Note: The range of each symbol is instruction dependent. Refer to **Section 5. "Instruction Descriptions"** for the specific instruction range.

1.6 Related Documents

Microchip, as well as other sources, offer additional documentation which can aid in your development with dsPIC30F MCUs. These lists contain the most common documentation, but other documents may also be available. Please check the Microchip web site (www.microchip.com) for the latest published technical documentation.

1.6.1 Microchip Documentation

The following dsPIC30F documentation is available from Microchip at the time of this writing. Many of these documents provide application specific information that gives actual examples of using, programming and designing with dsPIC30F MCUs.

1. dsPIC30F MCU Family Reference Manual (DS70046)

The dsPIC30F Family Reference Manual provides information about the dsPIC30F architecture, peripherals and system integration features. The details of device operation are provided in this document, along with numerous code examples.

2. dsPIC30F High Performance 16-Bit Digital Signal Controller Family Overview (DS70043)

This document provides a summary of the available dsPIC30F family variants, including device pinouts, memory sizes and available peripherals.

3. dsPIC30F Data Sheets

The data sheets contain device specific information, such as pinout and packaging details, electrical specifications, and memory maps. Please check the Microchip web site (www.microchip.com) for a list of available device data sheets.

1.6.2 Third Party Documentation

There are several documents available from third party sources around the world. Microchip does not review these documents for technical accuracy. However, they may be a helpful source for understanding the operation of Microchip dsPIC30F devices. Please refer to the Microchip web site (www.microchip.com) for third party documentation related to the dsPIC30F.

NOTES:



Section 2. Programmer's Model

HIGHLIGHTS

This section of the manual contains overview information about the dsPIC30F devices. It contains the following major topics:

2.1	dsPIC30F Overview	. 2-2
2.2	Programmer's Model	. 2-3

2.1 dsPIC30F Overview

The dsPIC30F core is a 16-bit (data) modified Harvard architecture with an enhanced instruction set, including support for DSP. The core has a 24-bit instruction word, with a variable length opcode field. The program counter (PC) is 23-bits wide and addresses up to 4M x 24 bits of user program memory space. A single cycle instruction pre-fetch mechanism is used to help maintain throughput and provides predictable execution. The majority of instructions execute in a single cycle, and overhead free program loop constructs are supported using the DO and REPEAT instructions, both of which are interruptible.

The dsPIC30F has sixteen, 16-bit working registers. Each of the working registers can act as a data, address or offset register. The 16th working register (W15) operates as a software stack pointer for interrupts and calls.

The dsPIC30F instruction set has two classes of instructions: the MCU class of instructions and the DSP class of instructions. These two instruction classes are seamlessly integrated into the architecture and execute from a single execution unit. The instruction set includes many Addressing modes and was designed for optimum C compiler efficiency.

The data space can be addressed as 32K words or 64 Kbytes and is split into two blocks, referred to as X and Y data memory. Each memory block has its own independent Address Generation Unit (AGU). The MCU class of instructions operate solely through the X memory AGU, which accesses the entire memory map as one linear data space. The DSP dual source class of instructions operates through the X and Y AGUs, which splits the data address space into two parts. The X and Y data space boundary is arbitrary and device specific.

The upper 32 Kbytes of the data space memory map can optionally be mapped into program space at any 16K program word boundary, defined by the 8-bit Program Space Visibility Page (PSVPAG) register. The program to data space mapping feature lets any instruction access program space as if it were data space, which is useful for storing data coefficients.

Overhead free circular buffers (modulo addressing) are supported in both X and Y address spaces. The modulo addressing removes the software boundary checking overhead for DSP algorithms. Furthermore, the X AGU circular addressing can be used with any of the MCU class of instructions. The X AGU also supports bit-reverse addressing, to greatly simplify input or output data reordering for radix-2 FFT algorithms.

The core supports Inherent (no operand), Relative, Literal, Memory Direct, Register Direct, Register Indirect and Register Offset Addressing modes. Each instruction is associated with a predefined Addressing mode group, depending upon its functional requirements. As many as 7 Addressing modes are supported for each instruction.

For most instructions, the dsPIC30F 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 can be supported, allowing A+B=C operations to be executed in a single cycle.

The DSP engine 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. The barrel shifter is capable of shifting a 40-bit value, up to 16-bits right, or up to 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 instruction and other associated instructions can concurrently fetch two data operands from memory while multiplying two working registers. This requires that the data space be 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.

The dsPIC30F has a vectored exception scheme with up to 8 sources of non-maskable traps and 54 interrupt sources. Each interrupt source can be assigned to one of seven priority levels.

2.2 Programmer's Model

The programmer's model diagram for the dsPIC30F is shown in Figure 2-1.

All registers in the programmer's model are memory mapped and can be manipulated directly by the instruction set. A description of each register is provided in Table 2-1.

Register	Description
ACCA, ACCB	40-bit DSP Accumulators
CORCON	CPU Core Configuration register
DCOUNT	DO Loop Count register
DOEND	DO Loop End Address register
DOSTART	DO Loop Start Address register
PC	23-bit Program Counter
PSVPAG	Program Space Visibility Page Address register
RCOUNT	Repeat Loop Count register
SPLIM	Stack Pointer Limit Value register
SR	ALU and DSP Engine Status register
TBLPAG	Table Memory Page Address register
W0 - W15	Working register array

 Table 2-1:
 Programmer's Model Register Descriptions

2.2.1 Working Register Array

The 16 working (W) registers can function as data, address or offset registers. The function of a W register is determined by the instruction that accesses it.

Byte instructions, which target the working register array, only affect the Least Significant Byte of the target register. Since the working registers are memory mapped, the Least *and* Most Significant Bytes can be manipulated through byte wide data memory space accesses.

2.2.2 Default Working Register (WREG)

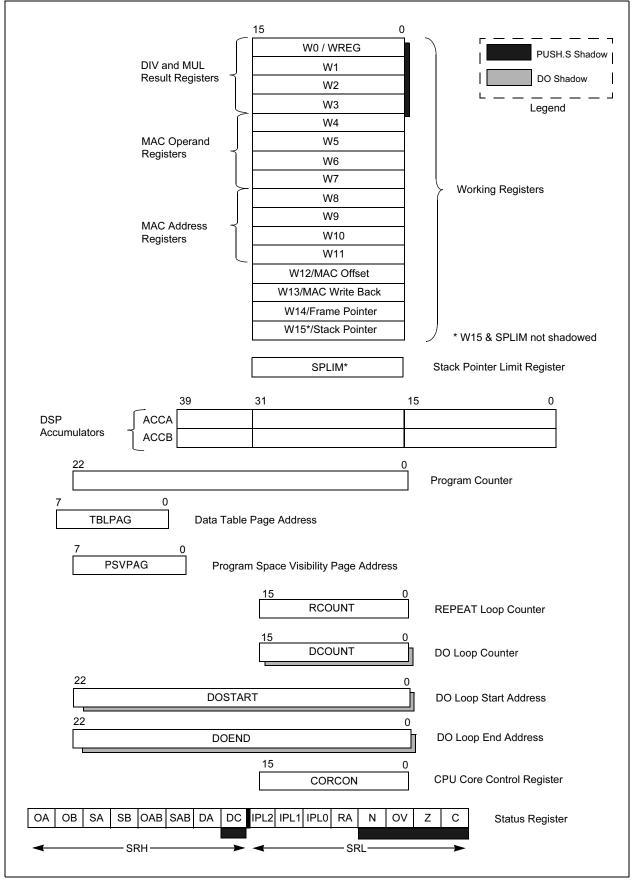
The dsPIC30F instruction set can be divided into two instruction types: working register instructions and file register instructions. The working register instructions use the working register array as data values, or as addresses that point to a memory location. In contrast, file register instructions operate on a specific memory address contained in the instruction opcode.

File register instructions that also utilize a working register do not specify the working register that is to be used for the instruction. Instead, a default working register (WREG) is used for these file register instructions. Working register W0 is assigned to be the WREG. The WREG assignment is not programmable.

2.2.3 Software Stack Frame Pointer

A frame is a user defined section of memory in the stack, used by a function to allocate memory for local variables. W14 has been assigned for use as a stack frame pointer with the link (LNK) and unlink (ULNK) instructions. However, if a stack frame pointer and the LNK and ULNK instructions are not used, W14 can be used by any instruction in the same manner as all other W registers. See **Section 4.7.3 "Software Stack Frame Pointer"** for detailed information about the Frame Pointer.

Figure 2-1: Programmer's Model Diagram



2.2.4 Software Stack Pointer

W15 serves as a dedicated software stack pointer, and will be automatically modified by function calls, exception processing and returns. However, W15 can be referenced by any instruction in the same manner as all other W registers. This simplifies reading, writing and manipulating the stack pointer. Refer to **Section 4.7.1 "Software Stack Pointer"** for detailed information about the stack pointer.

2.2.5 Stack Pointer Limit Register (SPLIM)

The SPLIM is a 16-bit register associated with the stack pointer. It is used to prevent the stack pointer from overflowing and accessing memory beyond the user allocated region of stack memory. Refer to **Section 4.7.5 "Stack Pointer Overflow"** for detailed information about the SPLIM.

2.2.6 Accumulator A, Accumulator B

Accumulator A (ACCA) and Accumulator B (ACCB) are 40-bit wide registers, utilized by DSP instructions to perform mathematical and shifting operations. Each accumulator is composed of 3 memory mapped registers:

- AccxU (bits 39 32)
- AccxH (bits 31 16)
- AccxL (bits 15 0)

Refer to **Section 4.12 "Accumulator Usage"** for details on using ACCA and ACCB.

2.2.7 Program Counter

The Program Counter (PC) is 23-bits wide. Instructions are addressed in the 4M x 24-bit user program memory space by PC<22:1>, where PC<0> is always set to '0' to maintain instruction word alignment and provide compatibility with data space addressing. This means that during normal instruction execution, the PC increments by 2.

Program memory located at 0x8000000 and above is utilized for device configuration data, Unit ID and Device ID. This region is not available for user code execution and the PC can not access this area. However, one may access this region of memory using Table instructions. Refer to the dsPIC30F MCU Family Reference Manual for details on accessing the configuration data, Unit ID and Device ID.

2.2.8 TBLPAG Register

The TBLPAG register is used to hold the upper 8 bits of a program memory address during table read and write operations. Table instructions are used to transfer data between program memory space and data memory space. Refer to the dsPIC30F MCU Family Reference Manual for details on accessing program memory with the Table instructions.

2.2.9 PSVPAG Register

Program space visibility allows the user to map a 32 Kbyte section of the program memory space into the upper 32 Kbytes of data address space. This feature allows transparent access of constant data through dsPIC30F instructions that operate on data memory. The PSVPAG register selects the 32 Kbyte region of program memory space that is mapped to the data address space. Refer to the dsPIC30F MCU Family Reference Manual for details on program space visibility.

2.2.10 RCOUNT Register

The 14-bit RCOUNT register contains the loop counter for the REPEAT instruction. When a REPEAT instruction is executed, RCOUNT is loaded with the repeat count of the instruction, either "lit14" for the "REPEAT #lit14" instruction, or the contents of Wn for the "REPEAT Wn" instruction. The REPEAT loop will be executed RCOUNT+1 times.

- **Note 1:** If a REPEAT loop is executing and gets interrupted, RCOUNT may be cleared by the Interrupt Service Routine to break out of the REPEAT loop when the foreground code is re-entered.
 - Refer to the dsPIC30F Family Reference Manual for complete details about REPEAT loops.

2.2.11 DCOUNT Register

The 14-bit DCOUNT register contains the loop counter for hardware DO loops. When a DO instruction is executed, DCOUNT is loaded with the loop count of the instruction, either "lit14" for the "DO #lit14, Expr" instruction, or the 14 Least Significant bits of Ws for the "DO Ws, Expr" instruction. The DO loop will be executed DCOUNT+1 times.

Note 1: DCOUNT contains a shadow register. See Section 2.2.16 "Shadow Registers" for information on shadowing.

2: Refer to the dsPIC30F Family Reference Manual for complete details about DO loops.

2.2.12 DOSTART Register

The DOSTART register contains the starting address for a hardware DO loop. When a DO instruction is executed, DOSTART is loaded with the address of the instruction following the DO instruction. This location in memory is the start of the DO loop. When looping is activated, program execution continues with the instruction stored at the DOSTART address after the last instruction in the DO loop is executed. This mechanism allows for zero overhead looping.

Note 1: DOSTART has a shadow register. See **Section 2.2.16 "Shadow Registers"** for information on shadowing.

2: Refer to the dsPIC30F Family Reference Manual for complete details about DO loops.

2.2.13 DOEND Register

The DOEND register contains the ending address for a hardware DO loop. When a DO instruction is executed, DOEND is loaded with the address specified by the expression in the DO instruction. This location in memory specifies the last instruction in the DO loop. When looping is activated and the instruction stored at the DOEND address is executed, program execution will continue from the DO loop start address (stored in the DOSTART register).

Note 1: DOEND has a shadow register. See Section 2.2.16 "Shadow Registers" for information on shadowing.

2: Refer to the dsPIC30F Family Reference Manual for complete details about DO loops.

2.2.14 Status Register

The 16-bit Status register, shown in Register 2-1, maintains status information for instructions which have most recently been executed. Operation status bits exist for MCU operations, loop operations and DSP operations. Additionally, the Status register contains the CPU Interrupt Priority Level bits, IPL<2:0>, which are used for interrupt processing.

2.2.14.1 MCU ALU Status Bits

The MCU operation status bits are either affected or used by the majority of instructions in the instruction set. Most of the Logic, Math, Rotate/Shift and Bit instructions modify the MCU status bits after execution, and the conditional Branch instructions use the state of individual status bits to determine the flow of program execution. All conditional Branch instructions are listed in **Section 4.8 "Conditional Branch Instructions"**.

The Carry, Zero, Overflow, Negative and Digit Carry (C, Z, OV, N and DC) bits are used to show the immediate status of the MCU ALU. They indicate when an operation has resulted in a carry, zero, overflow, negative result and digit carry, respectively. When a subtract operation is performed, the C flag is used as a Borrow flag.

The Z status bit is a special zero status bit that is useful for extended precision arithmetic. The Z bit functions like a normal Z flag for all instructions except those that use a carry or borrow input (ADDC, CPB, SUBB and SUBBR). See **Section 4.9 "Z Status Bit**" for usage of the Z status bit.

- Note 1: All MCU bits are shadowed during execution of the PUSH.S instruction and they are restored on execution of the POP.S instruction.
 - 2: All MCU bits, except the DC flag (which is not in the SRL), are stacked during exception processing (see Section 4.7.1 "Software Stack Pointer").

2.2.14.2 Loop Status Bits

The DO Active and REPEAT Active (DA, RA) bits are used to indicate when looping is active. The DO instructions affect the DA flag, which indicates that a DO loop is active. The DA flag is set to '1' when the first instruction of the DO loop is executed, and it is cleared when the last instruction of the loop completes final execution. Likewise, the RA flag indicates that a REPEAT instruction is being executed, and it is only affected by the REPEAT instructions. The RA flag is set to '1' when the instruction being repeated begins execution, and it is cleared when the instruction being repeated begins execution, and it is cleared when the instruction being repeated completes execution for the last time.

The DA flag is read only. This means that looping may not be initiated by writing a '1' to DA, nor may looping be terminated by writing a '0' to DA. If a DO loop must be terminated prematurely, the EDT bit, CORCON<11>, should be used.

Since the RA flag is also read only, it may not be directly cleared. However, if a REPEAT or its target instruction is interrupted, the Interrupt Service Routine may clear the RA flag of the SRL, which resides on the stack. This action will disable looping once program execution returns from the Interrupt Service Routine, because the restored RA will be '0'.

2.2.14.3 DSP ALU Status Bits

The high byte of the Status Register (SRH) is used by the DSP class of instructions, and it is modified when data passes through one of the adders. The SRH provides status information about overflow and saturation for both accumulators. The Saturate A, Saturate B, Overflow A and Overflow B (SA, SB, OA, OB) bits provide individual accumulator status, while the Saturate AB and Overflow AB (SAB, OAB) bits provide combined accumulator status. The SAB and OAB bits provide the software developer efficiency in checking the register for saturation or overflow.

The OA and OB bits are used to indicate when an operation has generated an overflow into the guard bits (bits 32 through 39) of the respective accumulator. This condition can only occur when the processor is in Super Saturation mode, or if saturation is disabled. It indicates that the operation has generated a number which cannot be represented with the lower 31 bits of the accumulator.

The SA and SB bits are used to indicate when an operation has generated an overflow out of the Most Significant bit of the respective accumulator. The SA and SB bits are active, regardless of the Saturation mode (Disabled, Normal or Super) and may be considered "sticky". Namely, once the SA or SB is set to '1', it can only be cleared manually by software, regardless of subsequent DSP operations. When required, it is recommended that the bits be cleared with the BCLR instruction.

For convenience, the OA and OB bits are logically ORed together to form the OAB flag, and the SA and SB bits are logically ORed to form the SAB flag. These cumulative status bits provide efficient overflow and saturation checking when an algorithm is implemented, which utilizes both accumulators. Instead of interrogating the OA and the OB bits independently for arithmetic overflows, a single check of OAB may be performed. Likewise, when checking for saturation, SAB may be examined instead of checking both the SA and SB bits. Note that clearing the SAB flag will clear both the SA and SB bits.

2.2.14.4 Interrupt Priority Level Status Bits

The three IPL bits of the SRL, SR<7:5>, and the IPL3 bit, CORCON<3>, set the CPU's Interrupt Priority Level (IPL) which is used for exception processing. Exceptions consist of interrupts and hardware traps. Interrupts have a user defined priority level between 0 and 7, while traps have a fixed priority level between 8 and 15. The fourth Interrupt Priority Level bit, IPL3, is a special IPL bit that may only be read or cleared by the user. This bit is only set when a hardware trap is activated and it is cleared after the trap is serviced.

The CPU's IPL identifies the lowest level exception which may interrupt the processor. The interrupt level of a pending exception must always be greater than the CPU's IPL for the CPU to process the exception. This means that if the IPL is '0', all exceptions at priority Level 1 and above may interrupt the processor. If the IPL is '7', only hardware traps may interrupt the processor.

When an exception is serviced, the IPL is automatically set to the priority level of the exception being serviced, which will disable all exceptions of equal and lower priority. However, since the IPL field is read/write, one may modify the lower three bits of the IPL in an Interrupt ServiceRoutine to control which exceptions may preempt the exception processing. Since the SRL is stacked during exception processing, the original IPL is always restored after the exception is serviced. If required, one may also prevent exceptions from nesting by setting the NSTDIS bit, INTCON1<15>.

Note: Refer to the dsPIC30F Family Reference Manual for complete details on exception processing.

2.2.15 Core Control Register

The 16-bit CPU Core Control Register (CORCON), shown in Register 2-2, is used to set the configuration of the dsPIC30F CPU. This register provides the ability to:

- map program space into data space
- set the ACCA and ACCB saturation enable
- · set the Data Space Write Saturation mode
- set the Accumulator Saturation and Rounding modes
- · set the Multiplier mode for DSP operations
- terminate DO loops prematurely

On device RESET, the CORCON is set to 0x0020, which sets the following mode:

- Program Space not Mapped to Data Space (PSV = 0)
- ACCA and ACCB Saturation Disabled (SATA = 0, SATB = 0)
- Data Space Write Saturation Enabled (SATDW = 1)
- Accumulator Saturation mode set to normal (ACCSAT = 0)
- Accumulator Rounding mode set to unbiased (RND = 0)
- DSP Multiplier mode set to signed fractional (**US** = 0, **IF** = 0)

In addition to setting CPU modes, the CORCON contains status information about the DO loop nesting level (**DL**<2:0>) and the **IPL**<3> status bit, which indicates if a trap exception is being processed.

2.2.16 Shadow Registers

A shadow register is used as a temporary holding register and can transfer its contents to or from the associated host register upon some event. Some of the registers in the programmer's model have a shadow register, which is utilized during the execution of a DO, POP.S or PUSH.S instruction. Shadow register usage is shown in Table 2-2.

Location	DO	POP.S/PUSH.S
DCOUNT	Yes	—
DOSTART	Yes	—
DOEND	Yes	—
Status Register - DC, N, OV, Z and C bits	_	Yes
W0 - W3	—	Yes

Table 2-2: Automatic Shadow Register Usage

Since the DCOUNT, DOSTART and DOEND registers are shadowed, the ability to nest DO loops without additional overhead is provided. Since all shadow registers are one register deep, up to one level of DO loop nesting is possible. Further nesting of DO loops is possible in software, with support provided by the DO Loop Nesting Level Status bits in the CORCON, CORCON<10:8>.

Note: All shadow registers are one register deep and are not directly accessible. Additional shadowing may be performed in software using the software stack.

dsPIC30F Programmer's Reference Manual

High By	te (SRH):								
R-0	R-0	R/C-0	R/C-0	R-0	R/C-C		-	V-0	
OA 	OB	SA	SB	OAB	SAB	DA		C	
bit 15								bit 8	
	Low	Byte (SRL):							
		V-0 R/W	-0 R/W	-0 F	R-0	R/W-0	R/W-0	R/W-0	R/W-0
		IPL<2	2:0>	F	RA	Ν	OV	Z	С
	bit 7								bit C
bit 15	1 = Accumu	ator A Overflo ator A overflov ator A has not	wed						
bit 14	OB: Accumul 1 = Accumu	ator B Overflo ator B overflov ator B has not	w bit ved						
bit 13	1 = Accumu	ator A Saturati ator A is satur ator A is not s	ated or has b	een saturat	ed at som	ne time			
		his bit may be ince this bit is a				/ software.			
pit 12	SB: Accumul 1 = Accumu	ator B Saturati ator B is satur ator B is not s	on bit ated or has b						
		his bit may be ince this bit is				y software.			
bit 11	1 = Accumu	B Combined A ators A or B h Accumulators A	ave overflowe	d					
bit 10	1 = Accumu	B Combined A ators A or B a Accumulators A	re saturated c	r have bee	n saturate	ed at some	time in the	e past	
	2: C	his bit may be ince this bit is a learing this bit	set, it must be	cleared m		y software.			
bit 9	DA: DO Loop 1 = DO loop 0 = DO loop		s						
	Note: The second	nis bit is read o	only.						
bit 8	1 = A carry-	U Half Carry b out from the M -out from the N	ost Significan						
bit 7-5	111 = CPU II 110 = CPU II 101 = CPU II 100 = CPU II 011 = CPU II 010 = CPU II 001 = CPU II	errupt Priority Interrupt Priority Interrupt Priority Interrupt Priority Interrupt Priority Interrupt Priority Interrupt Priority Interrupt Priority	/ Level is 7 (1 / Level is 6 (1 / Level is 5 (1 / Level is 4 (1 / Level is 3 (1 / Level is 2 (1 / Level is 1 (9	4) 3) 2) 1) 0)	errupts di	sabled.			

Note: The IPL<2:0> bits are concatenated with the IPL<3> bit (CORCON<3>) to form the CPU Interrupt Priority Level. The value in parentheses indicates the IPL, if IPL<3> = 1.

Register 2-1: SR, Status Register (Continued)

- bit 4 RA: REPEAT Loop Active bit
 1 = REPEAT loop in progress
 0 = REPEAT loop not in progress
 bit 3 N: MCU ALU Negative bit
 - 1 = The result of the operation was negative
 - 0 = The result of the operation was not negative
- bit 2 OV: MCU ALU Overflow bit
 - 1 = Overflow occurred
 - 0 = No overflow occurred
- bit 1 **Z:** MCU ALU Zero bit
 - 1 = The result of the operation was zero
 - 0 = The result of the operation was not zero
 - Note: Refer to Section 4.9 "Z Status Bit" for operation with ADDC, CPB, SUBB and SUBBR instructions.
- bit 0 C: MCU ALU Carry/Borrow bit
 - 1 = A carry-out from the Most Significant bit occurred
 - 0 = No carry-out from the Most Significant bit occurred

Legend:		
R = Readable bit	W = Writable bit	C = Clearable bit
-n = Value at POR	1 = bit is set	0 = bit is cleared

Programmer's Model

dsPIC30F Programmer's Reference Manual

High Byte	e:							
U	U	U	R/W-0	R(0)/W-0	R-0	R-0	R/W-0	
—	—	_	US	EDT		DL<2:0>	•	
bit 15							bit 8	
	Low E	-						
	R/W						R/W-0 R/W-0	
	SAT	TA SAT	B SATI	DW ACCS	SAI	IPL3	PSV RND	
	bit 7							bit C
bit 15-13	Unused							
bit 12	US: Unsigned	or Signed Mi	Iltinlier Mode	Select hit				
	1 = Unsigned 0 = Signed m	I mode enable	d for DSP m	ultiply operati				
bit 11	EDT: Early DC							
	1 = Terminate		O loop at end	d of current ite	eration			
	0 = No effect							
		is bit will alwa	-					
bit 10-8	DL<2:0>: DO			bits				
	110 = DO loop	-						
	110 = DO loop							
	110 = DO loop 011 = DO loop							
	011 - DO 100							
	001 = DO loop	oing is active,	but not neste	ed (just 1 leve	el)			
	000 = DO loop	oing is not act	ive					
		_<2:1> are rea le first two lev	•	op nesting are	handled	by hardware	Э.	
bit 7	SATA: ACCA							
	1 = Accumula 0 = Accumula							
bit 6	SATB: ACCB							
	1 = Accumula							
	0 = Accumula							
bit 5	SATDW: Data				on Enable	e bit		
	1 = Data space 0 = Data space							
bit 4	ACCSAT: Acc							
DIL 4	1 = 9.31 satu			Select bit				
	0 = 1.31 satu							
bit 3	IPL3: Interrup							
	1 = CPU Inte							
	0 = CPU Inte				-	activated)		
		iis bit may be iis bit is conca				':5>) to form	the CPU Interru	ot Priority Level
					(,		
bit 2		Space Visihi	lity in Data S	pace Enable I	oit			
bit 2	PSV: Program 1 = Program				oit			

bit 1 RND: Rounding Mode Select bit

- 1 = Biased (conventional) rounding enabled
- 0 = Unbiased (convergent) rounding enabled
- bit 0 IF: Integer or Fractional Multiplier Mode Select bit
 - 1 = Integer mode enabled for DSP multiply operations
 - 0 = Fractional mode enabled for DSP multiply operations

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

NOTES:



Section 3. Instruction Set Overview

HIGHLIGHTS

This section of the manual contains the following major topics:

3.1	Introduction	. 3-2
3.2	Instruction Set Overview	. 3-2
3.3	Instruction Set Summary Tables	. 3-3

3.1 Introduction

The dsPIC30F instruction set provides a broad suite of instructions, which supports traditional microcontroller applications and a class of instructions, which supports math intensive applications. Since almost all of the functionality of the PICmicro[®] MCU instruction set has been maintained, this hybrid instruction set allows a friendly DSP migration path for users already familiar with the PICmicro microcontroller.

3.2 Instruction Set Overview

The dsPIC30F instruction set contains 84 instructions, which can be grouped into the ten functional categories shown in Table 3-1. Table 1-2 defines the symbols used in the instruction summary tables, Table 3-2 through Table 3-11. These tables define the syntax, description, storage and execution requirements for each instruction. Storage requirements are represented in 24-bit instruction words and execution requirements are represented in instruction cycles.

Functional Group	Summary Table	Page #
Move Instructions	Table 3-2	3-3
Math Instructions	Table 3-3	3-4
Logic Instructions	Table 3-4	3-5
Rotate/Shift Instructions	Table 3-5	3-6
Bit Instructions	Table 3-6	3-7
Compare/Skip Instructions	Table 3-7	3-8
Program Flow Instructions	Table 3-8	3-9
Shadow/Stack Instructions	Table 3-9	3-10
Control Instructions	Table 3-10	3-10
DSP Instructions	Table 3-11	3-10

Table 3-1:dsPIC30F Instruction Groups

Most instructions have several different Addressing modes and execution flows, which require different instruction variants. For instance, there are six unique ADD instructions and each instruction variant has its own instruction encoding. Instruction format descriptions and specific instruction operation are provided in **Section 3. "Instruction Set Overview"**. Additionally, a composite alphabetized instruction set table is provided in **Section 6. "Reference"**.

3.2.1 Multi-Cycle Instructions

As the instruction summary tables show, most instructions execute in a single cycle, with the following exceptions:

- Instructions DO, MOV.D, POP.D, PUSH.D, TBLRDH, TBLRDL, TBLWTH and TBLWTL require 2 cycles to execute.
- Instructions DIV.S, DIV.U and DIVF are single cycle instructions, which should be executed 18 consecutive times as the target of a REPEAT instruction.
- Instructions that change the program counter also require 2 cycles to execute, with the extra cycle executed as a NOP. SKIP instructions, which skip over a 2-word instruction, require 3 instruction cycles to execute, with 2 cycles executed as a NOP.
- The RETFIE, RETLW and RETURN are a special case of an instruction that changes the program counter. These execute in 3 cycles, unless an exception is pending and then they execute in 2 cycles.

Note: Instructions which access program memory as data, using Program Space Visibility, will incur a one or two cycle delay. However, when the target instruction of a REPEAT loop accesses program memory as data, only the first execution of the target instruction is subject to the delay. See the dsPIC30F Family Reference Manual for details.

3.2.2 Multi-Word Instructions

As defined by **Subsection Table 3-2: "Move Instructions"**, almost all instructions consume one instruction word (24-bits), with the exception of the CALL, DO and GOTO instructions, which are Program Flow Instructions, listed in Table 3-8. These instructions require two words of memory because their opcodes embed large literal operands.

3.3 Instruction Set Summary Tables

Table 3-2: Move Instructions						
Assembly	Syntax	Description	Words	Cycles	Page #	
EXCH	Wns,Wnd	Swap Wns and Wnd	1	1	5-115	
MOV	f {,WREG} ^(see Note)	Move f to destination	1	1	5-145	
MOV	WREG,f	Move WREG to f	1	1	5-146	
MOV	f,Wnd	Move f to Wnd	1	1	5-147	
MOV	Wns,f	Move Wns to f	1	1	5-148	
MOV.B	#lit8,Wnd	Move 8-bit literal to Wnd	1	1	5-149	
MOV	#lit16,Wnd	Move 16-bit literal to Wnd	1	1	5-150	
MOV	[Ws+Slit10],Wnd	Move [Ws + signed 10-bit offset] to Wnd	1	1	5-151	
MOV	Wns,[Wd+Slit10]	Move Wns to [Wd + signed 10-bit offset]	1	1	5-152	
MOV	Ws,Wd	Move Ws to Wd	1	1	5-153	
MOV.D	Ws,Wnd	Move double Ws to Wnd:Wnd+1	1	2	5-155	
MOV.D	Wns,Wd	Move double Wns:Wns+1 to Wd	1	2	5-157	
SWAP	Wn	Wn = byte or nibble swap Wn	1	1	5-249	
TBLRDH	Ws,Wd	Read high program word to Wd	1	2	5-250	
TBLRDL	Ws,Wd	Read low program word to Wd	1	2	5-252	
TBLWTH	Ws,Wd	Write Ws to high program word	1	2	5-254	
TBLWTL	Ws,Wd	Write Ws to low program word	1	2	5-256	

Table 3-2: Move Instructions

Note: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

Table 3-3: Math Instructions							
Assembly	/ Syntax	Description	Words	Cycles	Page #		
ADD	f {,WREG} ⁽¹⁾	Destination = f + WREG	1	1	5-7		
ADD	#lit10,Wn	Wn = lit10 + Wn	1	1	5-8		
ADD	Wb,#lit5,Wd	Wd = Wb + lit5	1	1	5-9		
ADD	Wb,Ws,Wd	Wd = Wb + Ws	1	1	5-10		
ADDC	f {,WREG} ⁽¹⁾	Destination = f + WREG + (C)	1	1	5-14		
ADDC	#lit10,Wn	Wn = lit10 + Wn + (C)	1	1	5-15		
ADDC	Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1	5-16		
ADDC	Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1	5-17		
DAW.B	Wn	Wn = decimal adjust Wn	1	1	5-95		
DEC	f {,WREG} ⁽¹⁾	Destination = f – 1	1	1	5-96		
DEC	Ws,Wd	Wd = Ws - 1	1	1	5-97		
DEC2	f {,WREG} ⁽¹⁾	Destination = f – 2	1	1	5-98		
DEC2	Ws,Wd	Wd = Ws – 2	1	1	5-99		
DIV.S	Wm, Wn	Signed 16/16-bit integer divide	1	18 ⁽²⁾	5-101		
DIV.SD	Wm, Wn	Signed 32/16-bit integer divide	1	18 ⁽²⁾	5-101		
DIV.U	Wm, Wn	Unsigned 16/16-bit integer divide	1	18 ⁽²⁾	5-103		
DIV.UD	Wm, Wn	Unsigned 32/16-bit integer divide	1	18 ⁽²⁾	5-103		
DIVF	Wm, Wn	Signed 16/16-bit fractional divide	1	18 ⁽²⁾	5-105		
INC	f {,WREG} ⁽¹⁾	Destination = f + 1	1	1	5-124		
INC	Ws,Wd	Wd = Ws + 1	1	1	5-125		
INC2	f {,WREG} ⁽¹⁾	Destination = f + 2	1	1	5-126		
INC2	Ws,Wd	Wd = Ws + 2	1	1	5-127		
MUL	f	W3:W2 = f * WREG	1	1	5-169		
MUL.SS	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * sign(Ws)	1	1	5-170		
MUL.SU	Wb,#lit5,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(lit5)	1	1	5-172		
MUL.SU	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(Ws)	1	1	5-174		
MUL.US	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * sign(Ws)	1	1	5-176		
MUL.UU	Wb,#lit5,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(lit5)	1	1	5-178		
MUL.UU	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(Ws)	1	1	5-179		
SE	Ws,Wnd	Wnd = sign-extended Ws	1	1	5-220		
SUB	f {,WREG} ⁽¹⁾	Destination = f – WREG	1	1	5-230		
SUB	#lit10,Wn	Wn = Wn - lit10	1	1	5-231		
SUB	Wb,#lit5,Wd	Wd = Wb – lit5	1	1	5-232		
SUB	Wb,Ws,Wd	Wd = Wb – Ws	1	1	5-233		
SUBB	f {,WREG} ⁽¹⁾	Destination = $f - WREG - (\overline{C})$	1	1	5-236		
SUBB	#lit10,Wn	$Wn = Wn - lit10 - (\overline{C})$	1	1	5-237		
SUBB	Wb,#lit5,Wd	$Wd = Wb - lit5 - (\overline{C})$	1	1	5-238		
SUBB	Wb,Ws,Wd	$Wd = Wb - Ws - (\overline{C})$	1	1	5-239		
SUBBR	f {,WREG} ⁽¹⁾	Destination = WREG – f – (\overline{C})	1	1	5-241		
SUBBR	Wb,#lit5,Wd	$Wd = lit5 - Wb - (\overline{C})$	1	1	5-242		
SUBBR	Wb,Ws,Wd	$Wd = Ws - Wb - (\overline{C})$	1	1	5-243		
SUBR	f {,WREG} ⁽¹⁾	Destination = WREG – f	1	1	5-245		
SUBR	Wb,#lit5,Wd	Wd = lit5 – Wb	1	1	5-246		
SUBR	Wb,Ws,Wd	Wd = Ws – Wb	1	1	5-247		
ZE	Ws,Wnd	Wnd = zero-extended Ws	1	1	5-264		

Table 3-3: Math Instructions

Note 1: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

2: The divide instructions must be preceded with a "REPEAT #17" instruction, such that they are executed 18 consecutive times.

Assembly Syntax		Description	Words	Cycles	Page #
AND	f {,WREG} ^(see Note)	Destination = f .AND. WREG	1	1	5-19
AND	#lit10,Wn	Wn = lit10 .AND. Wn	1	1	5-20
AND	Wb,#lit5,Wd	Wd = Wb .AND. lit5	1	1	5-21
AND	Wb,Ws,Wd	Wd = Wb .AND. Ws	1	1	5-22
CLR	f	$f = 0 \times 0000$	1	1	5-75
CLR	WREG	WREG = 0x0000	1	1	5-75
CLR	Wd	Wd = 0x0000	1	1	5-76
COM	f {,WREG} ^(see Note)	Destination = \overline{f}	1	1	5-80
СОМ	Ws,Wd	$Wd = \overline{Ws}$	1	1	5-81
IOR	f {,WREG} ^(see Note)	Destination = f .IOR. WREG	1	1	5-128
IOR	#lit10,Wn	Wn = lit10 .IOR. Wn	1	1	5-129
IOR	Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1	5-130
IOR	Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1	5-131
NEG	f {,WREG} ^(see Note)	Destination = \overline{f} + 1	1	1	5-181
NEG	Ws,Wd	$Wd = \overline{Ws} + 1$	1	1	5-182
SETM	f	f = 0xFFFF	1	1	5-221
SETM	WREG	WREG = 0xffff	1	1	5-221
SETM	Wd	Wd = 0xFFFF	1	1	5-222
XOR	f {,WREG} ^(see Note)	Destination = f .XOR. WREG	1	1	5-259
XOR	#lit10,Wn	Wn = lit10 .XOR. Wn	1	1	5-260
XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1	5-261
XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1	5-262

Table 3-4: Logic Instructions

Note: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

Table 3-5: Rotate/Shift Instructions						
Assembly Syntax		Description	Words	Cycles	Page #	
ASR	f {,WREG} ^(see Note)	Destination = arithmetic right shift f	1	1	5-24	
ASR	Ws,Wd	Wd = arithmetic right shift Ws	1	1	5-25	
ASR	Wb,#lit4,Wnd	Wnd = arithmetic right shift Wb by lit4	1	1	5-27	
ASR	Wb,Wns,Wnd	Wnd = arithmetic right shift Wb by Wns	1	1	5-28	
LSR	f {,WREG} ^(see Note)	Destination = logical right shift f	1	1	5-136	
LSR	Ws,Wd	Wd = logical right shift Ws	1	1	5-137	
LSR	Wb,#lit4,Wnd	Wnd = logical right shift Wb by lit4	1	1	5-139	
LSR	Wb,Wns,Wnd	Wnd = logical right shift Wb by Wns	1	1	5-140	
RLC	f {,WREG} ^(see Note)	Destination = rotate left through Carry f	1	1	5-204	
RLC	Ws,Wd	Wd = rotate left through Carry Ws	1	1	5-205	
RLNC	f {,WREG} ^(see Note)	Destination = rotate left (no Carry) f	1	1	5-207	
RLNC	Ws,Wd	Wd = rotate left (no Carry) Ws	1	1	5-208	
RRC	f {,WREG} ^(see Note)	Destination = rotate right through Carry f	1	1	5-210	
RRC	Ws,Wd	Wd = rotate right through Carry Ws	1	1	5-211	
RRNC	f {,WREG} ^(see Note)	Destination = rotate right (no Carry) f	1	1	5-213	
RRNC	Ws,Wd	Wd = rotate right (no Carry) Ws	1	1	5-214	
SL	f {,WREG} ^(see Note)	Destination = left shift f	1	1	5-225	
SL	Ws,Wd	Wd = left shift Ws	1	1	5-226	
SL	Wb,#lit4,Wnd	Wnd = left shift Wb by lit4	1	1	5-228	
SL	Wb,Wns,Wnd	Wnd = left shift Wb by Wns	1	1	5-229	

Table 3-5: Rotate/Shift Instructions

Note: When the optional {,WREG} operand is specified, the destination of the instruction is WREG. When {,WREG} is not specified, the destination of the instruction is the file register f.

Section 3. Instruction Set Overview

Assembly	Syntax	Description	Words	Cycles	Page #
BCLR	f,#bit4	Bit clear f	1	1	5-29
BCLR	Ws,#bit4	Bit clear Ws	1	1	5-30
BSET	f,#bit4	Bit set f	1	1	5-54
BSET	Ws,#bit4	Bit set Ws	1	1	5-55
BSW.C	Ws,Wb	Write C bit to Ws <wb></wb>	1	1	5-56
BSW.Z	Ws,Wb	Write \overline{Z} bit to Ws <wb></wb>	1	1	5-56
BTG	f,#bit4	Bit toggle f	1	1	5-58
BTG	Ws,#bit4	Bit toggle Ws	1	1	5-59
BTST	f,#bit4	Bit test f	1	1	5-67
BTST.C	Ws,#bit4	Bit test Ws to C	1	1	5-68
BTST.Z	Ws,#bit4	Bit test Ws to Z	1	1	5-68
BTST.C	Ws,Wb	Bit test Ws <wb> to C</wb>	1	1	5-69
BTST.Z	Ws,Wb	Bit test Ws <wb> to Z</wb>	1	1	5-69
BTSTS	f,#bit4	Bit test f then set f	1	1	5-71
BTSTS.C	Ws,#bit4	Bit test Ws to C then set Ws	1	1	5-72
BTSTS.Z	Ws,#bit4	Bit test Ws to Z then set Ws	1	1	5-72
FBCL	Ws,Wnd	Find bit change from left (MSb) side	1	1	5-116
FF1L	Ws,Wnd	Find first one from left (MSb) side	1	1	5-118
FF1R	Ws,Wnd	Find first one from right (LSb) side	1	1	5-120

Table 3_6: Bit Instruction

Assemb	ly Syntax	Description	Words	Cycles ^(see Note)	Page #
BTSC	f,#bit4	Bit test f, skip if clear	1	1 (2 or 3)	5-60
BTSC	Ws,#bit4	Bit test Ws, skip if clear	1	1 (2 or 3)	5-62
BTSS	f,#bit4	Bit test f, skip if set	1	1 (2 or 3)	5-64
BTSS	Ws,#bit4	Bit test Ws, skip if set	1	1 (2 or 3)	5-65
CP	f	Compare (f – WREG)	1	1	5-82
CP	Wb,#lit5	Compare (Wb – lit5)	1	1	5-83
CP	Wb,Ws	Compare (Wb – Ws)	1	1	5-84
CP0	f	Compare (f - 0x0000)	1	1	5-85
CP0	Ws	Compare (Ws - 0x0000)	1	1	5-86
CPB	f	Compare with Borrow (f – WREG – \overline{C})	1	1	5-87
CPB	Wb,#lit5	Compare with Borrow (Wb – lit5 – \overline{C})	1	1	5-88
CPB	Wb,Ws	Compare with Borrow (Wb – Ws – \overline{C})	1	1	5-89
CPSEQ	Wb, Wn	Compare (Wb – Wn), skip if =	1	1 (2 or 3)	5-91
CPSGT	Wb, Wn	Compare (Wb – Wn), skip if >	1	1 (2 or 3)	5-92
CPSLT	Wb, Wn	Compare (Wb – Wn), skip if <	1	1 (2 or 3)	5-93
CPSNE	Wb, Wn	Compare (Wb – Wn), skip if ≠	1	1 (2 or 3)	5-94

Table 3-7: Compare/Skip Instructions

Note: Conditional skip instructions execute in 1 cycle if the skip is not taken, 2 cycles if the skip is taken over a one-word instruction and 3 cycles if the skip is taken over a two-word instruction.

Table 3-8:	Program F	low Instructions			
Assembly	Syntax	Description	Words	Cycles	Page #
BRA	Expr	Branch unconditionally	1	2	5-31
BRA	Wn	Computed branch	1	2	5-32
BRA	C,Expr	Branch if Carry (no Borrow)	1	1 (2) ⁽¹⁾	5-33
BRA	GE,Expr	Branch if greater than or equal	1	1 (2) ⁽¹⁾	5-35
BRA	GEU,Expr	Branch if unsigned greater than or equal	1	1 (2) ⁽¹⁾	5-33
BRA	GT,Expr	Branch if greater than	1	1 (2) ⁽¹⁾	5-37
BRA	GTU,Expr	Branch if unsigned greater than	1	1 (2) ⁽¹⁾	5-38
BRA	LE,Expr	Branch if less than or equal	1	1 (2) ⁽¹⁾	5-39
BRA	LEU,Expr	Branch if unsigned less than or equal	1	1 (2) ⁽¹⁾	5-40
BRA	LT,Expr	Branch if less than	1	1 (2) ⁽¹⁾	5-41
BRA	LTU,Expr	Branch if unsigned less than	1	1 (2) ⁽¹⁾	5-44
BRA	N,Expr	Branch if Negative	1	1 (2) ⁽¹⁾	5-43
BRA	NC,Expr	Branch if not Carry (Borrow)	1	1 (2) ⁽¹⁾	5-44
BRA	NN,Expr	Branch if not Negative	1	1 (2) ⁽¹⁾	5-45
BRA	NOV,Expr	Branch if not Overflow	1	1 (2) ⁽¹⁾	5-46
BRA	NZ,Expr	Branch if not Zero	1	1 (2) ⁽¹⁾	5-47
BRA	OA,Expr	Branch if Accumulator A Overflow	1	1 (2) ⁽¹⁾	5-48
BRA	OB,Expr	Branch if Accumulator B Overflow	1	1 (2) ⁽¹⁾	5-49
BRA	OV,Expr	Branch if Overflow	1	1 (2) ⁽¹⁾	5-50
BRA	SA,Expr	Branch if Accumulator A Saturate	1	1 (2) ⁽¹⁾	5-51
BRA	SB,Expr	Branch if Accumulator B Saturate	1	1 (2) ⁽¹⁾	5-52
BRA	Z,Expr	Branch if Zero	1	1 (2) ⁽¹⁾	5-53
CALL	Expr	Call subroutine	2	2	5-73
CALL	Wn	Call indirect subroutine	1	2	5-74
DO	#lit14,Expr	Do code through PC+Expr, (lit14+1) times	2	2	5-107
DO	Wn,Expr	Do code through PC+Expr, (Wn+1) times	2	2	5-109
GOTO	Expr	Go to address	2	2	5-122
GOTO	Wn	Go to address indirectly	1	2	5-123
RCALL	Expr	Relative call	1	2	5-196
RCALL	Wn	Computed call	1	2	5-196
REPEAT	#lit14	Repeat next instruction (lit14+1) times	1	1	5-197
REPEAT	Wn	Repeat next instruction (Wn+1) times	1	1	5-198
RETFIE		Return from interrupt enable	1	3 (2) ⁽²⁾	5-201
RETLW	#lit10,Wn	Return with lit10 in Wn	1	3 (2) ⁽²⁾	5-202
RETURN		Return from subroutine	1	3 (2) ⁽²⁾	5-203

Table 3-8: Program Flow Instructions

Note 1: Conditional branch instructions execute in 1 cycle if the branch is not taken, or 2 cycles if the branch is taken.

2: RETURN instructions execute in 3 cycles, but if an exception is pending, they execute in 2 cycles.

Table 3-9: Shadow/Stack Instructions

Assembly	y Syntax	Description	Words	Cycles	Page #
LNK	#lit14	Link frame pointer	1	1	5-135
POP	f	Pop TOS to f	1	1	5-186
POP	Wd	Pop TOS to Wd	1	1	5-187
POP.D	Wnd	Double pop from TOS to Wnd:Wnd+1	1	2	5-188
POP.S		Pop shadow registers	1	1	5-189
PUSH	f	Push f to TOS	1	1	5-190
PUSH	Ws	Push Ws to TOS	1	1	5-191
PUSH.D	Wns	Push double Wns:Wns+1 to TOS	1	2	5-192
PUSH.S		Push shadow registers	1	1	5-193
ULNK		Unlink frame pointer	1	1	5-258

Table 3-10: Control Instructions

Assembly Syntax		Description		Cycles	Page #
CLRWDT		Clear Watchdog Timer	1	1	5-79
DISI	#lit14	Disable interrupts for (lit14+1) instruction cycles	1	1	5-100
NOP		No operation	1	1	5-184
NOPR		No operation	1	1	5-185
PWRSAV	#lit1	Enter Power Saving mode lit1	1	1	5-194
RESET		Software device RESET	1	1	5-200

Table 3-11: DSP Instructions

Assembly	y Syntax	Description	Words	Cycles	Page #
ADD	Acc	Add accumulators	1	1	5-11
ADD	Ws,#Slit4,Acc	16-bit signed add to Acc	1	1	5-12
CLR	Acc,Wx,Wxd,Wy,Wyd,AWB	Clear Acc	1	1	5-77
ED	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance (no accumulate)	1	1	5-111
EDAC	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance	1	1	5-113
LAC	Ws,#Slit4,Acc	Load Acc	1	1	5-133
MAC	Wm*Wn,Acc,Wx,Wxd,Wy, Wyd,AWB	Multiply and accumulate	1	1	5-141
MAC	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square and accumulate	1	1	5-143
MOVSAC	Acc,Wx,Wxd,Wy,Wyd,AWB	Move Wx to Wxd and Wy to Wyd	1	1	5-159
MPY	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wn by Wm to Acc	1	1	5-161
MPY	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square to Acc	1	1	5-163
MPY.N	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wn by Wm) to Acc	1	1	5-165
MSC	Wm*Wn,Acc,Wx,Wxd,Wy, Wyd,AWB	Multiply and subtract from Acc	1	1	5-167
NEG	Acc	Negate Acc	1	1	5-183
SAC	Acc,#Slit4,Wd	Store Acc	1	1	5-216
SAC.R	Acc,#Slit4,Wd	Store rounded Acc	1	1	5-218
SFTAC	Acc,#Slit6	Arithmetic shift Acc by Slit6	1	1	5-223
SFTAC	Acc,Wn	Arithmetic shift Acc by (Wn)	1	1	5-224
SUB	Acc	Subtract accumulators	1	1	5-235



Section 4. Instruction Set Details

HIGHLIGHTS

This section of the manual contains the following major topics:

4.1	Data Addressing Modes	4-2
4.2	Program Addressing Modes	4-11
4.3	Instruction Stalls	4-12
4.4	Byte Operations	4-13
4.5	Word Move Operations	4-16
4.6	Using 10-bit Literal Operands	4-19
4.7	Software Stack Pointer and Frame Pointer	
4.8	Conditional Branch Instructions	4-25
4.9	Z Status Bit	4-26
4.10	Assigned Working Register Usage	4-27
4.11	DSP Data Formats	
4.12	Accumulator Usage	
4.13		
4.14	DSP MAC Instructions	
4.15	DSP Accumulator Instructions	4-37
4.16	Scaling Data with the FBCL Instruction	
4.17	Normalizing the Accumulator with the FBCL Instruction	4-39

4.1 Data Addressing Modes

The dsPIC30F supports three native Addressing modes for accessing data memory, along with several forms of immediate addressing. Data accesses may be performed using file register, register direct or register indirect addressing, and immediate addressing allows a fixed value to be used by the instruction.

File register addressing provides the ability to operate on data stored in the lower 8K of data memory (Near RAM), and also move data between the working registers and the entire 64K data space. Register direct addressing is used to access the 16 memory mapped working registers, W0:W15. Register indirect addressing is used to efficiently operate on data stored in the entire 64K data space, using the contents of the working registers as an effective address. Immediate addressing does not access data memory, but provides the ability to use a constant value as an instruction operand. The address range of each mode is summarized in Table 4-1.

Table 4-1: dsPIC30F Addressing Modes

Addressing Mode	Address Range
File Register	0x0000 - 0x1FFF ^(see Note)
Register Direct	0x0000 - 0x001F (working register array W0:W15)
Register Indirect	0x0000 - 0xFFFF
Immediate	N/A (constant value)

Note: The address range for the File Register MOV is 0x0000 - 0xFFFE.

4.1.1 File Register Addressing

File register addressing is used by instructions which use a predetermined data address as an operand for the instruction. The majority of instructions that support file register addressing provide access to the lower 8 Kbytes of data memory, which is called the Near RAM. However, the MOV instruction provides access to all 64 Kbytes of memory using file register addressing. This allows one to load data from any location in data memory to any working register, and store the contents of any working register to any location in data memory. It should be noted that file register addressing supports both byte and word accesses of data memory, with the exception of the MOV instruction, which accesses all 64K of memory as words. Examples of file register addressing are shown in Example 4-1.

Most instructions, which support file register addressing, perform an operation on the specified file register and the default working register WREG (see Section 2.2.2 "Default Working Register (WREG)"). If only one operand is supplied in the instruction, WREG is an implied operand and the operation results are stored back to the file register. In these cases, the instruction is effectively a read-modify-write instruction. However, when both the file register and WREG are specified in the instruction, the operation results are stored in WREG and the contents of the file register are unchanged. Sample instructions which show the interaction between the file register and WREG are shown in Example 4-2.

Note: Instructions which support file register addressing use 'f' as an operand in the instruction summary tables of **Section 3. "Instruction Set Overview"**.

Example 4-1: File Register Addressing

```
DEC
        0x1000
                        ; decrement data stored at 0x1000
Before Instruction:
  Data Memory 0x1000 = 0x5555
After Instruction:
  Data Memory 0x1000 = 0x5554
MOV
        0x27FE, W0
                       ; move data stored at 0x27FE to W0
Before Instruction:
  W0 = 0x5555
  Data Memory 0x27FE = 0x1234
After Instruction:
  W0 = 0x1234
  Data Memory 0x27FE = 0x1234
```

```
Example 4-2: File Register Addressing and WREG
```

```
AND
        0x1000
                         ; AND 0x1000 with WREG, store to 0x1000
Before Instruction:
  W0 (WREG) = 0x332C
  Data Memory 0x1000 = 0x5555
After Instruction:
  W0 (WREG) = 0x332C
  Data Memory 0 \times 1000 = 0 \times 1104
AND
        0x1000, WREG ; AND 0x1000 with WREG, store to WREG
Before Instruction:
  W0 (WREG) = 0x332C
  Data Memory 0x1000 = 0x5555
After Instruction:
  W0 (WREG) = 0 \times 1104
  Data Memory 0x1000 = 0x5555
```

4.1.2 Register Direct Addressing

Register direct addressing is used to access the contents of the 16 working registers (W0:W15). The Register Direct Addressing mode is fully orthogonal, which allows any working register to be specified for any instruction which uses register direct addressing, and it supports both byte and word accesses. Instructions which employ register direct addressing use the contents of the specified working register as data to execute the instruction, so this Addressing mode is useful only when data already resides in the working register core. Sample instructions which utilize register direct addressing are shown in Example 4-3.

Another feature of register direct addressing is that it provides the ability for dynamic flow control. Since variants of the DO and REPEAT instruction support register direct addressing, one may generate flexible looping constructs using these instructions.

Note: Instructions which must use register direct addressing, use the symbols Wb, Wn, Wns and Wnd in the summary tables of Section 3. "Instruction Set Overview". Commonly, register direct addressing may also be used when register indirect addressing may be used. Instructions which use register indirect addressing, use the symbols Wd and Ws in the summary tables of Section 3. "Instruction Set Overview".

Example 4-3: Register Direct Addressing

EXCH W2, W3 Before Instruction: W2 = 0x3499 W3 = 0x003D	; Exchange W2 and W3
After Instruction: W2 = 0x003D W3 = 0x3499	
IOR #0x44, W0 Before Instruction: W0 = 0x9C2E After Instruction:	; Inclusive-OR 0x44 and W0
W0 = 0x9C6E	; Shift left W6 by W7, and store to W8
Before Instruction: W6 = 0x000C W7 = 0x0008 W8 = 0x1234	
After Instruction: W6 = 0x000C W7 = 0x0008 W8 = 0x0C00	

4.1.3 **Register Indirect Addressing**

Register indirect addressing is used to access any location in data memory by treating the contents of a working register as an effective address (EA) to data memory. Essentially, the contents of the working register become a pointer to the location in data memory which is to be accessed by the instruction.

This Addressing mode is powerful, because it also allows one to modify the contents of the working register, either before or after the data access is made, by incrementing or decrementing the EA. By modifying the EA in the same cycle that an operation is being performed, register indirect addressing allows for the efficient processing of data that is stored sequentially in memory. The modes of indirect addressing supported by the dsPIC30F are shown in Table 4-2.

Indirect Mode	Syntax	Function (Byte Instruction)	Function (Word Instruction)	Description
No Modification	[Wn]	EA = [Wn]	EA = [Wn]	The contents of Wn forms the EA.
Pre-Increment	[++Wn]	EA = [Wn+=1]	EA = [Wn+=2]	Wn is pre-incremented to form the EA.
Pre-Decrement	[Wn]	EA = [Wn-=1]	EA = [Wn-=2]	Wn is pre-decremented to form the EA.
Post-Increment	[Wn++]	EA = [Wn]+= 1	EA = [Wn]+= 2	The contents of Wn forms he EA, then Wn is post-incremented.
Post-Decrement	[Wn]	EA = [Wn]-= 1	EA = [Wn]-= 2	The contents of Wn forms the EA, then Wn is post-decremented.
Register Offset	[Wn+Wb]	EA = [Wn+Wb]	EA = [Wn+Wb]	The sum of Wn and Wb forms the EA. Wn and Wb are not modified.

Table 4-2: Indirect Addressing Modes

Table 4-2 shows that four Addressing modes modify the EA used in the instruction, and this allows the following updates to be made to the working register: post-increment, post-decrement, pre-increment and pre-decrement. Since all EAs must be given as byte addresses, support is provided for Word mode instructions by scaling the EA update by 2. Namely, in Word mode, pre/post-decrements subtract 2 from the EA stored in the working register, and pre/post-increments add 2 to the EA. This feature ensures that after an EA modification is made, that the EA will point to the next adjacent word in memory. Example 4-4 shows how indirect addressing may be used to update the EA.

Table 4-2 also shows that the Register Offset mode addresses data which is offset from a base EA stored in a working register. This mode uses the contents of a second working register to form the EA by adding the two specified working registers. This mode does not scale for Word mode instructions, but offers the complete offset range of 64 Kbytes. Note that neither of the working registers used to form the EA are modified. Example 4-5 shows how register offset indirect addressing may be used to access data memory.

Note: The MOV with offset instructions (pages page 151 and page 152) provides a literal addressing offset ability to be used with indirect addressing. In these instructions, the EA is formed by adding the contents of a working register to a signed 10-bit literal. Example 4-6 shows how these instructions may be used to move data to and from the working register array.

```
MOV.B [W0++], [W13--]
                              ; byte move [W0] to [W13]
                               ; post-inc W0, post-dec W13
Before Instruction:
   W0 = 0x2300
   W13 = 0x2708
   Data Memory 0x2300 = 0x7783
   Data Memory 0x2708 = 0x904E
After Instruction:
   W0 = 0x2301
   W13 = 0x2707
   Data Memory 0x2300 = 0x7783
   Data Memory 0x2708 = 0x9083
       W1, [--W5], [++W8] ; pre-dec W5, pre-inc W8
ADD
                              ; add W1 to [W5], store in [W8]
Before Instruction:
   W1 = 0x0800
   W5 = 0x2200
   W8 = 0x2400
   Data Memory 0x21FE = 0x7783
   Data Memory 0x2402 = 0xAACC
After Instruction:
   W1 = 0x0800
   W5 = 0x21FE
   W8 = 0x2402
   Data Memory 0x21FE = 0x7783
   Data Memory 0x2402 = 0x7F83
```

```
MOV.B [W0+W1], [W7++]
                              ; byte move [W0+W1] to W7, post-inc W7
Before Instruction:
    W0 = 0x2300
    W1 = 0x01FE
    W7 = 0x1000
    Data Memory 0x24FE = 0x7783
    Data Memory 0x1000 = 0x11DC
After Instruction:
    W0 = 0x2300
    W1 = 0x01FE
    W7 = 0x1001
    Data Memory 0x24FE = 0x7783
    Data Memory 0x1000 = 0x1183
LAC
        [W0+W8], A
                               ; load ACCA with [W0+W8]
                                ; (sign-extend and zero-backfill)
Before Instruction:
    W0 = 0x2344
    W8 = 0 \times 0008
    ACCA = 0 \times 00 7877 9321
    Data Memory 0x234C = 0xE290
After Instruction:
    W0 = 0x2344
    W8 = 0 \times 0008
    ACCA = 0xFF E290 0000
    Data Memory 0x234C = 0xE290
```

Example 4-5: Indirect Addressing with Register Offset

```
Example 4-6: Move with Literal Offset Instructions
```

```
MOV
        [W0+0x20], W1
                              ; move [W0+0x20] to W1
Before Instruction:
    W0 = 0x1200
    W1 = 0x01FE
    Data Memory 0x1220 = 0xFD27
After Instruction:
    W0 = 0x1200
    W1 = 0xFD27
    Data Memory 0x1220 = 0xFD27
        W4, [W8-0x300]
MOV
                             ; move W4 to [W8-0x300]
Before Instruction:
    W4 = 0x3411
    W8 = 0x2944
    Data Memory 0x2644 = 0xCB98
After Instruction:
    W4 = 0x3411
    W8 = 0x2944
    Data Memory 0x2644 = 0x3411
```

S

Instruction

4.1.3.1 Register Indirect Addressing and the Instruction Set

The Addressing modes presented in Table 4-2 demonstrate the Indirect Addressing mode capability of the dsPIC30F. Due to operation encoding and functional considerations, not every instruction which supports indirect addressing supports all modes shown in Table 4-2. The majority of instructions which use indirect addressing support the No Modify, Pre-Increment, Pre-Decrement, Post-Increment and Post-Decrement Addressing modes. The MOV instructions, and several accumulator based DSP instructions, are also capable of using the Register Offset Addressing mode.

Note: Instructions which use register indirect addressing use the operand symbols Wd and Ws in the summary tables of **Section 3. "Instruction Set Overview"**.

4.1.3.2 DSP MAC Indirect Addressing Modes

A special class of Indirect Addressing modes is utilized by the DSP MAC instructions. As is described later in **Section 4.14 "DSP MAC Instructions"**, the DSP MAC class of instructions are capable of performing two fetches from memory using effective addressing. Since DSP algorithms frequently demand a broader range of address updates, the Addressing modes offered by the DSP MAC instructions provide greater range in the size of the effective address update which may be made. Table 4-3 shows that both X and Y pre-fetches support Post-Increment and Post-Decrement Addressing modes, with updates of 2, 4 and 6 bytes. Since DSP instructions only execute in Word mode, no provisions are made for odd sized EA updates.

X Memory	Y Memory
EA = [Wx]	EA = [Wy]
EA = [Wx]+= 2	EA = [Wy]+= 2
EA = [Wx]+= 4	EA = [Wy]+= 4
EA = [Wx]+= 6	EA = [Wy]+= 6
EA = [Wx]-= 2	EA = [Wy]-= 2
EA = [Wx]-= 4	EA = [Wy]-= 4
EA = [Wx]-= 6	EA = [Wy]-= 6
EA = [W9 + W12]	EA = [W11 + W12]
	EA = [Wx] $EA = [Wx]+= 2$ $EA = [Wx]+= 4$ $EA = [Wx]+= 6$ $EA = [Wx]-= 2$ $EA = [Wx]-= 4$ $EA = [Wx]-= 6$

Note: As described in Section 4.14 "DSP MAC Instructions", only W8 and W9 may be used to access X Memory, and only W10 and W11 may be used to access Y Memory.

4.1.3.3 Modulo and Bit-Reversed Addressing Modes

The dsPIC30F provides support for two special Register Indirect Addressing modes, which are commonly used to implement DSP algorithms. Modulo (or circular) addressing provides an automated means to support circular data buffers in X and/or Y memory. Modulo buffers remove the need for software to perform address boundary checks, which can improve the performance of certain algorithms. Similarly, Bit-Reversed addressing allows one to access the elements of a buffer in a non-linear fashion. This Addressing mode simplifies data re-ordering for radix-2 FFT algorithms and provides a significant reduction in FFT processing time.

Both of these Addressing modes are powerful features of the dsPIC30F architecture, which can be exploited by any instruction that uses indirect addressing. Refer to the dsPIC30F MCU Family Reference Manual for details on using Modulo and Bit-Reversed addressing.

4.1.4 Immediate Addressing

In immediate addressing, the instruction encoding contains a predefined constant operand, which is used by the instruction. This Addressing mode may be used independently, but it is more frequently combined with the File Register, Direct and Indirect Addressing modes. The size of the immediate operand which may be used varies with the instruction type. Constants of size 1-bit (#lit1), 4-bit (#bit4, #lit4 and #Slit4), 5-bit (#lit5), 6-bit (#Slit6), 8-bit (#lit8), 10-bit (#lit10 and #Slit10), 14-bit (#lit14) and 16-bit (#lit16) may be used. Constants may be signed or unsigned and the symbols #Slit4, #Slit6 and #Slit10 designate a signed constant. All other immediate constants are unsigned. Table 4-4 shows the usage of each immediate operand in the instruction set.

Operand	Instruction Usage	
#lit1	PWRSAV	
#bit4	BCLR, BSET, BTG, BTSC, BTSS, BTST, BTST . C, BTST . Z, BTSTS, BTSTS . C, BTSTS . Z	
#lit4	ASR, LSR, SL	
#Slit4	ADD, LAC, SAC, SAC.R	
#lit5	ADD, ADDC, AND, CP, CPB, IOR, MUL.SU, MUL.UU, SUB, SUBB, SUBBR, SUBR, XOR	
#Slit6	SFTAC	
#lit8	MOV.B	
#lit10	ADD, ADDC, AND, CP, CPB, IOR, RETLW, SUB, SUBB, XOR	
#Slit10	MOV	
#lit14	DISI, DO, LNK, REPEAT	
#lit16	MOV	

 Table 4-4:
 Immediate Operands in the Instruction Set

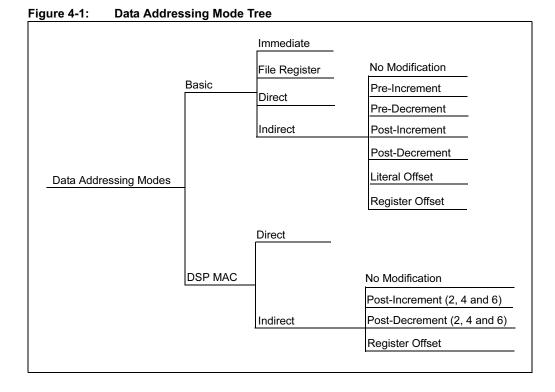
The syntax for immediate addressing requires that the number sign (#) must immediately precede the constant operand value. The "#" symbol indicates to the assembler that the quantity is a constant. If an out-of-range constant is used with an instruction, the assembler will generate an error. Several examples of immediate addressing are shown in Example 4-7.

Example 4-7: Immediate Addressing

PWRSAV #1	; Enter IDLE mode
ADD.B #0x10, W0	; Add 0x10 to W0 (byte mode)
Before Instruction: W0 = 0x12A9	
After Instruction: W0 = 0x12B9	
XOR W0, #1, [W1++]	; Exclusive-OR W0 and 0x1 ; Store the result to [W1] ; Post-increment W1
Before Instruction: W0 = 0xFFFF W1 = 0x0890 Data Memory 0x0890 = 0	0x0032
After Instruction: W0 = 0xFFFF W1 = 0x0892 Data Memory 0x0890 = 0	OxFFFE

4.1.5 Data Addressing Mode Tree

The Data Addressing modes of the dsPIC30F are summarized in Figure 4-1.



4.2 Program Addressing Modes

The dsPIC30F has a 23-bit Program Counter (PC). The PC addresses the 24-bit wide program memory to fetch instructions for execution, and it may be loaded in several ways. For byte compatibility with the Table Read and Table Write instructions, each instruction word consumes two locations in program memory. This means that during serial execution, the PC is loaded with PC+2.

Several methods may be used to modify the PC in a non-sequential manner, and both absolute and relative changes may be made to the PC. The change to the PC may be from an immediate value encoded in the instruction, or a dynamic value contained in a working register. When DO looping is active, the PC is loaded with the address stored in the DOSTART register, after the instruction at the DOEND address is executed. For exception handling, the PC is loaded with the address of the exception handler, which is stored in the interrupt vector table. When required, the software stack is used to return scope to the foreground process from where the change in program flow occurred.

Table 4-5 summarizes the instructions which modify the PC of the dsPIC30F. When performing function calls, it is recommended that RCALL be used instead of CALL, since RCALL only consumes 1 word of program memory.

Condition/Instruction	PC Modification	Software Stack Usage
Sequential Execution	PC = PC + 2	None
_{BRA Expr} (1) (Branch Unconditionally)	PC = PC + 2*Slit16	None
BRA Condition, Expr ⁽¹⁾ (Branch Conditionally)	PC = PC + 2 (condition false) PC = PC + 2*Slit16 (condition true)	None
CALL Expr ⁽¹⁾ (Call Subroutine)	PC = lit23	PC+4 is pushed on the stack ⁽²⁾
CALL Wn (Call Subroutine Indirect)	PC = Wn	PC+2 is pushed on the stack ⁽²⁾
_{GOTO Expr} (1) (Unconditional Jump)	PC = lit23	None
GOTO Wn (Unconditional Indirect Jump)	PC = Wn	None
_{RCALL Expr} (1) (Relative Call)	PC = PC + 2*Slit16	PC+2 is pushed on the stack ⁽²⁾
RCALL Wn (Computed Relative Call)	PC = PC + 2*Wn	PC+2 is pushed on the stack ⁽²⁾
Exception Handling	PC = address of the exception handler (read from vector table)	PC+2 is pushed on the stack ⁽³⁾
PC = Target REPEAT instruction (REPEAT Looping)	PC not modified (if REPEAT active)	None
PC = DOEND address (DO Looping)	PC = DOSTART (if Do active)	None

Table 4-5: Methods of Modifying Program Flow

Note 1: For BRA, CALL and GOTO, the Expr may be a label, absolute address, or expression, which is resolved by the linker to a 16-bit or 23-bit value (Slit16 or lit23). See Section 5. "Instruction Descriptions" for details.

- **2:** After CALL or RCALL is executed, RETURN or RETLW will pop the top-of-stack back into the PC.
- 3: After an exception is processed, RETFIE will pop the top-of-stack back into the PC.

4.3 Instruction Stalls

In order to maximize the data space EA calculation and operand fetch time, the X data space read and write accesses are partially pipelined. A consequence of this pipelining is that address register data dependencies may arise between successive read and write operations using common registers.

'Read After Write' (RAW) dependencies occur across instruction boundaries and are detected by the hardware. An example of a RAW dependency would be a write operation that modifies W5, followed by a read operation that uses W5 as an address pointer. The contents of 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, which allows the write to complete before the next read is started.

4.3.1 RAW Dependency Detection

During the instruction pre-decode, the core determines if any address register dependency is imminent across an instruction boundary. The stall detection logic compares the W register (if any) used for the destination EA of the instruction currently being executed with the W register to be used by the source EA (if any) of the pre-fetched instruction. When a match between the destination and source registers is identified, a set of rules are applied to decide whether or not to stall the instruction by one cycle. Table 4-6 lists various RAW conditions which cause an instruction execution stall.

Destination Address Mode Using Wn	Source Address Mode Using Wn	Stall Required ?		Examples (Wn = W2)
Direct	Direct	No Stall	ADD.W MOV.W	,
Indirect	Direct	No Stall	ADD.W MOV.W	., ,
Indirect	Indirect	No Stall	ADD.W MOV.W	., ,
Indirect	Indirect with pre/post-modification	No Stall	ADD.W MOV.W	-, , , , ,
Indirect with pre/post-modification	Direct	No Stall	ADD.W MOV.W	W0, W1, [W2++] W2, W3
Direct	Indirect	Stall ⁽¹⁾	ADD.W MOV.W	W0, W1, W2 [W2], W3
Direct	Indirect with pre/post-modification	Stall ⁽¹⁾	ADD.W MOV.W	W0, W1, W2 [W2++], W3
Indirect	Indirect	Stall ⁽¹⁾	ADD.W MOV.W	W0, W1, [W2](2) [W2], W3(2)
Indirect	Indirect with pre/post-modification	Stall ⁽¹⁾	ADD.W MOV.W	W0, W1, [W2](2) [W2++], W3 ⁽²⁾
Indirect with pre/post-modification	Indirect	Stall ⁽¹⁾	ADD.W MOV.W	W0, W1, [W2++] [W2], W3
Indirect with pre/post-modification	Indirect with pre/post-modification	Stall ⁽¹⁾	ADD.W MOV.W	W0, W1, [W2++] [W2++], W3

Table 4-6:	Raw Dependency	v Rules	Detection B	v Hardware)	1

Note 1: When stalls are detected, one cycle is added to the instruction execution time.

2: For these examples, the contents of W2 = the mapped address of W2 (0x0004).

4.3.2 Instruction Stalls and Exceptions

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

4.3.3 Instruction Stalls and Instructions that Change Program Flow

CALL and RCALL write to the stack using W15 and may, therefore, be subject to an instruction stall if the source read of the subsequent instruction uses W15.

GOTO, RETFIE and RETURN instructions are never subject to an instruction stall because they do not perform write operations to the working registers.

4.3.4 Instruction Stalls and DO/REPEAT Loops

Instructions operating in a DO or REPEAT loop are subject to instruction stalls, just like any other instruction. Stalls may occur on loop entry, loop exit and also during loop processing.

4.3.5 Instruction Stalls and PSV

Instructions operating in PSV address space are subject to instruction stalls, just like any other instruction. Should a data dependency be detected in the instruction immediately following the PSV data access, the second cycle of the instruction will initiate a stall. Should a data dependency be detected in the instruction immediately before the PSV data access, the last cycle of the previous instruction will initiate a stall.

Note: Refer to the dsPIC30F Family Reference Manual for more detailed information about RAW instruction stalls.

4.4 Byte Operations

Since the dsPIC30F data memory is byte addressable, most of the base instructions may operate in either Byte mode or Word mode. When these instructions operate in Byte mode, the following rules apply:

- all direct working register references use the Least Significant Byte of the 16-bit working register and leave the Most Significant Byte unchanged
- all indirect working register references use the data byte specified by the 16-bit address stored in the working register
- · all file register references use the data byte specified by the byte address
- · the Status Register is updated to reflect the result of the byte operation

It should be noted that data addresses are always represented as **byte** addresses. Additionally, the native data format is little-endian, which means that words are stored with the Least Significant Byte at the lower address, and the Most Significant Byte at the adjacent, higher address (as shown in Figure 4-2). Example 4-8 shows sample byte move operations and Example 4-9 shows sample byte math operations.

Note:	Instructions which operate in Byte mode must use the ".b" or ".B" instruction extension to specify a byte instruction. For example, the following two instructions are valid forms of a byte clear operation:
	CLR.b W0
	CLR.B WO

```
MOV.B #0x30, W0
                      ; move the literal byte 0x30 to W0
Before Instruction:
  W0 = 0x5555
After Instruction:
  W0 = 0x5530
MOV.B 0x1000, W0
                      ; move the byte at 0x1000 to W0
Before Instruction:
  W0 = 0x5555
  Data Memory 0x1000 = 0x1234
After Instruction:
  W0 = 0x5534
  Data Memory 0x1000 = 0x1234
                      ; byte move W0 to address 0x1001
MOV.B W0, 0x1001
Before Instruction:
   W0 = 0x1234
  Data Memory 0x1000 = 0x5555
After Instruction:
   W0 = 0x1234
   Data Memory 0x1000 = 0x3455
MOV.B W0, [W1++]
                      ; byte move W0 to [W1], then post-inc W1
Before Instruction:
  W0 = 0x1234
  W1 = 0x1001
  Data Memory 0x1000 = 0x5555
After Instruction:
  W0 = 0x1234
  W1 = 0x1002
  Data Memory 0x1000 = 0x3455
```

Example 4-8: Sample Byte Move Operations

```
Example 4-9: Sample Byte Math Operations
```

```
CLR.B [W6--]
                           ; byte clear [W6], then post-dec W6
Before Instruction:
   W6 = 0x1001
   Data Memory 0x1000 = 0x5555
After Instruction:
   W6 = 0 \times 1000
   Data Memory 0x1000 = 0x0055
SUB.B W0, #0x10, W1
                          ; byte subtract literal 0x10 from W0
                            ; and store to W1
Before Instruction:
   W0 = 0x1234
   W1 = 0xFFFF
After Instruction:
   W0 = 0x1234
   W1 = 0xFF24
ADD.B W0, W1, [W2++]
                          ; byte add W0 and W1, store to [W2]
                           ; and post-inc W2
Before Instruction:
  W0 = 0x1234
  W1 = 0x5678
  W2 = 0x1000
  Data Memory 0x1000 = 0x5555
After Instruction:
   W0 = 0x1234
   W1 = 0x5678
   W2 = 0x1001
   Data Memory 0x1000 = 0x55AC
```

4.5 Word Move Operations

Even though the dsPIC30F data space is byte addressable, all move operations made in Word mode must be word aligned. This means that for all source and destination operands, the Least Significant address bit must be '0'. If a word move is made to or from an odd address, an address error exception is generated. Likewise, all double-words must be word aligned. Figure 4-2 shows how bytes and words may be aligned in data memory. Example 4-10 contains several legal word move operations.

When an exception is generated due to a misaligned access, the exception is taken after the instruction executes. If the illegal access occurs from a data read, the operation will be allowed to complete, but the Least Significant bit of the source address will be cleared to force word alignment. If the illegal access occurs during a data write, the write will be inhibited. Example 4-11 contains several *illegal* word move operations.

-			
0x1001		b0	0x1000
0x1003	b1		0x1002
0x1005	b3	b2	0x1004
0x1007	b5	b4	0x1006
0x1009	b7	b6	0x1008
0x100B		b8	0x100A
Legend: b0 - byte stored at 0x1000 b1 - byte stored at 0x1003 b3:b2 - word stored at 0x1005:1004 (b2 is LSB) b7:b4 - double-word stored at 0x1009:0x1006 (b4 is LSB) b8 - byte stored at 0x100A			

Figure 4-2: Data Alignment in Memory

Note:	Instructions which operate in Word mode are not required to use an instruction extension. However, they may be specified with an optional ".w" or ".w" extension if desired. For example, the following instructions are valid forms of a word clear operation:	
	CLR	WO
	CLR.w	WO
	CLR.W	WO

```
Example 4-10: Legal Word Move Operations
```

```
#0x30, W0
MOV
                            ; move the literal word 0x30 to W0
Before Instruction:
   W0 = 0x5555
After Instruction:
   W0 = 0 \times 0030
MOV
        0x1000, W0
                           ; move the word at 0x1000 to W0
Before Instruction:
  W0 = 0x5555
   Data Memory 0x1000 = 0x1234
After Instruction:
   W0 = 0x1234
  Data Memory 0x1000 = 0x1234
MOV
        [WO], [W1++]
                          ; word move [W0] to [W1],
                            ; then post-inc W1
Before Instruction:
  W0 = 0x1234
  W1 = 0 \times 1000
  Data Memory 0x1000 = 0x5555
  Data Memory 0x1234 = 0xAAAA
After Instruction:
  W0 = 0x1234
  W1 = 0x1002
  Data Memory 0x1000 = 0xAAAA
  Data Memory 0x1234 = 0xAAAA
```

```
MOV
        0x1001, W0
                           ; move the word at 0x1001 to W0
Before Instruction:
 W0 = 0x5555
 Data Memory 0x1000 = 0x1234
 Data Memory 0x1002 = 0x5678
After Instruction:
 W0 = 0x1234
 Data Memory 0x1000 = 0x1234
 Data Memory 0x1002 = 0x5678
 ADDRESS ERROR TRAP GENERATED
 (source address is misaligned, so MOV is performed)
MOV
        W0, 0x1001
                          ; move W0 to the word at 0x1001
Before Instruction:
 W0 = 0x1234
 Data Memory 0x1000 = 0x5555
 Data Memory 0x1002 = 0x6666
After Instruction:
 W0 = 0x1234
 Data Memory 0x1000 = 0x5555
 Data Memory 0x1002 = 0x6666
 ADDRESS ERROR TRAP GENERATED
 (destination address is misaligned, so MOV is not performed)
                           ; word move [W0] to [W1],
MOV
        [WO], [W1++]
                           ; then post-inc W1
Before Instruction:
 W0 = 0x1235
 W1 = 0x1000
 Data Memory 0x1000 = 0x1234
 Data Memory 0x1234 = 0xAAAA
 Data Memory 0x1236 = 0xBBBB
After Instruction:
 W0 = 0x1235
 W1 = 0x1002
 Data Memory 0x1000 = 0xAAAA
 Data Memory 0x1234 = 0xAAAA
 Data Memory 0x1236 = 0xBBBB
 ADDRESS ERROR TRAP GENERATED
 (source address is misaligned, so MOV is performed)
```

Example 4-11: Illegal Word Move Operations

4.6 Using 10-bit Literal Operands

Several instructions which support Byte and Word mode have 10-bit operands. For byte instructions, a 10-bit literal is too large to use. So when 10-bit literals are used in Byte mode, the range of the operand must be reduced to 8-bits or the assembler will generate an error. Table 4-7 shows that the range of a 10-bit literal is 0:1023 in Word mode and 0:255 in Byte mode.

Instructions which employ 10-bit literals in Byte and Word mode are: ADD, ADDC, AND, IOR, RETLW, SUB, SUBB and XOR. Example 4-12 shows how positive and negative literals are used in Byte mode for the ADD instruction.

Literal Value	Word Mode	Byte Mode		
	kk kkkk kkkk	kkkk kkkk		
0	00 0000 0000	0000 0000		
1	00 0000 0001	0000 0001		
2	00 0000 0010	0000 0010		
127	00 0111 1111	0111 1111		
128	00 1000 0000	1000 0000		
255	00 1111 1111	1111 1111		
256	01 0000 0000	N/A		
512	10 0000 0000	N/A		
1023	11 1111 1111	N/A		

Table 4-7: 10-bit Literal Coding

Example 4-12: Using 10-bit Literals For Byte Operands

ADD.B #0x8	0, WO ;	add 128 (or -128) to W0	
ADD.B #0x3	80, WO ;	ERROR Illegal syntax for byte m	ode
ADD.B #0xF	'F, WO ;	add 255 (or -1) to W0	
ADD.B #0x3	FF, WO ;	ERROR Illegal syntax for byte m	ode
ADD.B #0xF	', WO ;	add 15 to WO	
ADD.B #0x7	'F, WO ;	add 127 to W0	
ADD.B #0x1	.00, W0 ;	ERROR Illegal syntax for byte m	ode

Note: Using a literal value greater than 127 in Byte mode is functionally identical to using the equivalent negative two's complement value, since the Most Significant bit of the byte is set. When operating in Byte mode, the Assembler will accept either a positive or negative literal value (i.e., #-10).

4.7 Software Stack Pointer and Frame Pointer

4.7.1 Software Stack Pointer

The dsPIC30F features a software stack which facilitates function calls and exception handling. W15 is the default Stack Pointer (SP) and after any RESET, it is initialized to 0x0800. This ensures that the SP will point to valid RAM in all dsPIC30F devices and permits stack availability for exceptions, which may occur before the SP is set by the user software. The user may reprogram the SP during initialization to any location within data space.

The SP always points to the first available free word (top-of-stack) and fills the software stack, working from lower addresses towards higher addresses. It pre-decrements for a stack pop (read) and post-increments for a stack push (write).

The software stack is manipulated using the PUSH and POP instructions. The PUSH and POP instructions are the equivalent of a MOV instruction, with W15 used as the destination pointer. For example, the contents of W0 can be pushed onto the top-of-stack (TOS) by

PUSH WO

This syntax is equivalent to

MOV W0, [W15++]

The contents of the TOS can be returned to W0 by

POP W0

This syntax is equivalent to

MOV [--W15],W0

During any CALL instruction, the PC is pushed onto the stack, such that when the subroutine completes execution, program flow may resume from the correct location. When the PC is pushed onto the stack, PC<15:0> is pushed onto the first available stack word, then PC<22:16> is pushed. When PC<22:16> is pushed, the Most Significant 7 bits of the PC are zero-extended before the push is made, as shown in Figure 4-3. During exception processing, the Most Significant 7 bits of the PC are concatenated with the lower byte of the Status Register (SRL) and IPL<3>, CORCON<3>. This allows the primary Status Register contents and CPU Interrupt Priority Level to be automatically preserved during interrupts.

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

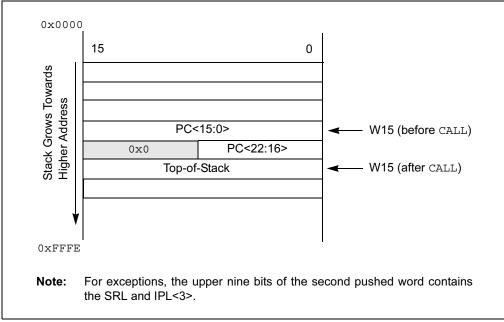
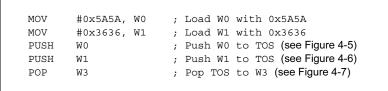


Figure 4-3: Stack Operation for CALL Instruction

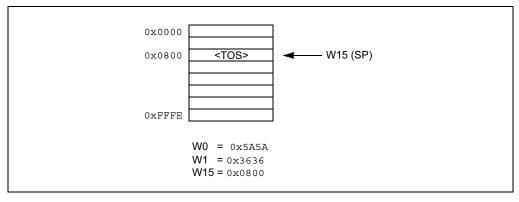
4.7.2 Stack Pointer Example

Figure 4-4 through Figure 4-7 show how the software stack is modified for the code snippet shown in Example 4-13. Figure 4-4 shows the software stack before the first PUSH has executed. Note that the SP has the initialized value of 0×0800 . Furthermore, the example loads $0 \times 5A5A$ and 0×3636 to W0 and W1, respectively. The stack is pushed for the first time in Figure 4-5 and the value contained in W0 is copied to TOS. W15 is automatically updated to point to the next available stack location, and the new TOS is 0×0802 . In Figure 4-6, the contents of W1 are pushed onto the stack, and the new TOS becomes 0×0804 . In Figure 4-7, the stack is popped, which copies the last pushed value (W1) to W3. The SP is decremented during the POP operation, and at the end of the example, the final TOS is 0×0802 .

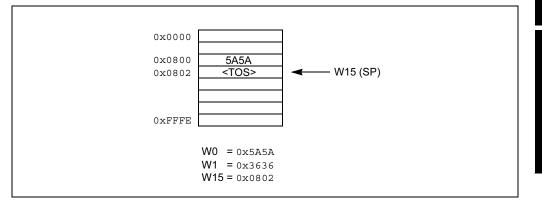
Example 4-13: Stack Pointer Usage











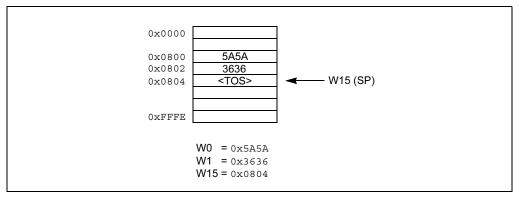
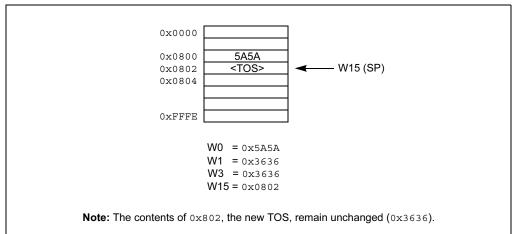


Figure 4-6: Stack Pointer After "PUSH W1" Instruction





4.7.3 Software Stack Frame Pointer

A stack frame is a user defined section of memory residing in the software stack. It is used to allocate memory for temporary variables which a function uses and one stack frame may be created for each function. W14 is the default Stack Frame Pointer (FP) and it is initialized to 0×0000 on any RESET. If the stack frame pointer is not used, W14 may be used like any other working register.

The link (LNK) and unlink (ULNK) instructions provide stack frame functionality. The LNK instruction is used to create a stack frame. It is used during a call sequence to adjust the SP, such that the stack may be used to store temporary variables utilized by the called function. After the function completes execution, the ULNK instruction is used to remove the stack frame created by the LNK instruction. The LNK and ULNK instructions must always be used together to avoid stack overflow.

4.7.4 Stack Frame Pointer Example

Figure 4-8 through Figure 4-10 show how a stack frame is created and removed for the code snippet shown in Example 4-14. Figure 4-8 shows the stack condition at the beginning of the example, before any registers are pushed to the stack. Here, W15 points to the first free stack location (TOS) and W14 points to a portion of stack memory allocated for the routine that is currently executing.

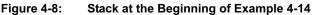
Before calling the function "COMPUTE", the parameters of the function (W0, W1 and W2) are pushed on the stack. After the "CALL COMPUTE" instruction is executed, the PC changes to the address of "COMPUTE" and the return address of the function "TASKA" is placed on the stack (Figure 4-9). Function "COMPUTE" then uses the "LNK #4" instruction to push the calling routine's frame pointer value onto the stack and the new frame pointer will be set to point to the current stack pointer. Then, the literal 4 is added to the stack pointer address in W15, which reserves memory for two words of temporary data (Figure 4-10).

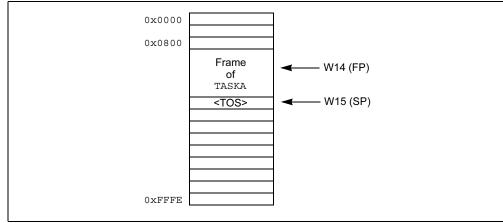
Inside the function "COMPUTE", the FP is used to access the function parameters and temporary (local) variables. [W14+n] will access the temporary variables used by the routine and [W14-n] is used to access the parameters. At the end of the function, the ULNK instruction is used to copy the frame pointer address to the stack pointer and then pop the calling subroutine's frame pointer back to the W14 register. The ULNK instruction returns the stack back to the state shown in Figure 4-9.

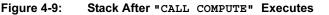
A RETURN instruction will return to the code that called the subroutine. The calling code is responsible for removing the parameters from the stack. The RETURN and POP instructions restore the stack to the state shown in Figure 4-8.

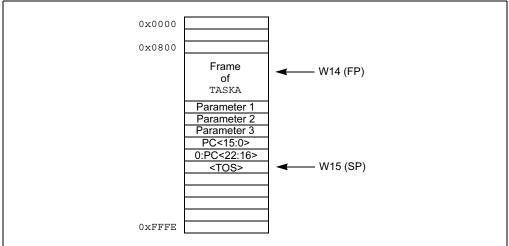
Example 4-14: Frame Pointer Usage

TASKA:			
PUSH	WO	;	Push parameter 1
PUSH	Wl	;	Push parameter 2
PUSH	W2	;	Push parameter 3
CALL	COMPUTE	;	Call COMPUTE function
POP	W2	;	Pop parameter 3
POP	Wl	;	Pop parameter 2
POP	WO	;	Pop parameter 1
COMPUTE:			
LNK	#4	;	Stack FP, allocate 4 bytes for local variables
		-	· · ·
ULNK		;	Free allocated memory, restore original FP
RETUR	N		Return to TASKA
		'	

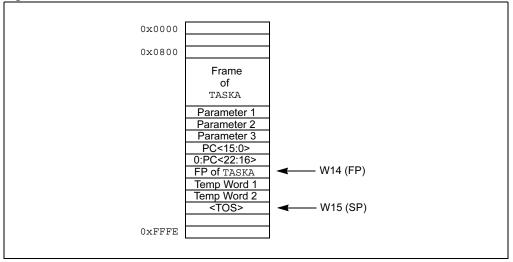












4.7.5 Stack Pointer Overflow

There is a stack limit register (SPLIM) associated with the stack pointer that is reset to 0x0000. SPLIM is a 16-bit register, but SPLIM<0> is fixed to '0', because all stack operations must be word aligned.

The stack overflow check will not be enabled until a word write to SPLIM occurs, after which time it can only be disabled by a device RESET. All effective addresses generated using W15 as a source or destination are compared against the value in SPLIM. Should the effective address be greater than the contents of SPLIM, then a stack error trap is generated.

If stack overflow checking has been enabled, a stack error trap will also occur if the W15 effective address calculation wraps over the end of data space (0xFFFF).

Refer to the dsPIC30F Family Reference Manual for more information on the stack error trap.

4.7.6 Stack Pointer Underflow

The stack is initialized to 0×0800 during RESET. A stack error trap will be initiated should the stack pointer address ever be less than 0×0800 .

Note: Locations in data space between 0x0000 and 0x07FF are, in general, reserved for core and peripheral special function registers.

4.8 Conditional Branch Instructions

Conditional branch instructions are used to direct program flow, based on the contents of the Status Register. These instructions are generally used in conjunction with a Compare class instruction, but they may be employed effectively after any operation that modifies the Status Register.

The compare instructions CP, CP0 and CPB, perform a subtract operation (minuend - subtrahend), but do not actually store the result of the subtraction. Instead, compare instructions just update the flags in the Status Register, such that an ensuing conditional branch instruction may change program flow by testing the contents of the updated Status Register. If the result of the Status Register test is true, the branch is taken. If the result of the Status Register test is false, the branch is not taken.

The conditional branch instructions supported by the dsPIC30F devices are shown in Table 4-8. This table identifies the condition in the Status Register which must be true for the branch to be taken. In some cases, just a single bit is tested (as in BRA C), while in other cases, a complex logic operation is performed (as in BRA GT). It is worth noting that both signed and unsigned conditional tests are supported, and that support is provided for DSP algorithms with the OA, OB, SA and SB condition mnemonics.

Condition Mnemonic ⁽¹⁾	Description	Status Test
С	Carry (not Borrow)	С
GE	Signed greater than or equal	(<u>N</u> && <u>OV</u>) (N&&OV)
GEU ⁽²⁾	Unsigned greater than or equal	С
GT	Signed greater than	$(\overline{Z}\&\&\overline{N}\&\&\overline{OV}) \parallel (\overline{Z}\&\&N\&\&OV)$
GTU	Unsigned greater than	C&&Z
LE	Signed less than or equal	Z (N&&OV) (N&&OV)
LEU	Unsigned less than or equal	<u>¯</u> C∥Z
LT	Signed less than	(N&&OV) (N&&OV)
LTU ⁽³⁾	Unsigned less than	C
N	Negative	Ν
NC	Not Carry (Borrow)	ō
NN	Not Negative	N
NOV	Not Overflow	<u>ov</u>
NZ	Not Zero	Z
OA	Accumulator A overflow	OA
ОВ	Accumulator B overflow	OB
OV	Overflow	OV
SA	Accumulator A saturate	SA
SB	Accumulator B saturate	SB
Z	Zero	Z

Table 4-8: Conditional Branch Instructions

Note 1: Instructions are of the form: BRA mnemonic, Expr.

2: GEU is identical to C and will reverse assemble to BRA C, Expr.

3: LTU is identical to NC and will reverse assemble to BRA NC, Expr.

Note: The "Compare and Skip" instructions (CPSEQ, CPSGT, CPSLT, CPSNE) do not modify the Status Register.

4.9 Z Status Bit

The Z status bit is a special zero status bit that is useful for extended precision arithmetic. The Z bit functions like a normal Z flag for all instructions, except those that use the carry/borrow input (ADDC, CPB, SUBB and SUBBR). For the ADDC, CPB, SUBB and SUBBR instructions, the Z bit can only be cleared and never set. If the result of one of these instructions is non-zero, the Z bit will be cleared and will remain cleared, *regardless of the result of subsequent ADDC, CPB, SUBB or SUBBR operations*. This allows the Z bit to be used for performing a simple zero check on the result of a series of extended precision operations.

A sequence of instructions working on multi-precision data (starting with an instruction with no carry/borrow input) will automatically logically AND the successive results of the zero test. All results must be zero for the Z flag to remain set at the end of the sequence of operations. If the result of the ADDC, CPB, SUBB or SUBBR instruction is non-zero, the Z bit will be cleared and remain cleared for all subsequent ADDC, CPB, SUBB or SUBBR instructions. Example 4-15 shows how the Z bit operates for a 32-bit addition. It shows how the Z bit is affected for a 32-bit addition implemented with an ADD/ADDC instruction sequence. The first example generates a zero result for only the MSWord, and the second example generates a zero result for both the LSWord and MSWord.



ADD WO, W	2, W4 ; Add LSWord and store to W4
	3, W5 ; Add MSWord and store to W5
- ,	
Boforo 32-bit A	ddition (zero result for MSWord):
W0 = 0x234	
W0 = 0x234 W1 = 0xFFF	
WI = 0XFFF $W2 = 0X39A$	-
$W_2 = 0 \times 39 A$ $W_3 = 0 \times 001$	
W3 = 0x001 W4 = 0x000	
W4 = 0x000 W5 = 0x000	
SR = 0x000	
After 32-bit Add	
W0 = 0x234	
W1 = 0xFFF	
W2 = 0x39A	
W3 = 0x001	
W4 = 0x5CE	
$W5 = 0 \times 000$	
SR = 0x020	1 (DC,C=1)
Refore 32-hit A	ddition (zero result for LSWord and MSWord):
$W0 = 0 \times B76$	
	В
$W1 = 0 \times FB7$	
W1 = 0xFB7 $W2 = 0x489$	2
$W1 = 0 \times FB7$	2 4
W1 = 0xFB7 W2 = 0x489 W3 = 0x048 W4 = 0x000	2 4 0
W1 = 0xFB7 W2 = 0x489 W3 = 0x048	2 4 0 0
W1 = 0xFB7 W2 = 0x489 W3 = 0x048 W4 = 0x000 W5 = 0x000 SR = 0x000	2 4 0 0 0
W1 = 0xFB7 W2 = 0x489 W3 = 0x048 W4 = 0x000 W5 = 0x000 SR = 0x000 After 32-bit Add	2 4 0 0 0 dition:
W1 = 0xFB7 W2 = 0x489 W3 = 0x048 W4 = 0x000 W5 = 0x000 SR = 0x000 After 32-bit Add W0 = 0xB76	2 4 0 0 0 dition: E
W1 = 0xFB7 W2 = 0x489 W3 = 0x048 W4 = 0x000 W5 = 0x000 SR = 0x000 After 32-bit Add W0 = 0xB76 W1 = 0xFB7	2 4 0 0 0 5 dition: E B
$\begin{array}{rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr$	2 4 0 0 0 5 dition: E B 2
	2 4 0 0 0 0 dition: E B 2 5
	2 4 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0

4.10 Assigned Working Register Usage

The 16 working registers of the dsPIC30F provide a large register set for efficient code generation and algorithm implementation. In an effort to maintain an instruction set that provides advanced capability, a stable run-time environment and backwards compatibility with earlier Microchip processor cores, some working registers have a pre-assigned usage. Table 4-9 summarizes these working register assignments, with details provided in subsections Section 4.10.1 "Implied DSP Operands" through Section 4.10.3 "PICmicro[®] Microcontroller Compatibility".

Register	Special Assignment
W0	Default WREG, Divide Quotient
W1	Divide Remainder
W2	"MUL f" Product Least Significant Word
W3	"MUL f" Product Most Significant Word
W4	MAC Operand
W5	MAC Operand
W6	MAC Operand
W7	MAC Operand
W8	MAC Pre-fetch Address (X Memory)
W9	MAC Pre-fetch Address (X Memory)
W10	MAC Pre-fetch Address (Y Memory)
W11	MAC Pre-fetch Address (Y Memory)
W12	MAC Pre-fetch Offset
W13	MAC Write Back Destination
W14	Frame Pointer
W15	Stack Pointer

Table 4-9: **Special Working Register Assignments**

4.10.1 **Implied DSP Operands**

To assist instruction encoding and maintain uniformity among the DSP class of instructions, some working registers have pre-assigned functionality. For all DSP instructions which have pre-fetch ability, the following 10 register assignments must be adhered to:

- · W4-W7 are used for arithmetic operands
- W8-W11 are used for pre-fetch addresses (pointers)
- · W12 is used for the pre-fetch register offset index
- W13 is used for the accumulator write back destination

These restrictions only apply to the DSP MAC class of instructions, which utilize working registers and have pre-fetch ability (described in Section 4.15 "DSP Accumulator Instructions"). The affected instructions are CLR, ED, EDAC, MAC, MOVSAC, MPY, MPY.N and MSC.

The DSP Accumulator class of instructions (described in Section 4.15 "DSP Accumulator Instructions") are not required to follow the working register assignments in Table 4-9 and may freely use any working register when required.

4.10.2 **Implied Frame and Stack Pointer**

To accommodate software stack usage, W14 is the implied frame pointer (used by the LNK and ULNK instructions) and W15 is the implied stack pointer (used by the CALL, LNK, POP, PUSH, RCALL, RETFIE, RETLW, RETURN, TRAP and ULNK instructions). Even though W14 and W15 have this implied usage, they may still be used as generic operands in any instruction, with the exceptions outlined in Section 4.10.1 "Implied DSP Operands". If W14 and W15 must be used for other purposes (it is strongly advised that they remain reserved for the Frame and Stack pointer), extreme care must be taken such that the run-time environment is not corrupted.

Details

4.10.3 PICmicro[®] Microcontroller Compatibility

4.10.3.1 Default Working Register WREG

To ease the migration path for users of the Microchip PICmicro families, the dsPIC30F has matched the functionality of the PICmicro instruction sets as closely as possible. One major difference between the dsPIC30F and the PICmicro processors is the number of working registers provided. The PICmicro families only provide one 8-bit working register, while the dsPIC30F provides sixteen, 16-bit working registers. To accommodate for the one working register of the PICmicro MCU, the dsPIC30F instruction set has designated one working register to be the default working register for all legacy file register instructions. The default working register is set to W0, and it is used by all instructions which use file register addressing.

Additionally, the syntax used by the dsPIC30F assembler to specify the default working register is similar to that used by the PICmicro assembler. As shown in the detailed instruction descriptions in **Section 5. "Instruction Descriptions**", "WREG" must be used to specify the default working register. Example 4-16 shows several instructions which use WREG.

Example 4-16: Using the Default Working Register WREG

ADD RAM100	; add	RAM100 and WREG, st	core in RAM100
ASR RAM100,	WREG ; shif	t RAM100 right, sto	ore in WREG
CLR.B WREG	; clea	r the WREG LS Byte	
DEC RAM100,	WREG ; decr	ement RAM100, store	e in WREG
MOV WREG, F	AM100 ; move	WREG to RAM100	
SETM WREG	; set	all bits in the WRN	EG
XOR RAM100	; XOR	RAM100 and WREG, st	core in RAM100

4.10.3.2 PRODH:PRODL Register Pair

Another significant difference between the Microchip PICmicro and dsPIC30F architectures is the multiplier. Some PICmicro families support an 8-bit x 8-bit multiplier, which places the multiply product in the PRODH:PRODL register pair. The dsPIC30F has a 17-bit x 17-bit multiplier, which may place the result into any two successive working registers (starting with an even register), or an accumulator.

Despite this architectural difference, the dsPIC30F still supports the legacy file register multiply instruction (MULWF) with the "MUL{.B} f" instruction (described on page 5-169). Supporting the legacy MULWF instruction has been accomplished by mapping the PRODH:PRODL registers to the working register pair W3:W2. This means that when "MUL{.B} f" is executed in Word mode, the multiply generates a 32-bit product which is stored in W3:W2, where W3 has the Most Significant Word of the product and W2 has the Least Significant Word of the product. When "MUL{.B} f" is executed in Byte mode, the 16-bit product is stored in W2, and W3 is unaffected. Examples of this instruction are shown in Example 4-17.

```
MUL.B 0x100
                  ; (0x100)*WREG (byte mode), store to W2
Before Instruction:
  W0 (WREG) = 0x7705
  W2 = 0x1235
  W3 = 0x1000
  Data Memory 0x0100 = 0x1255
After Instruction:
  W0 (WREG) = 0x7705
  W2 = 0x01A9
  W3 = 0 \times 1000
  Data Memory 0x0100 = 0x1255
MUL
        0x100
                  ; (0x100)*WREG (word mode), store to W3:W2
Before Instruction:
  W0 (WREG) = 0x7705
  W2 = 0x1235
  W3 = 0 \times 1000
  Data Memory 0x0100 = 0x1255
After Instruction:
  W0 (WREG) = 0x7705
  W2 = 0xDEA9
  W3 = 0x0885
  Data Memory 0x0100 = 0x1255
```

Example 4-17: Unsigned f and WREG Multiply (Legacy MULWF Instruction)

4.11 DSP Data Formats

4.11.1 Integer and Fractional Data

The dsPIC30F devices support both integer and fractional data types. Integer data is inherently represented as a signed two's complement value, where the Most Significant bit 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 $-32768 (0 \times 8000)$ to $32767 (0 \times 7FFF)$, including 0. For a 32-bit integer, the data range is $-2,147,483,648 (0 \times 8000 - 0000)$ to $2,147,483,647 (0 \times 7FFF)$ FFFF).

Fractional data is represented as a two's complement number, where the Most Significant bit is defined as a sign bit, and the radix point is implied to lie just after the sign bit. This format is commonly referred to as 1.15 (or Q15) format, where 1 is the number of bits used to represent the integer portion of the number, and 15 is the number of bits used to represent the fractional portion. 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 1.15 data range is -1.0 (0x8000) to 0.999969482 (0x7FFF), including 0.0 and it has a precision of 3.01518x10⁻⁵. In Normal Saturation mode, the 32-bit accumulators use a 1.31 format, which enhances the precision to 4.6566x10⁻¹⁰.

Super Saturation mode expands the dynamic range of the accumulators by using the 8 bits of the Upper Accumulator register (ACCxU) as guard bits. Guard bits are used if the value stored in the accumulator overflows beyond the 32^{nd} bit, and they are useful for implementing DSP algorithms. This mode is enabled when the **ACCSAT** bit (CORCON<4>), is set to '1' and it expands the accumulators to 40-bits. The accumulators then support an integer range of -5.498x10¹¹ (0x80 0000 0000) to 5.498x10¹¹ (0x7F FFFF FFFF). In Fractional mode, the guard bits of the accumulator do not modify the location of the radix point and the 40-bit accumulator, justified with a 1.31 radix point. As in Integer mode, the guard bits merely increase the dynamic range of the accumulator. 9.31 fractions have a range of -256.0 (0x80 0000 0000) to (256.0 - 4.65661x10⁻¹⁰) (0x7F FFFF FFFF). Table 4-10 identifies the range and precision of integers and fractions on the dsPIC30F devices for 16-bit, 32-bit and 40-bit registers.

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 devices use 1.31 format for the accumulators, a DSP multiply in Fractional mode also includes a left shift of 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 (e.g., $0.5 \times 0.5 = 0.25$).

Register Size	Integer Range	Fraction Range	Fraction Resolution
16-bit	-32768 to 32767	-1.0 to (1.0 – 2 ⁻¹⁵)	3.052 x 10 ⁻⁵
32-bit	-2,147,483,648 to 2,147,483,647	-1.0 to (1.0 – 2 ⁻³¹)	4.657 x 10 ⁻¹⁰
40-bit	-549,755,813,888 to 549,755,813,887	-256.0 to (256.0 – 2 ⁻³¹)	4.657 x 10 ⁻¹⁰

Table 4-10: dsPIC30F Data Ranges

4.11.2 Integer and Fractional Data Representation

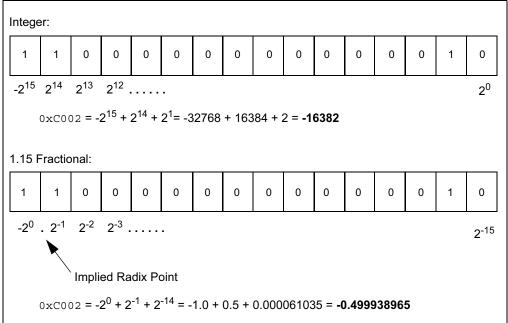
Having a working knowledge of how integer and fractional data are represented on the dsPIC30F is fundamental to working with the device. Both integer and fractional data treat the Most Significant bit as a sign bit, and the binary exponent decreases by one as the bit position advances towards the Least Significant bit. The binary exponent for an N-bit integer starts at (N-1) for the Most Significant bit, and ends at 0 for the Least Significant bit. For an N-bit fraction, the binary exponent starts at 0 for the Most Significant bit, and ends at (1-N) for the Least Significant bit. This is shown in Figure 4-11 for a positive value and in Figure 4-12 for a negative value.

Converting between integer and fractional representations can be performed using simple division and multiplication. To go from an N-bit integer to a fraction, divide the integer value by 2^{N-1} . Likewise, to convert an N-bit fraction to an integer, multiply the fractional value by 2^{N-1} .

Integer: ٥ Ο Ο 0 0 0 0 0 0 0 0 0 0 1 Ο 1 2^{15} 2^{14} 2¹³ 2¹² 20 $0 \times 4001 = 2^{14} + 2^0 = 16384 + 1 = 16385$ 1.15 Fractional: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 $-2^0 \cdot 2^{-1} \cdot 2^{-2} \cdot 2^{-3} \cdot \cdots \cdot$ 2-15 Implied Radix Point $0 \times 4001 = 2^{-1} + 2^{-15} = 0.5 + .000030518 = 0.500030518$

Figure 4-11: Different Representations of 0x4001







4.12 Accumulator Usage

Accumulators A and B are utilized by DSP instructions to perform mathematical and shifting operations. Since the accumulators are 40-bits wide and the X and Y data paths are only 16-bits, the method to load and store the accumulators must be understood.

Item A in Figure 4-13 shows that each 40-bit accumulator (ACCA and ACCB) consists of an 8-bit Upper register (ACCxU), a 16-bit High register (ACCxH) and a 16-bit Low register (ACCxL). To address the bus alignment requirement and provide the ability for 1.31 math, ACCxH is used as a destination register for loading the accumulator (with the LAC instruction), and also as a source register for storing the accumulator (with the SAC.R instruction). This is represented by Item B, Figure 4-13, where the upper and lower portions of the accumulator are shaded. In reality, during accumulator loads, ACCxL is zero backfilled and ACCxU is sign-extended to represent the sign of the value loaded in ACCxH.

When Normal (31-bit) Saturation is enabled, DSP operations (such as ADD, MAC, MSC, etc.) utilize solely ACCxH:ACCxL (Item C in Figure 4-13) and ACCxU is only used to maintain the sign of the value stored in ACCxH:ACCxL. For instance, when a MPY instruction is executed, the result is stored in ACCxH:ACCxL, and the sign of the result is extended through ACCxU.

When Super Saturation is enabled, all registers of the accumulator may be used (Item D in Figure 4-13) and the results of DSP operations are stored in ACCxU:ACCxH:ACCxL. The benefit of ACCxU is that it increases the dynamic range of the accumulator, as described in **Section 4.11.1 "Integer and Fractional Data"**. Refer to Table 4-10 to see the range of values which may be stored in the accumulator when in Normal and Super Saturation modes.

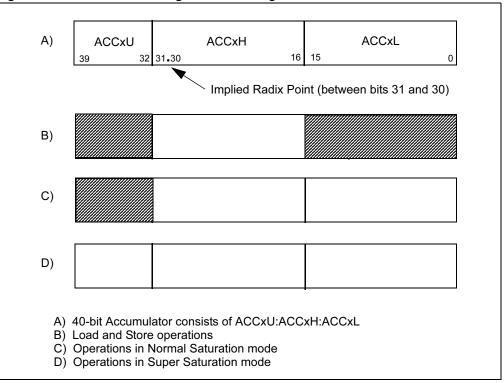


Figure 4-13: Accumulator Alignment and Usage

4.13 Accumulator Access

The six registers of Accumulator A and Accumulator B are memory mapped like any other special function register. This feature allows them to be accessed with file register or indirect addressing, using any instruction which supports such addressing. However, it is recommended that the DSP instructions LAC, SAC and SAC.R be used to load and store the accumulators, since they provide sign-extension, shifting and rounding capabilities. LAC, SAC and SAC.R instruction details are provided in **Section 5. "Instruction Descriptions"**.

Note: For convenience, ACCAU and ACCBU are sign-extended to 16-bits. This provides the flexibility to access these registers using either Byte or Word mode (when file register or indirect addressing is used).

4.14 DSP MAC Instructions

The DSP Multiply and Accumulate (MAC) operations are a special suite of instructions which provide the most efficient use of the dsPIC30F architecture. The DSP MAC instructions, shown in Table 4.14, utilize both the X and Y data paths of the CPU core, which enables these instructions to perform the following operations all in one cycle:

- two reads from data memory using pre-fetch working registers (MAC Pre-fetches)
- two updates to pre-fetch working registers (MAC Pre-fetch Register Updates)
- one mathematical operation with an accumulator (MAC Operations)

In addition, four of the ten DSP MAC instructions are also capable of performing an operation with one accumulator, while storing out the rounded contents of the alternate accumulator. This feature is called Accumulator Write Back (WB) and it provides flexibility for the software developer. For instance, the Accumulator WB may be used to run two algorithms concurrently, or efficiently process complex numbers, among other things.

Instruction	Description	Accumulator WB?
CLR	Clear accumulator	Yes
ED	Euclidean distance (no accumulate)	No
EDAC	Euclidean distance	No
MAC	Multiply and accumulate	Yes
MAC	Square and accumulate	No
MOVSAC	Move from X and Y bus	Yes
MPY	Multiply to accumulator	No
MPY	Square to accumulator	No
MPY.N	Negative multiply to accumulator	No
MSC	Multiply and subtract	Yes

 Table 4-11:
 DSP MAC Instructions

4.14.1 MAC Pre-Fetches

Pre-Fetches (or data reads) are made using the effective address stored in the working register. The two pre-fetches from data memory must be specified using the working registers assignments shown in Table 4-9. One read must occur from the X data bus using W8 or W9, and one read must occur from the Y data bus using W10 or W11. Allowable destination registers for both pre-fetches are W4-W7.

As shown in Table 4-3, one special Addressing mode exists for the MAC class of instructions. This mode is the Register Offset Addressing mode and utilizes W12. In this mode, the pre-fetch is made using the effective address of the specified working register, plus the 16-bit signed value stored in W12. Register Offset Addressing may only be used in the X space with W9, and in the Y-space with W11.

4.14.2 MAC Pre-Fetch Register Updates

After the MAC pre-fetches are made, the effective address stored in each pre-fetch working register may be modified. This feature enables efficient single cycle processing for data stored sequentially in X and Y memory. Since all DSP instructions execute in Word mode, only even numbered updates may be made to the effective address stored in the working register. Allowable address modifications to each pre-fetch register are -6, -4, -2, 0 (no update), +2, +4 and +6. This means that effective address updates may be made up to 3 words in either direction.

When the Register Offset Addressing mode is used, no update is made to the base pre-fetch register (W9 or W11), or the offset register (W12).

4.14.3 MAC Operations

The mathematical operations performed by the MAC class of DSP instructions center around multiplying the contents of two working registers and either adding or storing the result to either Accumulator A or Accumulator B. This is the operation of the MAC, MPY, MPY.N and MSC instructions. Table 4-9 shows that W4-W7 must be used for data source operands in the MAC class of instructions. W4-W7 may be combined in any fashion, and when the same working register is specified for both operands, a square or square and accumulate operation is performed.

For the ED and EDAC instructions, the same multiplicand operand *must* be specified by the instruction, because this is the definition of the Euclidean Distance operation. Another unique feature about this instruction is that the values pre-fetched from X and Y memory are not actually stored in W4-W7. Instead, only the *difference* of the pre-fetched data words is stored in W4-W7.

The two remaining MAC class instructions, CLR and MOVSAC, are useful for initiating or completing a series of MAC or EDAC instructions and do not use the multiplier. CLR has the ability to clear Accumulator A or B, pre-fetch two values from data memory and store the contents of the other accumulator. Similarly, MOVSAC has the ability to pre-fetch two values from data memory and store the contents of either accumulator.

4.14.4 MAC Write Back

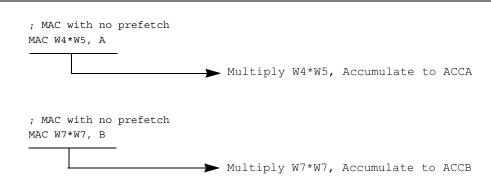
The write back ability of the MAC class of DSP instructions facilitates efficient processing of algorithms. This feature allows one mathematical operation to be performed with one accumulator, and the rounded contents of the other accumulator to be stored in the same cycle. As indicated in Table 4-9, register W13 is assigned for performing the write back, and two Addressing modes are supported: Direct and Indirect with Post-increment.

The CLR, MOVSAC and MSC instructions support accumulator write back, while the ED, EDAC, MPY and MPY.N instructions do not support accumulator write back. The MAC instruction, which multiplies two working registers which are not the same, also supports accumulator write back. However, the square and accumulate MAC instruction does not support accumulator write back (see Table 4.14).

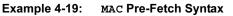
4.14.5 MAC Syntax

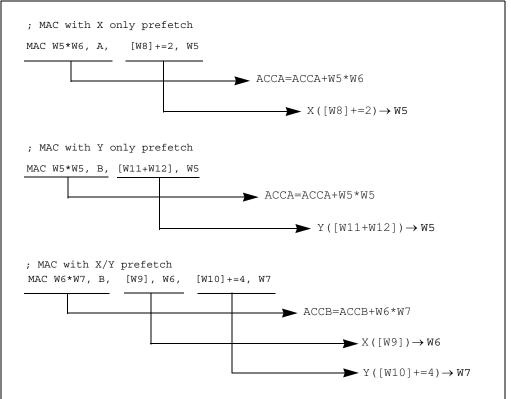
The syntax of the MAC class of instructions can have several formats, which depend on the instruction type and the operation it is performing, with respect to pre-fetches and accumulator write back. With the exception of the CLR and MOVSAC instructions, all MAC class instructions must specify a target accumulator along with two multiplicands, as shown in Example 4-18.



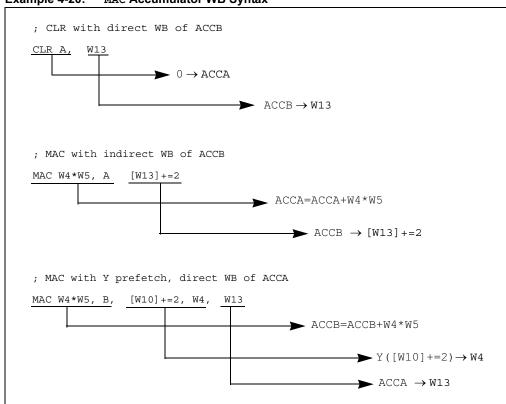


If a pre-fetch is used in the instruction, the assembler is capable of discriminating the X or Y data pre-fetch based on the register used for the effective address. [W8] or [W9] specifies the X pre-fetch and [W10] or [W11] specifies the Y pre-fetch. Brackets around the working register are required in the syntax, and they designate that indirect addressing is used to perform the pre-fetch. When address modification is used, it must be specified using a minus-equals or plus-equals "C"- like syntax (i.e., "[W8]-=2" or "[W8]+=6"). When Register Offset Addressing is used for the pre-fetch, W12 is placed inside the brackets ([W9+W12] for X pre-fetches and [W11+W12] for Y pre-fetches). Each pre-fetch operation must also specify a pre-fetch destination register (W4-W7). In the instruction syntax, the destination register appears before the pre-fetch are shown in Example 4-19.



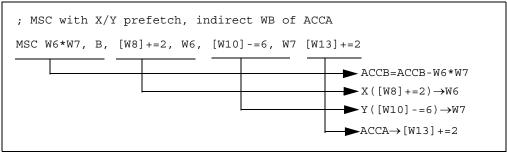


If an accumulator write back is used in the instruction, it is specified last. The write back must use the W13 register, and allowable forms for the write back are "W13" for direct addressing and "[W13]+=2" for indirect addressing with post-increment. By definition, the accumulator not used in the mathematical operation is stored, so the write back accumulator is **not** specified in the instruction. Legal forms of accumulator write back (WB) are shown in Example 4-20.



Putting it all together, an MSC instruction which performs two pre-fetches and a write back is shown in Example 4-21.

Example 4-21: MSC Instruction with Two Pre-Fetches and Accumulator Write Back



Example 4-20: MAC Accumulator WB Syntax

4.15 **DSP Accumulator Instructions**

The DSP Accumulator instructions do not have pre-fetch or accumulator WB ability, but they do provide the ability to add, negate, shift, load and store the contents of either 40-bit accumulator. In addition, the ADD and SUB instructions allow the two accumulators to be added or subtracted from each other. DSP Accumulator instructions are shown in Table 4-12 and instruction details are provided in Section 5. "Instruction Descriptions".

Instruction	Description	Accumulator WB?
ADD	Add accumulators	No
ADD	16-bit signed accumulator add	No
LAC	Load accumulator	No
NEG	Negate accumulator	No
SAC	Store accumulator	No
SAC.R	Store rounded accumulator	No
SFTAC	Arithmetic shift accumulator by Literal	No
SFTAC	Arithmetic shift accumulator by (Wn)	No
SUB	Subtract accumulators	No

Table 4-12: **DSP** Accumulator Instructions

4.16 Scaling Data with the FBCL Instruction

To minimize quantization errors that are associated with data processing using DSP instructions, it is important to utilize the complete numerical result of the operations. This may require scaling data up to avoid underflow (i.e., when processing data from a 12-bit ADC), or scaling data down to avoid overflow (i.e., when sending data to a 10-bit DAC). The scaling, which must be performed to minimize guantization error, depends on the dynamic range of the input data which is operated on, and the required dynamic range of the output data. At times, these conditions may be known beforehand and fixed scaling may be employed. In other cases, scaling conditions may not be fixed or known, and then dynamic scaling must be used to process data.

The FBCL instruction (Find First Bit Change Left) can efficiently be used to perform dynamic scaling, because it determines the exponent of a value. A fixed point or integer value's exponent represents the amount which the value may be shifted before overflowing. This information is valuable, because it may be used to bring the data value to "full scale", meaning that it's numeric representation utilizes all the bits of the register it is stored in.

The FBCL instruction determines the exponent of a word by detecting the first bit change starting from the value's sign bit and working towards the LSB. Since the dsPIC[™] device's barrel shifter uses negative values to specify a left shift, the FBCL instruction returns the negated exponent of a value. If the value is being scaled up, this allows the ensuing shift to be performed immediately with the value returned by FBCL. Additionally, since the FBCL instruction only operates on signed quantities, FBCL produces results in the range of -15:0. When the FBCL instruction returns '0', it indicates that the value is already at full scale. When the instruction returns -15, it indicates that the value cannot be scaled (as is the case with 0x0 and 0xFFFF). Table 4-13 shows word data with various dynamic ranges, their exponents, and the value after scaling each data to maximize the dynamic range. Example 4-22 shows how the FBCL instruction may be used for block processing.

Details

Word Value	Exponent	Full Scale Value (Word Value << Exponent)
x0001	14	0x4000
x0002	13	0x4000
x0004	12	0x4000
x0100	6	0x4000
x01FF	6	0x7FC0
x0806	3	0x4030
x2007	1	0x400E
x4800	0	0x4800
x7000	0	0x7000
x8000	0	0x8000
x900A	0	0x900A
xE001	2	0x8004
xFF07	7	0x8380

Table 4-13:Scaling Examples

Note: For the word values 0x0000 and 0xFFFF, the FBCL instruction returns -15.

As a practical example, assume that block processing is performed on a sequence of data with very low dynamic range stored in 1.15 fractional format. To minimize quantization errors, the data may be scaled up to prevent any quantization loss which may occur as it is processed. The FBCL instruction can be executed on the sample with the largest magnitude to determine the optimal scaling value for processing the data. Note that scaling the data up is performed by left shifting the data. This is demonstrated with the code snippet below.

Example 4-22: Scaling with FBCL

; assume W0 contains the la ; assume W4 points to the b ; assume the block of data	5 5
; determine the exponent to FBCL W0, W2 ;	5
	1 5
SCALE:	
SAC A, [W4++] ;	store scaled input (overwrite original)
; now process the data ; (processing block goes he	re)

4.17 Normalizing the Accumulator with the FBCL Instruction

The process of scaling a quantized value for its maximum dynamic range is known as normalization (the data in the third column in Table 4-13 contains normalized data). Accumulator normalization is a technique used to ensure that the accumulator is properly aligned before storing data from the accumulator, and the FBCL instruction facilitates this function.

The two 40-bit accumulators each have 8 guard bits from the AccU register, which expands the dynamic range of the accumulators from 1.31 to 9.31, when operating in Super Saturation mode (see **Section 4.11.1 "Integer and Fractional Data"**). However, even in Super Saturation mode, the Store Rounded Accumulator (SAC.R) instruction only stores 16-bit data (in 1.15 format) from AccH, as described in **Section 4.12 "Accumulator Usage"**. Under certain conditions, this may pose a problem.

Proper data alignment for storing the contents of the accumulator may be achieved by scaling the accumulator down if AccU is in use, or scaling the accumulator up if all of the AccH bits are not being used. To perform such scaling, the FBCL instruction must operate on the AccU byte and it must operate on the AccH word. If a shift is required, the ALU's 40-bit shifter is employed, using the SFTAC instruction to perform the scaling. Example 4-23 contains a code snippet for accumulator normalization.

Example 4-23: Normalizing with FBCL

	1	CA has just completed (SR intact) n super saturation mode
	-	b be the address of ACCAH (0x24)
MOV	#ACCAH, W5	; W5 points to ACCAH
BRA	OA, FBCL_GUARD	; if overflow we right shift
FBCL_HI:		
FBCL	[W5], WO	; extract exponent for left shift
BRA	SHIFT_ACC	; branch to the shift
FBCL_GUARD	:	
FBCL	[++W5], WO	; extract exponent for right shift
ADD.B	WO, #15, WO	; adjust the sign for right shift
SHIFT_ACC:		
SFTAC	A, WO	; shift ACCA to normalize

NOTES:



HIGHLIGHTS

This section of the manual contains the following major topics:

5.1	Instruction Symbols	5-2
5.2	Instruction Encoding Field Descriptors Introduction	5-2
5.3	Instruction Description Example	5-6
5.4	Instruction Descriptions	5-7

5.1 Instruction Symbols

All symbols used in Section 5.4 "Instruction Descriptions" are shown in Table 1-2.

5.2 Instruction Encoding Field Descriptors Introduction

All instruction encoding field descriptors used in **Section 5.4 "Instruction Descriptions"** are shown in Table 5.2 through Table 5-12.

Table 5-1: Instruction Encoding Field Descriptors

Field	Description	
A	Accumulator selection bit: 0=ACCA; 1=ACCB	
aa	Accumulator Write Back mode (see Table 5-12)	
В	Byte mode selection bit: 0=word operation; 1=byte operation	
bbbb	4-bit bit position select: 0000=LSB; 1111=MSB	
D	Destination address bit: 0=result stored in WREG;	
	1=result stored in file register	
dddd	Wd destination register select: 0000=W0; 1111=W15	
f ffff ffff ffff	13-bit register file address (0x0000 to 0x1FFF)	
fff ffff ffff ffff	15-bit register file word address (implied 0 LSB)	
	(0x0000 to 0xFFFE)	
ffff ffff ffff ffff	16-bit register file byte address (0x0000 to 0xFFFF)	
aaa	Register Offset Addressing mode for Ws source register	
	(see Table 5-4)	
hhh	Register Offset Addressing mode for Wd destination register	
	(see Table 5-5)	
iiii	Pre-Fetch X Operation (see Table 5-6)	
jjjj	Pre-Fetch Y Operation (see Table 5-8)	
k	1-bit literal field, constant data or expression	
kkkk	4-bit literal field, constant data or expression	
kk kkkk	6-bit literal field, constant data or expression	
kkkk kkkk	8-bit literal field, constant data or expression	
kk kkkk kkkk	10-bit literal field, constant data or expression	
kk kkkk kkkk kkkk	14-bit literal field, constant data or expression	
kkkk kkkk kkkk kkkk	16-bit literal field, constant data or expression	
mm	Multiplier source select with same working registers (see Table 5-10)	
mmm	Multiplier source select with different working registers	
	(see Table 5-11)	
nnnn nnnn nnnn nnn0	23-bit program address for CALL and GOTO instructions	
nnn nnnn		
nnnn nnnn nnnn nnnn	16-bit program offset field for relative branch/call instructions	
ppp	Addressing mode for Ws source register (see Table 5-2)	
ddd	Addressing mode for Wd destination register (see Table 5-3)	
rrrr	Barrel shift count	
SSSS	Ws source register select: 0000=W0; 1111=W15	
tttt	Dividend select, Most Significant Word	
vvvv	Dividend select, Least Significant Word	
W	Double-Word mode selection bit: 0=word operation;	
	1=double-word operation	
WWWW	Wb base register select: 0000=W0; 1111=W15	
XX	Pre-Fetch X Destination (see Table 5-7)	
xxxx xxxx xxxx xxxx	16-bit unused field (don't care)	
УУ	Pre-Fetch Y Destination (see Table 5-9)	
Z	Bit test destination: 0=C flag bit; 1=Z flag bit	

ррр	Addressing Mode	Source Operand
000	Register Direct	Ws
001	Indirect	[Ws]
010	Indirect with Post-Decrement	[Ws]
011	Indirect with Post-Increment	[Ws++]
100	Indirect with Pre-Decrement	[Ws]
101	Indirect with Pre-Increment	[++Ws]
11x	Unused	•

 Table 5-2:
 Addressing Modes for Ws Source Register

Table 5-3: Addressing Modes for Wd Destination Register

qqq	Addressing Mode	Destination Operand
000	Register Direct	Wd
001	Indirect	[Wd]
010	Indirect with Post-Decrement	[Wd]
011	Indirect with Post-Increment	[Wd++]
100	Indirect with Pre-Decrement	[Wd]
101	Indirect with Pre-Increment	[++Wd]
11x	Unused (an attempt to use this Addressing	mode will force a RESET instruction)

Table 5-4:	Offset Addressing Modes for W	/s Source Register (with Register Offset)

ggg	Addressing Mode	Source Operand
000	Register Direct	Ws
001	Indirect	[Ws]
010	Indirect with Post-Decrement	[Ws]
011	Indirect with Post-Increment	[Ws++]
100	Indirect with Pre-Decrement	[Ws]
101	Indirect with Pre-Increment	[++Ws]
11x	Indirect with Register Offset	[Ws+Wb]

Table 5-5: Offset Addressing Modes for Wd Destination Register (with Register Offset)

hhh	Addressing Mode	Source Operand
000	Register Direct	Wd
001	Indirect	[Wd]
010	Indirect with Post-Decrement	[Wd]
011	Indirect with Post-Increment	[Wd++]
100	Indirect with Pre-Decrement	[Wd]
101	Indirect with Pre-Increment	[++Wd]
11x	Indirect with Register Offset	[Wd+Wb]

Table 5-6: X Data Space Pre-Fetch Operation

iiii	Operation	
0000	Wxd=[W8]	
0001	Wxd=[W8], W8 = W8 + 2	
0010	Wxd=[W8], W8 = W8 + 4	
0011	Wxd=[W8], W8 = W8 + 6	
0100	No Pre-fetch for X Data Space	
0101	Wxd=[W8], W8 = W8 – 6	
0110	Wxd=[W8], W8 = W8 – 4	
0111	Wxd=[W8], W8 = W8 – 2	
1000	Wxd=[W9]	
1001	Wxd=[W9], W9 = W9 + 2	
1010	Wxd=[W9], W9 = W9 + 4	
1011	Wxd=[W9], W9 = W9 + 6	
1100	Wxd=[W9+W12]	
1101	Wxd=[W9], W9 = W9 – 6	
1110	Wxd=[W9], W9 = W9 – 4	
1111	Wxd=[W9], W9 = W9 – 2	

Table 5-7: X Data Space Pre-Fetch Destination

xx	Wxd
00	W4
01	W5
10	W6
11	W7

Table 5-8: Y Data Space Pre-Fetch Operation

jjjj	Operation
0000	Wyd=[W10]
0001	Wyd=[W10], W10 = W10 + 2
0010	Wyd=[W10], W10 = W10 + 4
0011	Wyd=[W10], W10 = W10 + 6
0100	No Pre-fetch for Y Data Space
0101	Wyd=[W10], W10 = W10 – 6
0110	Wyd=[W10], W10 = W10 – 4
0111	Wyd=[W10], W10 = W10 – 2
1000	Wyd=[W11]
1001	Wyd=[W11], W11 = W11 + 2
1010	Wyd=[W11], W11 = W11 + 4
1011	Wyd=[W11], W11 = W11 + 6
1100	Wyd=[W11+W12]
1101	Wyd=[W11], W11 = W11 – 6
1110	Wyd=[W11], W11 = W11 – 4
1111	Wyd=[W11], W11 = W11 – 2

Table 5-9: Y Data Space Pre-Fetch Destination

уу	Wyd
00	W4
01	W5
10	W6
11	W7

Table 5-10: MAC or MPY Source Operands (Same Working Register)

mm	Multiplicands	
00	W4 * W4	
01	W5 * W5	
10	W6 * W6	
11	W7 * W7	

Table 5-11: MAC or MPY Source Operands (Different Working Register)

mmm	Multiplicands
000	W4 * W5
001	W4 * W6
010	W4 * W7
011	Invalid
100	W5 * W6
101	W5 * W7
110	W6 * W7
111	Invalid

Table 5-12: MAC Accumulator Write Back Selection

aa	Write Back Selection
00	W13 = Other Accumulator (Direct Addressing)
01	[W13]+=2 = Other Accumulator (Indirect Addressing with Post-Increment)
10	No Write Back
11	Invalid

5.3 Instruction Description Example

The example description below is for the fictitious instruction FOO. The following example instruction was created to demonstrate how the table fields (syntax, operands, operation, etc.) are used to describe the instructions presented in **Section 5.4 "Instruction Descriptions"**.

FOO	The Header field summarizes what the instruction does
Syntax:	The Syntax field consists of an optional label, the instruction mnemonic, any optional extensions which exist for the instruction, and the operands for the instruction. Most instructions support more than one operand variant to support the various dsPIC30F Addressing modes. In these circumstances, all possible instruction operands are listed beneath each other (as in the case of op2a, op2b and op2c above). Optional operands are enclosed in braces.
Operands:	The Operands field describes the set of values which each of the operands may take. Operands may be accumulator registers, file registers, literal constants (signed or unsigned), or working registers.
Operation:	The Operation field summarizes the operation performed by the instruction.
Status Affected:	The Status Affected field describes which bits of the Status Register are affected by the instruction. Status bits are listed by bit position in descending order.
Encoding:	The Encoding field shows how the instruction is bit encoded. Individual bit fields are explained in the Description field, and complete encoding details are provided in Table 5.2.
Description:	The Description field describes in detail the operation performed by the instruction. A key for the encoding bits is also provided.
Words:	The Words field contains the number of program words that are used to store the instruction in memory.
Cycles:	The Cycles field contains the number of instruction cycles that are required to execute the instruction.
Examples:	The Examples field contains examples which demonstrate how the instruction operates. "Before" and "After" register snapshots are provided, which allow the user to clearly understand what operation the instruction performs.

5.4 Instruction Descriptions

ADD		Add f to WF	REG			
Syntax:	{label:}	ADD{.B}	f	{,WREG}		
Operands:	f ∈ [0 8	1911				
Operation:	-	$\Xi G) \rightarrow destina$	tion designat	ed by D		
Status Affected:	DC, N, O	-	tion debignat			
Encoding:	1011	0100	0BDf	fff	ffff	ffff
Description:	the file reg optional V specified,	ontents of the gister and plac VREG operand the result is st tored in the file	e the result in determines ored in WRE	n the destinat the destination	ion register. on register. If	The WREG is
	The 'D' bi	selects byte of selects the de select the ad	estination (01	for WREG, 1		
			a word opera rd operation,	tion. You may but it is not re	/ use a . w ex equired.	
Words:	1					
Cycles:	1					
Example 1	DD.B	RAM100	; Add	WREG to RA	M100 (Byte	mode)
WRE RAM10 S		7	After Instructio REG CC80 100 FF40 SR 0005]		
Example 2 A	DD	RAM200, WRE	G ; Add R	AM200 to W	REG (Word	mode)
WRE RAM20 SI	0 FFC0	WRI RAM2	00 FFC0	(C=1)		

ADD		Add Litera	l to Wn			
Syntax:	{label:}	ADD{.B}	#lit10,	Wn		
Operands:			e operation ord operation			
Operation:	lit10 + (Wn	$) \rightarrow Wn$				
Status Affected:	DC, N, OV	, Z, C				
Encoding:	1011	0000	0Bkk	kkkk	kkkk	dddd
Description:				nd to the conte k into the work		
	The 'k' bits	specify the l	iteral operand	ation (0 for wor I. working regist		
		rather than denote a wo For byte ope value [0:255	a word operation, ord operation, erations, the li oj. See Sectio	instruction de ation. You may but it is not re- teral must be : n 4.6 "Using 1 0-bit literal ope	y use a .w e quired. specified as a l 0-bit Literal (n unsigned Operands"
Words:	1					
Cycles:	1					
Example 1	ADD.B	#0xFF, W	17 ; Ad	d -1 to W7	(Byte mode)	
	Before Instructi W7 12C SR 000	on 0	After Instruct W7 12B SR 000	ion F		
Example 2	ADD	#0xFF, W	11 ; Ad	d 255 to W1	(Word mode	e)
	Before Instructi W1 12C SR 000	on 0	After Instruct W1 13B SR 000	ion F		

ADD		Add Wb to	Short Lite	eral		
Syntax:	{label:}	ADD{.B}	Wb,	#lit5,	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 lit5 ∈ [0 Wd ∈ [W0	. 31]				
Operation:	- (Wb) + lits					
Status Affected:	DC, N, O	/, Z, C				
Encoding:	0100	0www	wBqq	qddd	d11k	kkkk
Description:	operand a direct add	ind place the	result in the	e destination r	5-bit unsigned egister Wd. Re egister direct o	egister
	The 'B' bit The 'q' bit The 'd' bit	selects byte s select the d s select the a s provide the	or word op estination / ddress of tl literal opera	Address mode he destination and, a five-bit	word, 1 for byte	r.
	Note.	rather than	a word ope		ay use a .w e	
Words:	1					
Cycles:	1					
Example 1	ADD.B	W0, #0x1F	, W7		d 31 (Byte e result in	
	Before Instruction W0 2290 W7 1200 SR 0000		Afte Instruc W0 22 W7 12 SR 00	otion 90		
Example 2	ADD	W3, #0x6,	[W4]		nd 6 (Word m e result in	
Data (Data		Data Data	W4 0F 0FFE 60 1000 DD			

Instruction Descriptions

ADD		Add Wb	to Ws			
Syntax:	{label:}	ADD{.B}	Wb,	Ws,	Wd	
				[Ws],	[Wd]	
				[Ws++],	[Wd++]	
				[Ws],	[Wd]	
				[++Ws],	[++Wd]	
				[Ws],	[Wd]	
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wd ∈ [W0) W15]				
Operation:	(Wb) + (W	/s) → Wd				
Status Affected:	DC, N, 0	/, Z, C				
Encoding:	0100	0www	wBqq	qddd	dppp	SSSS
Description:	register W direct add addressin	/b and place ressing mind g may be i	ce the result in	i the destination r Wb. Either ro nd Wd.	the contents o on register Wd egister direct o	. Register
	The 'q' bit The 'd' bit The 'p' bit	s select th s select th s select th s select th	e destination A e address of th e source Addr e address of th	Address mode he destination ess mode. he source reg	register.	
		rather the		eration. You m	ay use a .we	
Words:	1					
Cycles:	1					
Example 1	ADD.B W	15, W6, W		dd W5 to W6 Byte mode)	, store res	ult in N
	Before		After			
	Instruction		Instructio	n 1		
	Instruction W5 AB00		Instruction W5 AB00	n] 		
	Instruction		Instruction W5 AB00	n 		
	Instruction W5 AB00 W6 0030		Instruction W5 AB00 W6 0030	n 		
Example 2	Instruction W5 AB00 W6 0030 W7 FFFF SR 00000	₹5, W6, V	Instruction W5 AB00 W6 0030 W7 FF30 SR 0000		, store res	ult in V
Example 2	Instruction W5 AB00 W6 0030 W7 FFFF SR 0000 ADD W Before	∛5, W6, V	Instruction W5 AB00 W6 0030 W7 FF30 SR 0000	dd W5 to W6 Word mode)	, store rest	ult in V
Example 2	Instruction W5 AB00 W6 0030 W7 FFFF SR 0000 ADD W Before Instruction	₹5, W6, V	Instruction W5 AB00 W6 0030 W7 FF30 SR 0000 77 ; A ; (After Instruction	dd W5 to W6 Word mode)	, store res	ult in V
Example 2	Instruction W5 AB00 W6 0030 W7 FFFF SR 0000 ADD W Before Instruction W5 AB00	√5, W6, P	Instruction W5 AB00 W6 0030 W7 FF30 SR 0000 I7 ; A ; (After Instruction W5 AB00	dd W5 to W6 Word mode)	, store res	ult in V
Example 2	Instruction W5 AB00 W6 0030 W7 FFFF SR 0000 ADD W Before Instruction	N5, W6, V	Instruction W5 AB00 W6 0030 W7 FF30 SR 0000 77 ; A ; (After Instruction	dd W5 to W6 Word mode)	, store res	ult in V

ADD		Add Accum	ulators			
Syntax:	{label:}	ADD	Acc			
Operands:	$Acc \in [A,B]$]				
Operation:	Else:): · (ACCB) → A · (ACCB) → A				
Status Affected:	OA, OB, O	AB, SA, SB, S	SAB			
Encoding:	1100	1011	A000	0000	0000	0000
Description:		ntents of Acc esult in the se tion.				
	The 'A' bit	specifies the	destination ad	cumulator.		
Words:	1					
Cycles:	1					
Example 1 AD	D Z	A	; Add A	CCB to ACC	A	
ACC/ ACCE SF	3 00 1833	etion 2 3300	ACCA ACCB SR	After Instructio 00 1855 7 000 1833 4	858	
Example 2 ADI) B		; Assume :	A to ACCB Super Satu: =1, SATA=1	ration mode , SATB=1)	e enable
ACCA ACCB SR	00 7654	tion 2222		After Instruction 0 E111 222 1 5765 543 480	2	=1)

© 2003 Microchip Technology Inc.

Syntax:	{label:	}	ADD	Ws,	{#Slit4,}	Acc	
o y maxi	labon	, I		[Ws],	[" 01111,]	100	
				[Ws++],			
				[Ws],			
				[Ws],			
				[++Ws],			
				[Ws+Wb],			
Operands:	Wb \in	[W0 [W0 : [-8 [A,B]	W15]				
Operation:	Shift _{Sli}	_{it4} (Exte	end(Ws)) + (A	Acc) \rightarrow Acc			
Status Affected	d: OA, O	B, OAE	3, SA, SB, S	AB			
Encoding:	110	00	1001	Awww	wrrr	rggg	SSSS
				s of a working	iequster or a	n effective ad	uiess. 116
	sign-e: The va	xtendir alue ad	ng and zero b	o the Most Sig backfilling the s ccumulator ma	gnificant Wor source opera	d of the accur nd prior to the	e operatior
	sign-e: The va literal I The 'A The 'w The 'r' The 'g	xtendir alue ad before d bit sp d bits s bits er d bits se	ng and zero b ded to the ad the addition ecifies the do pecify the of noode the op elect the sou	o the Most Sig backfilling the s ccumulator ma is made. estination acci fset register W	gnificant Wor source opera ay also be sh umulator. /b. node.	d of the accur nd prior to the	e operation
	sign-e: The va literal I The 'A The 'w The 'r' The 'g	xtendir alue ad before ' bits s ' bits er ' bits se ' bits se ' bits se ' bits se alue au	ng and zero b ded to the ac the addition ecifies the do pecify the of neode the op elect the sou becify the so ositive value nd negative	o the Most Sig backfilling the ccumulator ma is made. estination accu fset register W tional shift. rce Address n	gnificant Wor source opera ay also be sh umulator. /b. node. Vs. Slit4 represer and Slit4 rep	d of the accu nd prior to the ifted by a 4-b nt an arithme resent an arit	e operation it signed tic shift rig thmetic sh
Words:	sign-e The va literal I The 'A The 'w The 'r' The 'g The 's	xtendir alue ad before ' bits s ' bits er ' bits se ' bits se ' bits se ' bits se alue au	ng and zero b ded to the ac the addition ecifies the do pecify the of neode the op elect the sou becify the so ositive value nd negative	o the Most Sig backfilling the ccumulator mais made. estination accu fset register W tional shift. rce Address n urce register V s of operand s values of oper	gnificant Wor source opera ay also be sh umulator. /b. node. Vs. Slit4 represer and Slit4 rep	d of the accu nd prior to the ifted by a 4-b nt an arithme resent an arit	e operation it signed tic shift rig thmetic sh
	sign-e. The va literal I The 'A The 'w The 'r' The 'g The 's Not	xtendir alue ad before ' bits s ' bits er ' bits se ' bits se ' bits se ' bits se alue au	ng and zero b ded to the ac the addition ecifies the do pecify the of neode the op elect the sou becify the so ositive value nd negative	o the Most Sig backfilling the ccumulator mais made. estination accu fset register W tional shift. rce Address n urce register V s of operand s values of oper	gnificant Wor source opera ay also be sh umulator. /b. node. Vs. Slit4 represer and Slit4 rep	d of the accu nd prior to the ifted by a 4-b nt an arithme resent an arit	e operation it signed tic shift rig thmetic sh
	sign-e. The va literal I The 'A The 'w The 'r' The 'g The 's Not 1	xtendir alue ad before ' bits s ' bits er ' bits se ' bits se ' bits se ' bits se alue au	ng and zero b ded to the ac the addition ecifies the do pecify the of ncode the op elect the sou becify the so ositive value nd negative v ft. The conte	o the Most Sig backfilling the s ccumulator ma is made. estination accu fset register W tional shift. rce Address m urce register V s of operand s values of oper ents of the sou	gnificant Wor source opera ay also be sh umulator. /b. node. Vs. Slit4 represer and Slit4 rep rce register a	d of the accu nd prior to the ifted by a 4-b nt an arithme resent an arit	e operation it signed tic shift rig thmetic sh ed by Slit4
Words: Cycles: Example 1	sign-e. The va literal I The 'A The 'w The 'r' The 'g The 's Not 1	xtendir alue ad before ' bits s ' bits s ' bits s ' bits s e: Pr ar le W0, #:	ng and zero b ded to the ac the addition ecifies the do pecify the of noode the op elect the sou becify the so ositive value nd negative v ft. The conte	o the Most Sig backfilling the s ccumulator ma is made. estination accu fset register W tional shift. rce Address m urce register V s of operand s values of oper ents of the sou	gnificant Wor source opera ay also be sh umulator. /b. node. Vs. Slit4 represer and Slit4 rep rce register a cight-shift After	d of the accu nd prior to the ifted by a 4-b nt an arithme resent an ari are not affecte ced by 2 to	e operation it signed tic shift rig thmetic sh ed by Slit4
Cycles:	sign-e. The va literal I The 'A The 'w The 'r' The 'g The 's Not 1	xtendir alue ad before ' bits s ' bits s ' bits s ' bits s e: Po ar le	ng and zero b ded to the ac the addition ecifies the do pecify the of noode the op elect the sou becify the so ositive value nd negative v ft. The conte	o the Most Sig backfilling the s ccumulator ma is made. estination accu fset register W tional shift. rce Address m urce register V s of operand s values of oper ents of the sou	gnificant Wor source opera ay also be sh umulator. /b. hode. Vs. Slit4 represer and Slit4 rep rce register a cight-shift After Instructi	d of the accu nd prior to the ifted by a 4-b nt an arithme resent an ari are not affecte ced by 2 to	e operation it signed tic shift rig thmetic sh ed by Slit4

0000

SR

0000

SR

Example 2 ADD [W5++], A

; Add the effective value of W5 to ACCA ; Post-increment W5

Before Instruction					I	After nstruct	
W5			2000	W5			2002
ACCA	00 0	067	2345	ACCA	00	5067	2345
Data 2000			5000	Data 2000			5000
SR			0000	SR			0000

ADDC	Add f to WREG with Carry
Syntax:	{label:} ADDC{.B} f {,WREG}
Operands:	f ∈ [0 8191]
Operation:	(f) + (WREG) + (C) \rightarrow destination designated by D
Status Affected:	DC, N, OV, Z, C
Encoding:	1011 0100 1BDf ffff ffff ffff
Description:	Add the contents of the default working register WREG, the contents of the file register and the Carry bit and place the result in the destination register. The optional WREG operand determines the destination register. If WREG is specified, the result is stored in WREG. If WREG is not specified, the result is stored in the file register.
	The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 'D' bit selects the destination (0 for WREG, 1 for file register). The 'f' bits select the address of the file register.
	 Note 1: The extension . B in the instruction denotes a byte operation rather than a word operation. You may use a .w extension to denote a word operation, but it is not required. 2: The WREG is set to working register W0. 3: The Z flag is "sticky" for ADDC, CPB, SUBB and SUBBR. These instructions can only clear Z.
Words:	1
Cycles:	1
Example 1 AI	DDC.B RAM100 ; Add WREG and C bit to RAM100 ; (Byte mode)
WRE RAM10 S	0 8006 RAM100 8067
Example 2 A	DDC RAM200, WREG ; Add RAM200 and C bit to the WREG ; (Word mode)
WRE RAM20 S	0 3400 RAM200 3400

ADDC	Add Literal to Wn with Carry
Syntax:	{label:} ADDC{.B} #lit10, Wn
Operands:	lit10 \in [0 255] for byte operation lit10 \in [0 1023] for word operation Wn \in [W0 W15]
Operation:	lit10 + (Wn) + (C) \rightarrow Wn
Status Affected	DC, N, OV, Z, C
Encoding:	1011 0000 1Bkk kkkk kkkk dddd
Description:	Add the 10-bit unsigned literal operand, the contents of the working register Wn and the Carry bit and place the result back into the working register Wn.
	The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 'k' bits specify the literal operand. The 'd' bits select the address of the working register.
	 rather than a word operation. You may use a .w extension to denote a word operation, but it is not required. 2: For byte operations, the literal must be specified as an unsigned value [0:255]. See Section 2.7 "Using 10-bit Literal Operands" for information on using 10-bit literal operands in Byte mode. 3: The Z flag is "sticky" for ADDC, CPB, SUBB and SUBBR. These instructions can only clear Z.
Words:	1
Cycles:	1
Example 1	ADDC.B #0xFF, W7 ; Add -1 and C bit to W7 (Byte mode)
	Before After Instruction Instruction W7 12C0 W7 12BF SR 0000 (C=0) SR 0009 (N,C=1)
Example 2	ADDC #0xFF, W1 ; Add 255 and C bit to W1 (Word mode)
	Before After Instruction Instruction W1 12C0 W1 13C0 SR 0001 (C=1) SR 0000

ADDC		Add Wb to	Short Literal	with Carry		
Syntax:	{label:}	ADDC{.B}	Wb,	#lit5,	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 lit5 ∈ [0 Wd ∈ [W0	31]				
Operation:	=	$5 + (C) \rightarrow Wd$				
Status Affected:	, DC, N, O\	. ,				
Encoding:	0100	lwww	wBqq	qddd	d11k	kkkk
Description:	operand a Wd. Regis	ontents of the and the Carry b ster direct add ddressing may	bit and place the ressing must b	ne result in th be used for W	e destination	register
	The 'B' bit The 'q' bit The 'd' bit	ts select the ac selects byte of s select the de s select the ac s provide the li	or word operat estination Addi ddress of the d	ion (0 for wor ress mode. lestination reg	gister.	
		denote a wor The Z flag is	a word operat rd operation, b	ion. You may out it is not red DC,CPB, S	vuse a .we quired.	ktension 1
Words:	1					
Cycles:	1					
Example 1	ADDC.B	WO, #0x1F,			nd C bit (B sult in [W	-
۱ Data 12			After Instruct W0 CC W7 12 ta 12C0 B0 SR 00	ion 80 C0		
Example 2	ADDC	W3, #0x6, [d C bit (Wo sult in [
V Data 0FF Data 100	DDEE	Data	W4 0F a 0FFE 60 a 1000 DI			

Section	5.	Instruction	Descriptions
---------	----	-------------	--------------

ADDC		Add Wb to	Ws with Ca	ry		
Syntax:	{label:}	ADDC{.B}	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wd ∈ [W0) W15]				
Operation:	-	$(s) + (C) \rightarrow W$	d			
Status Affected:	DC, N, O	/, Z, C				
Encoding:	0100	lwww	wBqq	qddd	dppp	SSSS
	The 'q' bit The 'd' bit The 'p' bit	t selects byte of s select the de s select the act s select the so s select the act s select the set s set set set set set set set set set se	estination Add dress of the ource Address	dress mode. destination r s mode.	egister.	e).
		denote a wo The Z flag i	a word operat rd operation,	tion. You ma but it is not r ADDC, CE	y use a .w e equired.	xtension to
Words:	1					
Cycles:	1					
Example 1 ADD	OC.B WO	,[W1++],[W2+	; Sto		and C bit (ult in [W2] t W1, W2	_
W0 W1 W2 Data 0800 Data 1000	0800 1000 AB25	W W Data 080 Data 100	1 0801 2 1001 0 AB25			

SR

0000

SR

0001 (C=1)

ADDC

```
Example 2
```

; Add W3, [W2] and C bit (Word mode) ; Store the result in [W1] ; Post-increment W1, W2

I	Before Instruction Ir				
W1	1000		W1	1002	
W2	2000		W2	2002	
W3	0180		W3	0180	
Data 1000	8000		Data 1000	2681	
Data 2000	2500		Data 2000	2500	
SR	0001	(C=1)	SR	0000	

AND		AND f and	WREG			
Syntax:	{label:}	AND{.B}	f	{,WREG}		
On a new day	£ [0 0/	1041				
Operands:	f ∈ [0 8′	-				
Operation:		$(REG) \rightarrow de$	estination d	esignated by D		
Status Affected:	N, Z					1
Encoding:	1011	0110	OBDf	ffff on of the content	ffff	ffff
	the destination destination If WREG is	ation registe n register. If s not specifie	r. The optic WREG is s ed, the rest	of the file registe nal WREG oper- pecified, the result is stored in the	and determin ult is stored ir e file register.	es the 1 WREG.
	The 'D' bit	selects the	destination	peration (0 for wo (0 for WREG, 1 he file register.		
Mondor	2:	rather than denote a w	a word op ord operati	the instruction de eration. You ma on, but it is not re orking register V	y use a .w e equired.	
Words:	1					
Cycles:	1					
Example 1 AN	ID.B RAM1	.00	; AN	D WREG to RAM	1100 (Byte	mode)
	Before		Afte	r		
	Instruction		Instruc			
WREG RAM10			REG CC8			
SI		RAIN	1100 FF8 SR 000	08 (N=1)		
Example 2 An	ND RAM200	, WREG	; AN	ID RAM200 to V	VREG (Word	mode)
	Before		Afte			
	Instruction		Instruc			
WRE0 RAM20			REG 008			
RAM20 SI		RAN	1200 120 SR 000			
0			2			

AND		AND Liter	al and Wd			
Syntax:	{label:}	AND{.B}	#lit10,	Wn		
Operands:		. 255] for byt . 1023] for w W15]		n		
Operation:	lit10.AND.((Wn) → Wn				
Status Affected:	N, Z					
Encoding:	1011	0010	0Bkk	kkkk	kkkk	dddd
Description:	contents of	f the working	register Wn	and place th	literal operan e result back must be used	into the
	The 'k' bits	specify the	literal operar		vord, 1 for byt ister.	e).
	2:	For byte o unsigned va	perations, t lue [0:255].	See Section	required. ust be speci 4.6 "Using 1(10-bit literal o)-bit Literal
Words:	1					
Cycles:	1					
Example 1	AND.B #0x83	3, W7	; AND	0x83 to W7	(Byte mode	e)
	Before Instruction W7 12C0 SR 0000		After Instructio W7 1280 SR 0008			
Example 2	AND #0x333	, W1	; AND	0x333 to W	1 (Word mod	de)
	Before Instruction W1 12D0 SR 0000		After Instructio W1 0210 SR 0000			

AND		AND Wb a	nd Short Lite	eral		
Syntax:	{label:}	AND{.B}	Wb,	#lit5,	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 lit5 ∈ [0 Wd ∈ [W0	31]				
Operation: Status Affected:	(Wb).ANE N, Z	0.lit5 → Wd				
Encoding:	0110	0www	wBqq	qddd	d11k	kkkk
Description:	Wb and th Register o	e 5-bit literal	ID operation of and place the ing must be u	of the conten result in the sed for Wb.	ts of the base destination re Either register	gister Wd
	The 'B' bit The 'q' bit	is select the a selects byte s select the de s select the a	or word opera	ation (0 for w dress mode.	ord, 1 for byte	e).
	The 'k' bit Note:	s provide the The extensi rather than	literal operant on .	d, a five-bit ir instruction d tion. You ma	nteger number enotes a byte ly use a .w ex	e operatio
Words:	Note:	s provide the The extensi rather than	literal operant	d, a five-bit ir instruction d tion. You ma	nteger number enotes a byte ly use a .w ex	e operatio
Words: Cycles:		s provide the The extensi rather than	literal operan on . B in the a word opera	d, a five-bit ir instruction d tion. You ma	nteger number enotes a byte ly use a .w ex	operatio
Cycles:	Note: 1 1	s provide the The extensi rather than	literal operand on .B in the a word opera ord operation, -+] ; AND 1 ; Store	d, a five-bit ir instruction d tion. You ma but it is not r	nteger number enotes a byte y use a .w ex required. (Byte mode	e operatic ktension 1
Cycles:	Note: 1 1	s provide the The extensi rather than denote a wc	literal operand on .B in the a word opera ord operation, -+] ; AND 1 ; Store	d, a five-bit ir instruction d tion. You ma but it is not r but it is not r 0 and 0x3 to [W1] -increment	nteger number enotes a byte y use a .w ex required. (Byte mode	e operatic
Cycles: Example 1	Note: 1 1 AND.B W Before	s provide the The extensi rather than denote a wc	literal operand on .B in the a word opera ord operation, ++] ; AND t ; Store ; Post After	d, a five-bit ir instruction d tion. You ma but it is not r %0 and 0x3 e to [W1] -increment	nteger number enotes a byte y use a .w ex required. (Byte mode	operatic
Cycles: Example 1	Note: 1 1 AND.B W Before Instruction W0 23A5 W1 2211	s provide the The extensi rather than denote a wo	literal operand on .B in the a word operation, and operation, ; Store ; Post After Instructio W0 23A5 W1 2212	d, a five-bit ir instruction d tion. You ma but it is not r w0 and 0x3 e to [W1] -increment	nteger number enotes a byte y use a .w ex required. (Byte mode	operatic
Cycles: Example 1	Note: 1 1 AND.B W Before Instruction W0 23A5 W1 2211	s provide the The extensi rather than denote a wc	literal operand on .B in the a word operation, and operation, ; Store ; Post After Instructio W0 23A5 W1 2212	d, a five-bit ir instruction d tion. You ma but it is not r %0 and 0x3 e to [W1] -increment	nteger number enotes a byte y use a .w ex required. (Byte mode	operatic
Cycles: Example 1	Note: 1 1 AND.B W Before Instruction W0 23A5 W1 2211 210 9999	s provide the The extensi rather than denote a wo	literal operand on .B in the a word operation, and operation, ; AND (; Stord ; Post After Instruction W0 23A5 W1 2212 2210 0199 SR 0000 W1 ; AND	d, a five-bit ir instruction d tion. You ma but it is not r w0 and 0x3 e to [W1] -increment	nteger number enotes a byte y use a .w ex required. (Byte mode	e operatic ktension 1
Cycles: Example 1	Note: 1 1 AND.B W Before Instruction W0 23A5 W1 2211 210 9999 SR 0000 AND Before	Data 2	literal operand on .B in the a word operation, and operation, ; AND 0 ; Store ; Post After Instructio W0 23A5 W1 2212 2210 0199 SR 00000 V1 ; AND ; Store After After	d, a five-bit ir instruction d tion. You ma but it is not r %0 and 0x3 e to [W1] -increment n w0 and 0x1 w0 and 0x1	nteger number enotes a byte y use a .W ex required. (Byte mode W1	e operatic ktension (
Cycles: Example 1 2 Data 2 Example 2	Note: 1 1 AND.B W Before Instruction W0 23A5 W1 2211 210 9999 SR 0000 AND	Data 2	literal operand on .B in the a word operation, and operation, ; AND 0 ; Store ; Post After Instructio W0 23A5 W1 2212 2210 0199 SR 00000 V1 ; AND ; Store	d, a five-bit ir instruction d tion. You ma but it is not r %0 and 0x3 a to [W1] -increment n w0 and 0x1 re to W1	nteger number enotes a byte y use a .W ex required. (Byte mode W1	e operatio ktension
Cycles: Example 1 2 Data 2 Example 2	Note: 1 1 AND.B W Before Instruction W0 23A5 W1 2211 210 9999 SR 0000 AND Before Instruction	Data 2	literal operand on .B in the a word operation, and operation, ; AND 1 ; Store ; Post After Instructio W0 23A5 W1 2212 2210 0199 SR 0000 N1 ; AND ; Store After Instructio	d, a five-bit ir instruction d tion. You ma but it is not r %0 and 0x3 e to [W1] -increment n W0 and 0x1 re to W1	nteger number enotes a byte y use a .W ex required. (Byte mode W1	e operatic ktension (

AND		And Wb a	nd Ws			
Syntax:	{label:}	AND{.B}	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wd ∈ [W0) W15]				
Operation:	-	$D.(Ws) \rightarrow Wd$				
Status Affected:	N, Z					
Encoding:	0110	0www	wBqq	qddd	dppp	SSSS
	The 'B' bit The 'q' bit The 'd' bit The 'p' bit	ts select the a t selects byte s select the d s select the a s select the s s select the a	or word ope lestination Ad ddress of the ource Addre	ration (0 for v ddress mode e destination ss mode.	vord, 1 for byte register.	≥) .
	Note:	rather than	a word oper		denotes a byte ay use a .w e required.	
Words:	1					
Cycles:	1					
Example 1	AND.B	WO, W1 [W2	; sto	W0 and W1 re to [W2] t-incremen	(Byte mode))
Data	Before Instructio W0 AA55 W1 2211 W2 1001 1000 FFFF SR 0000	j ' Data	After Instruct W0 AAS W1 222 W2 100 1000 111 SR 000	ion 55 11 02 7F		

Example 2	AND	WO, [W1	++], W2	; store	W0 and [W1], and e to W2 (Word mode) -increment W1
	Before	Э		After	
	Instruct	ion	In	struction	
	WO AA5	5	W0	AA55	
	W1 100	0	W1	1002	
	W2 55A	A	W2	2214	
Data 1	1000 263	4 D	ata 1000	2634	
	SR 000	0	SR	0000	

ASR		Arithmetic	: Shift F	Right f			
Syntax:	{label:}	ASR{.B}	f	{	WREG}		
Operands: Operation:	f ∈ [0 8 [.] <u>For byte o</u>	-					
	(f<7>) - (f<7>) - (f<6:1>) (f<0>) - <u>For word c</u> (f<15>) (f<15>)	→ Dest<7> → Dest<6> → Dest<5:0 → C → Dest<15> → Dest<15> → Dest<14> >) → Dest<12					
Status Affected	, , -						
Encoding: Description:	1101	0101 ontents of the		BDf	fff	fff	ffff
	determine stored in V register. The 'B' bit The 'D' bit The 'f' bits	, the result is s the destina VREG. If WR selects byte selects the of select the ad The extens rather than denote a wo	tion reg EG is n or word lestinati ddress o ion .B i a word	operatio operatio on (0 for of the file n the ins operatior	REG is spi ed, the resi n (0 for wo WREG, 1 f register. truction de n. You may	ecified, the re ult is stored i rd, 1 for byte or file registe notes a byte use a . w ex	esult is n the file e). er). e operatior
	2:	The WREG					
Words:	1						
Cycles:	1						
Example 1	ASR.B RAM4	00, WREG		ASR RAM4 (Byte mo		tore to WF	REG
	Before Instruction REG 0600 M400 0823 SR 0000	WR RAM	Instr EG 400	fter uction 0611 0823 0001 (C	=1)		
Example 2	ASR RAM200)	;	ASR RAM	1200 (Wor	d mode)	
RA	Before Instruction M200 8009 SR 0000	RAM	Inst	After ruction C004 0009 (N	l, C=1)		

ASR	Arithmetic Shift Right Ws								
Syntax:	{label:}	ASR{.B}	Ws,	Wd					
			[Ws],	[Wd]					
			[Ws++],	[Wd++]					
			[Ws],	[Wd]					
			[++Ws],	[++Wd]					
			[Ws],	[Wd]					
Operands:	Ws ∈ [W0 Wd ∈ [W0								
Operation:	(Ws<7: (Ws<6: (Ws<0: <u>For word</u> (Ws<1: (Ws<1)	$\begin{array}{l} \rightarrow \rightarrow \forall d < 7 > \\ \rightarrow \forall d < 6 > \\ 1 > \rightarrow \forall d < 6 > \\ 1 >) \rightarrow \forall d < 5 : \\ \rightarrow \rightarrow C \\ \hline operation: \\ 5 >) \rightarrow \forall d < 15 \\ 5 >) \rightarrow \forall d < 14 \\ 4:1 >) \rightarrow \forall d < 2 > \\ \end{array}$	>						
Status Affected:	N, Z, C		I	1		1			
Encoding:	1101	0001	1Bqq	qddd	dppp	SSSS			
Description:	the result shifted int the result be used for The 'B' bi	in the destina o the Carry b is sign-extend or Ws and Wo t selects byte	ation register it of the Statu ded. Either re d. or word oper	ster Ws one bit Wd. The Least s Register. Afte gister direct or i ation (0 for wor	Significant bi r the shift is p ndirect addre	t of Ws is performed essing ma			
	The 'd' bit The 'p' bit	s select the s s select the a	ddress of the ource Addres ddress of the	destination reg	r.	e operatio			
		rather than	a word oper	ation. You may , but it is not rec	use a .we				
Words:	1								
vvolus.									

Example 1

ASR.B [W0++], [W1++]

; ASR [W0] and store to [W1] (Byte mode) ; Post-increment W0 and W1

I	Before nstructior	ı I	After nstructior	ı
W0	0600	W0	0601	
W1	0801	W1	0802	
Data 600	2366	Data 600	2366	
Data 800	FFC0	Data 800	33C0	
SR	0000	SR	0000	

Example 2

ASR W12, W13

; ASR W12 and store to W13 (Word mode)

I	Before Instruction					
W12	AB01					
W13	0322					
SR	0000					

After Instruction W12 AB01 W13 D580 SR 0009 (N, C=1)

ASR		Arithmetic	Shift Right b	by Short Li	erai	
Syntax:	{label:}	ASR	Wb,	#lit4,	Wnd	
Operands:	Wb ∈ [W0 lit4 ∈ [015] Wnd ∈ [W0 .	-				
Operation:		Vnd<15:15	5-Shift_Val+1> /nd<15-Shift_\			
Status Affected:	N, Z					
Encoding:	1101	1110	lwww	wddd	d100	kkkk
Description:	unsigned lite	ral and sto erformed, t	e contents of th re the result ir he result is sig nd.	n the destina	ation register	Wnd. Afte
	The 'd' bits s	elect the a	ddress of the ddress of the literal operanc	destination		
	Note: T	his instruc	tion operates i	n Word mo	de only.	
Words:	1					
Cycles:	1					
Example 1	ASR W0, #0x4	, W1	; ASR W	0 by 4 an	d store to	Wl
	Before Instruction W0 060F W1 1234 SR 0000		After Instruction W0 060F W1 0060 SR 0000			
Example 2	ASR W0, #0x6	, W1	; ASR WO) by 6 and	d store to	Wl
	Before Instruction W0 80FF W1 0060 SR 0000		After Instruction W0 80FF W1 FE03 SR 0008	(N=1)		
Example 3	ASR W0, #0xF	, W1	; ASR W	0 by 15 a	nd store to	W1
	Before Instruction W0 70FF W1 CC26 SR 0000		After Instruction W0 70FF W1 0000 SR 0002	(Z=1)		

Syntax:	{label:}	ASR	Wb,	Wns,	Wnd
	[,	,	
Operands:	Wb ∈ [W0 Wns ∈ [W Wnd ∈ [W	0W15]			
Operation:	Wb<15> -		5-Shift_Val+1 Vnd<15-Shift_		
Status Affected:	N, Z				
Encoding:	1101	1110	lwww	wddd	d000 ssss
Description:	Significant destination	t bits of Wns n register Wr	(up to 15 posind. After the s	itions) and hift is perfo	register Wb by the 4 Leas store the result in the prmed, the result is I for Wb, Wns and Wnd.
	The 'd' bit	s select the a	address of the address of the address of the	destination	n register.
		If Wns is g	ction operates greater than 1 FFFF if Wb is n	5, Wnd =	ode only. 0x0 if Wb is positive, ar
Words:	1				
Cycles:	1				
Example 1	ASR W0, WS	5, W6	; ASR N	W0 by W5	and store to W6
	Before		After		
	Instructior	1	Instructio	n T	
	WO 80FF		WO 80FF		
	W5 0004		W5 0004	-	
	W6 2633		W6 F80F		
	SR 0000		SR 0000		
Example 2	ASR W0, W	5, W6	; ASR	W0 by W5	and store to W6
	Before	_	After		
	Instruction W0 6688]	Instructio W0 6688	_	
	W5 000A		W5 000A	-	
	W6 FF00		W6 0019	_	
	SR 0000		SR 0000	_	
Example 3	ASR W11, W	W12, W13	; ASR	W11 by W1	12 and store to W13
	Before	_	After		
		ן ו		7	
	W11 8765 W12 88E4		W11 8765	-	
	N12 88E4 N13 A5A5		W12 88E4 W13 F876		
	CACA VII		101 10/0	1	

BCLR		Bit Clear f				
Syntax:	{label:}	BCLR{.B}	f,	#bit4		
Operands:	f ∈ [0 81 bit4 ∈ [0	91] for byte c 90] (even onl 7] for byte op 15] for byte c	ly) for word op peration	eration		
Operation:	$0 \rightarrow f \le 14$	>				
Status Affected:	None					
Encoding:	1010	1001	bbbf	ffff	ffff	fffb
Description:	with the Le	ast Significar	egister f specif nt bit (bit 0) an ons, bit 15 for	d advances t	the Most S	
			bit4 of the bit dress of the fi		e cleared.	
		denote a wor When this in address mus	a word operati rd operation, t struction oper st be word alig nstruction ope nd 7.	out it is not re ates in Word ned.	equired. I mode, the f	ile register
Words:	1					
Cycles:	1					
Example 1 BC	LR.B 0x800	0, #0x7	; Clear	bit 7 in ()x800	
Data 080 SF		Data 08	After Instruction 600 666F SR 0000			
Example 2 BC	LR 0x400	0, #0xA	; Clear	bit 10 in	0x400	
Data 040 Sł		Data 04	After Instruction 00 A855 SR 0000			

BCLR	Bit Clear in Ws
Syntax:	{label:} BCLR{.B} Ws, #bit4 [Ws], [Ws++], [Ws], [++Ws], [Ws],
Operands:	Ws \in [W0 W15] bit4 \in [0 7] for byte operation bit4 \in [0 15] for word operation
Operation:	$0 \rightarrow Ws < bit4 >$
Status Affected:	None
Encoding:	1010 0001 bbbb 0B00 0ppp ssss
Description:	Clear the bit in register Ws specified by 'bit4'. Bit numbering begins with the Least Significant bit (bit 0) and advances to the Most Significant bit (bit 7 for byte operations, bit 15 for word operations). Register direct or indirect addressing may be used for Ws.
	The 'b' bits select value bit4 of the bit position to be cleared. The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 's' bits select the address of the source/destination register. The 'p' bits select the source Address mode.
	 Note 1: The extension .B in the instruction denotes a byte operation rather than a word operation. You may use a .W extension to denote a word operation, but it is not required. 2: When this instruction operates in Word mode, the source register address must be word aligned. 3: When this instruction operates in Byte mode, 'bit4' must be between 0 and 7.
Words:	1
Cycles:	1
Example 1	BCLR.B W2, #0x2 ; Clear bit 3 in W2
	BeforeAfterInstructionInstructionW2F234W2SR0000SR
Example 2	BCLR [W0++], #0x0 ; Clear bit 0 in [W0] ; Post-increment W0
Data 2	Before After Instruction Instruction W0 2300 W0 2302 2300 5607 Data 2300 5606 SR 0000 SR 0000

BRA	Bra	nch Uncond	itionally			
Syntax:	{label:} BRA	λ Exp	or			
Operands:	Expr may be a la Expr is resolved					+32767
Operation:	(PC+2) + 2*Slit1 NOP \rightarrow Instruction					
Status Affected:	None					
Encoding:	0011	0111	nnnn	nnnn	nnnn	nnnn
Description:	The program will of the branch is t branches up to 3 resolved by the l expression. After 2*Slit16, since th	he 2's compl 2K instructio inker from the the branch i	ement nu ns forward e supplied s taken, th	mber '2*Slit16 d or backward l label, absolu ne new addre	6', which sup d. The Slit16 ute address c ss will be (P	ports value is or C+2) +
	The 'n' bits are a offset from (PC+)		l that spe	cifies the num	ber of progra	am words
Words:	1					
Cycles:	2					
Example 1	002000 HERE: 002002 002004 002006 002008 002008 THERE: 00200C	BRA THERI	Ξ	; B	ranch to T	HERE
	Before Instruction PC 00 2000 SR 0000		PC SR	After Instruction		
Example 2	002000 HERE: 002002 002004 002006 002008 00200A THERE: 00200C	BRA THERE 	+0x2	; Br	anch to TH	ERE+0x2
	Before Instruction PC 00 2000 SR 0000		PC SR	After Instruction 00 200C 0000		
Example 3	002000 HERE: 002002 002004	BRA 0x1360	5	; Br	anch to 0x	1366
	Before Instruction PC 00 2000 SR 0000		PC SR	After Instruction 00 1366 0000		

Instruction Descriptions

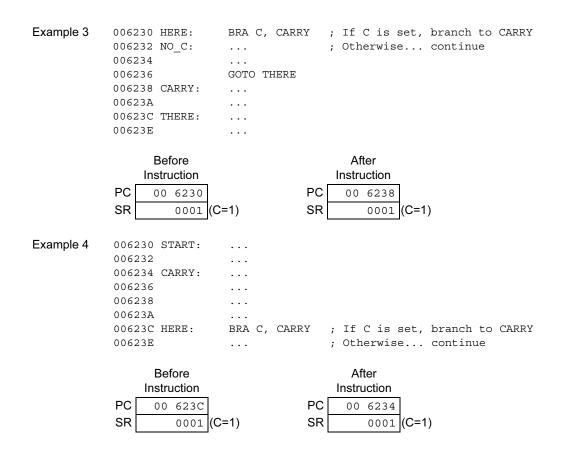
BRA		Computed E	Branch			
Syntax:	{label:}	BRA	Wn			
Operands:	Wn ∈ [W0	. W15]				
Operation:	(PC+2) + (2* NOP \rightarrow Insti	$Wn) \rightarrow PC$ ruction Regis	ter			
Status Affected:	None					
Encoding:	0000	0001	0110	0000	0000	SSSS
Description:	offset of the supports bra instruction e	branch is the inches up to a xecutes, the	unconditional sign-extende 32K instruction new PC will b h the next ins	ed 17-bit values forward one (PC+2)+2	ue (2*Wn), w or backward.	hich After this
	The 's' bits s	elect the add	lress of the se	ource registe	er.	
Words:	1					
Cycles:	2					
Example 1	002000 HERE: 002002 002106 002108 TABLE7 00210A	BRA W7		; Br	anch forwa	rd 2*W7
	Before Instruction PC 00 2000 W7 0084 SR 0000	-	PC W7 SR	After nstruction 00 2108 0084 0000		

BRA C	Ві	ranch if Ca	rry						
Syntax:	{label:} Bf	RA	С,	Expr					
Operands:	• •	pr may be a label, absolute address or expression. pr is resolved by the linker to a Slit16, where Slit16 \in [-32768 +32767].							
Operation:									
Status Affected	l: None								
Encoding:	0011	0001	nnnn	nnnn	nnnn	nnnn			
Description:	If the Carry flag The offset of th supports branc value is resolve expression. If the branch is	e branch is hes up to 3 ed by the lin taken, the	the 2's comp 2K instruction ker from the new address	blement numb ns forward or l supplied labe will be (PC+2	er '2*Slit16', backward. Th , absolute ac) + 2*Slit16, s	which ne Slit16 Idress or since the			
	PC will have in becomes a two The 'n' bits are	-cycle instr	uction, with a	NOP execute	d in the seco	nd cycle.			
	instruction wor	-		at specify the		0.2) 11			
Words:	1								
Cycles:	1 (2 if branch ta	aken)							
Example 1	002000 HERE: 002002 NO_C: 002004 002006 002008 CARRY: 00200A 00200C THERE: 00200E	BRA C, GOTO TI 	;	If C is se Otherwise.					
	Before			After					
	Instruction			nstruction					
	PC 00 2000 SR 0001	(C=1)	PC SR	00 2008 0001 (C	5=1)				
Example 2	002000 HERE: 002002 NO_C: 002004 002006 002008 CARRY: 00200A 00200C THERE: 00200E	BRA C, GOTO T 	;	If C is se Otherwise.					
	Ве	fore		Afi	er				
	Instruc	tion		Instructi	on				
	PC 00 2000 SR 0000		PC SR	00 2002					
	SR 0000			0000					

5

Instruction Descriptions

dsPIC30F Programmer's Reference Manual



BRA G	€ Β	ranch if Sig	gned Gre	ater Than or	Equal	
Syntax:	{label:} B	RA	GE,	Expr		
Operands:	Expr may be Expr is resolv Slit16 ∈ [-327	ed by the lir	nker to a	ress or expres Slit16, where	ssion.	
Operation:		N&&OV) (!N *Slit16 \rightarrow P struction Re	PC 29			
Status Affecte	ed: None		-			
Encoding:	0011	1101	nnnn	nnnn	nnnn	nnnn
	complement instructions for linker from the lf the branch the PC will ha	number '2*S prward or ba e supplied la is taken, the ave increme	Slit16', wh ackward. ⁻ abel, abso e new add nted to fe	ich supports b The Slit16 valu blute address tress will be (F tch the next in	the branch is t ranches up to ue is resolved or expression. PC+2) + 2*Slit1 struction. The	32K by the I6, since instructio
	cycle.	s a two-cyci	e instruct	ion, with a NO	executed in t	ne secon
	The 'n' bits ar instruction wo		gned liter	al that specify	the offset from	ו (PC+2) i
		ne assemble e used.	er will con	vert the specif	fied label into t	he offset
Words:	1					
Cycles:	1 (2 if branch	taken)				
Example 1	007600 LOOP: 007602 007604 007606 007608 HERE: 00760A NO_GE:	 BRA GE, 1	LOOP	-	, branch to wise con	
	Before			After		
	Instruction PC 00 7608		PC	Instruction		
	SR 0000		SR	0000		
Example 2	007600 LOOP: 007602 007604 007606 007608 HERE: 00760A NO GE:	 BRA GE, 1	LOOP		, branch to wise con	
	Before			After		
	Before Instruction			After Instruction		
	Before Instruction PC 00 7608	N=1)	PC SR	Instruction	(N=1)	

BRA G	EU		Branch if U	Jnsigned Gr	eater Than	or Equal	
Syntax:		{label:}	BRA	GEU,	Expr		
Operands:		Expr is res	be a label, al solved by the 32768 +32	linker to a S	lit16 offset th		an offset
Operation:							
Status Affected	:	None					
Encoding:		0011	0001	nnnn	nnnn	nnnn	nnnn
Description:		PC. The o which sup The Slit16	y flag is '1', t ffset of the b ports branch value is reso address or ex	ranch is the es up to 32K plved by the	2's complement instructions	ent number ' forward or b	2*Slit16', ackward.
		the PC wil	ch is taken, t l have incren then becom ond cycle.	nented to fet	ch the next ir	nstruction. Th	ne
		The 'n' bits in instructi	s are a 16-bit on words.	signed litera	l that specify	the offset fro	om (PC+2)
		Note:	Carry) instru		tical to the E as the same e t16.	-	•
Words:		1					
Cycles:		1 (2 if brar	nch taken)				
Example 1	00200 00200 00200 00200 00200	6 8 A C BYPASS:		, BYPASS ERE	; to	C is set, BYPASS erwise	
	PC SR	Before nstruction 00 2000 0001	(C=1)	PC SR	After nstruction 00 200C 0001 (C	C=1)	

BRA G	Г	Branch if	Signed Grea	ter Than		
Syntax:	{label:}	BRA	GT,	Expr		
Operands:		lved by the	linker to a S	ess or express lit16, where	sion.	
Operation:	lf (Conditior (PC+2) +			1&&!OV)		
Status Affected	None					
Encoding:	0011	1100	nnnn	nnnn	nnnn	nnnn
	instructions linker from t If the branct the PC will I	forward or he supplied n is taken, t nave incren	backward. T d label, abso the new addi nented to fet	which supports he Slit16 value lute address o ress will be (P0 ch the next ins	e is resolved r expression. C+2) + 2*Slit truction. The	by the 16, since instruction
	cycle. The 'n' bits a	are a 16-bit		on, with a NOP I that specify t		
	instruction v	vords.				
Words: Cycles:	1 1 (2 if branc	h taken)				
Example 1	002000 HERE: 002002 NO_GT: 002004 002006 002008 00200A 00200C BYPASS 00200E	 GOTO	T, BYPASS THERE		GT, branch erwise	
	Before Instruction PC 00 2000 SR 0001]	PC SR	After Instruction 00 200C 0001 (C	c=1)	

BRA GT	U	Branch if U	nsigned Gre	ater Than		
Syntax:	{label:}	BRA	GTU,	Expr		
Operands:	Expr is reso	e a label, ab olved by the l 2768 +327	inker to a Sli	ss or express 16, where	ion.	
Operation:						
Status Affected:	None					
Encoding:	0011	1110	nnnn	nnnn	nnnn	nnnn
	number '2*S or backward label, absol	Slit16', which d. The Slit16 ute address	supports bra value is resc or expressior		32K instruction nker from the	ns forward e supplied
	the PC will	have increme	ented to fetch	ss will be (PC a the next insi a, with a NOP	truction. The	instruction
	The 'n' bits offset from		literal that sp	pecifies the n	umber of inst	tructions
Words:	1					
Cycles:	1 (2 if brand	ch taken)				
	02000 HERE: 02002 NO_GTU: 02004 02006 02008 0200A 0200C BYPASS: 0200E	 GOTO TH	U, BYPASS IERE		U, branch wise cc	
-	Before Instruction PC 00 2000 SR 0001	(C=1)	Ins	After truction 0 200C 0001 (C=	1)	

BRA L	E i	Branch if S	igned Less [·]	Than or Equ	al	
Syntax:	{label:} [BRA	LE,	Expr		
Operands:		ved by the	solute addres linker to a Slit 767].	-	ion.	
Operation:			PC			
Status Affected	d: None					
Encoding:	0011	0100	nnnn	nnnn	nnnn	nnnn
Description:	program will 2's complem instructions f	branch rela ent number orward or b	tive to the ne r '2*Slit16', wl	xt PC. The of nich supports Slit16 value	is true, then t fset of the bra branches up is resolved by ssion.	anch is the to 32K
	PC will have becomes a t	incremente wo-cycle in:	ed to fetch the struction, with	next instruction a NOP exect	+2) + 2*Slit16 on. The instru ited in the sec	uction ther cond cycle
	The 'n' bits a offset from (F		l literal that sp	becifies the n	umber of instr	uctions
Words:	1					
Cycles:	1 (2 if branch	n taken)				
Example 1	002000 HERE: 002002 NO_LE: 002004 002006 002008 00200A 00200C BYPASS: 00200E	BRA LE GOTO I 	E, BYPASS THERE	BY	LE, branch PASS herwise	
	Before Instruction PC 00 2000 SR 0001	(C=1)	PC SR	After Instruction	C=1)	

BRA L	EU Branch if Unsigned Less Than or Equal
Syntax:	{label:} BRA LEU, Expr
Operands:	Expr may be a label, absolute address or expression. Expr is resolved by the linker to a Slit16, where Slit16 \in [-32768 +32767].
Operation:	Condition = $ C Z$ If (Condition) (PC+2) + 2*Slit16 \rightarrow PC NOP \rightarrow Instruction Register
Status Affecte	d: None
Encoding:	0011 0110 nnnn nnnn nnnn nnnn
Description:	If the logical expression $(!C Z)$ is true, then the program will branch relative to the next PC. The offset of the branch is the 2's complement number '2*Slit16', which supports branches up to 32K instructions forward or backward. The Slit16 value is resolved by the linker from the supplied label, absolute address or expression.
	If the branch is taken, the new address will be $(PC+2) + 2*Slit16$, since the PC will have incremented to fetch the next instruction. The instruction then becomes a two-cycle instruction, with a NOP executed in the second cycle.
	The 'n' bits are a signed literal that specifies the number of instructions offset from (PC+2).
Words:	1
Cycles:	1 (2 if branch taken)
Example 1	002000 HERE: BRA LEU, BYPASS ; If LEU, branch to BYPASS 002002 NO_LEU: ; Otherwise continue 002004 ; 002006 ; 002008 ; 00200A GOTO THERE ; 00200C BYPASS: 00200E ;
	Before After Instruction Instruction PC 00 2000 PC 00 200C SR 0001 (C=1) SR 0001 (C=1)

DS70030E-page 5-40

BRA L	Bra	anch if Sig	ned Less 1	ſhan		
Syntax:	{label:} BR	A L	.T,	Expr		
Operands:	Expr may be a Expr is resolve Slit16 ∈ [-3276	d by the lin	ker to a Slit	•	ion.	
Operation:	Condition = (N If (Condition) (PC+2) + 2^* NOP \rightarrow Inst	Slit16 \rightarrow PC)			
Status Affected	: None					
Encoding:	0011	0101	nnnn	nnnn	nnnn	nnnn
	will branch rela complement nu instructions for linker from the If the branch is the PC will hav	umber '2*SI ward or bac supplied la taken, the	it16', which ckward. The bel, absolu new addre:	supports bra e Slit16 value te address or ss will be (PC	anches up to e is resolved l r expression. C+2) + 2*Slit1	32K by the 6, since
	then becomes cycle.	a two-cycle	instruction	, with a NOP	executed in t	he second
	The 'n' bits are offset from (PC		erai triat sp			ructions
Words:	1					
Cycles:	1 (2 if branch t	aken)				
Example 1	002000 HERE: 002002 NO_LT: 002004 002006 002008 00200A	BRA LT, GOTO THI			T, branch rwise c	
	00200C BYPASS:					
	00200E			A.C.		
	Before Instruction		Ins	After truction		
	PC 00 2000			0 2002		
	SR 0001 (C	C=1)	SR	0001 (C=	=1)	

BRA L	Branch if Unsigned Less Than	
Syntax:	{label:} BRA LTU, Expr	
Operands:	Expr may be a label, absolute address or expression. Expr is resolved by the linker to a Slit16, where Slit16 \in [-32768 +32767].	
Operation:	Condition = !C If (Condition) (PC+2) + 2*Slit16 \rightarrow PC NOP \rightarrow Instruction Register	
Status Affecte	: None	
Encoding:	0011 1001 nnnn nnnn nnnn	n
Description:	If the Carry flag is '0', then the program will branch relative to the next F The offset of the branch is the 2's complement number '2*Slit16', whic supports branches up to 32K instructions forward or backward. The Sli value is resolved by the linker from the supplied label, absolute addres or expression.	:h t16
	If the branch is taken, the new address will be $(PC+2) + 2*Slit16$, since the PC will have incremented to fetch the next instruction. The instruct then becomes a two-cycle instruction, with a NOP executed in the seco cycle.	ion
	The 'n' bits are a signed literal that specifies the number of instructions offset from (PC+2).	s
	Note: This instruction is identical to the BRA_NC, Expr (Branch if Carry) instruction and has the same encoding. It will reveassemble as BRA_NC, Slit16.	
Words:	1	
Cycles:	1 (2 if branch taken)	
Example 1	002000 HERE: BRA LTU, BYPASS ; If LTU, branch to BY 002002 NO_LTU: 002004 002006 002008 002000 BYPASS: 002002 NO_LTU:	
	Before Instruction After Instruction PC 00 2000 SR 0001 (C=1)	

BRA N		Branch if	Negative			
Syntax:	{label:}	BRA	N,	Expr		
Operands:		lved by the	e linker to a S	ess or express lit16, where	ion.	
Operation:						
Status Affected	: None					
Encoding:	0011	0011	nnnn	nnnn	nnnn	nnnn
Description:	PC. The offs which suppo	et of the b orts branch is resolved	pranch is the les up to 32K d by the linke	program will bra 2's complemer instructions fo r from the sup	nt number '2*: rward or back	Slit16', ward. The
	PC will have	incremen	ted to fetch th	ess will be (PC ne next instruct ith a NOP exect	ion. The instru	uction ther
	The 'n' bits a offset from (ed literal that	specifies the n	umber of inst	ructions
Words:	1					
Cycles:	1 (2 if branc	h taken)				
Example 1	002000 HERE: 002002 NO_N: 002004 002006 002008 00200A 00200C BYPASS: 00200E	 GOTO	N, BYPASS		N, branch t	
	Before Instruction PC 00 2000 SR 0008	(N=1)	PC SR	After Instruction 00 200C 0008 (N	J=1)	

BRA N	Branch if Not Carry
Syntax:	{label:} BRA NC, Expr
Operands:	Expr may be a label, absolute address or expression. Expr is resolved by the linker to a Slit16, where Slit16 \in [-32768 +32767].
Operation:	Condition = !C If (Condition) (PC+2) + 2*Slit16 \rightarrow PC NOP \rightarrow Instruction Register
Status Affecte	ed: None
Encoding:	0011 1001 nnnn nnnn nnnn nnnn
	 The offset of the branch is the 2's complement number '2*Slit16', which supports branches up to 32K instructions forward or backward. The Slit16 value is resolved by the linker from the supplied label, absolute address or expression. If the branch is taken, the new address will be (PC+2) + 2*Slit16, since the PC will have incremented to fetch the next instruction. The instruction then becomes a two-cycle instruction, with a NOP executed in the second cycle.
	The 'n' bits are a signed literal that specifies the number of instructions offset from (PC+2).
Words:	1
Cycles:	1 (2 if branch taken)
Example 1	002000 HERE: BRA NC, BYPASS ; If NC, branch to BYPASS 002002 NO_NC: ; Otherwise continue 002004 ; 002006 ; 002008 ; 00200A GOTO THERE ; 00200C BYPASS: 00200E ;
	Before Instruction After Instruction PC 00 2000 0001 (C=1) PC 00 2002 0001 (C=1)

BRA N	IN		Branch if	Not Negativ	/e		
Syntax:		{label:}	BRA	NN,	Expr		
Operands:		Expr is res		e linker to a	ress or express Slit16, where	sion.	
Operation:							
Status Affect	ed:	None					
Encoding:		0011	1011	nnnn	nnnn	nnnn	nnnn
		Slit16 valu address o If the bran	ie is resolve r expressior ch is taken,	d by the link n. the new add	C instructions for er from the sup ress will be (Po tch the next ins	plied label, al C+2) + 2*Slit1	bsolute I6, since
		cycle.			on, with a NOP		
		offset from	0		1		
Words:		1					
Cycles:		1 (2 if brar	nch taken)				
Example 1	0020 0020 0020 0020 0020	06 08 0A 0C BYPASS	 GOTO	N, BYPASS THERE		, branch to	
	PC SR	Before Instruction 00 2000 0000			After struction 00 200C 0000		

Syntax:	{label:} BRA NOV, Expr
Operands:	Expr may be a label, absolute address or expression. Expr is resolved by the linker to a Slit16, where Slit16 \in [-32768 +32767].
Operation:	Condition = !OV If (Condition) (PC+2) + 2*Slit16 \rightarrow PC NOP \rightarrow Instruction Register
Status Affecte	d: None
Encoding:	0011 1000 nnnn nnnn nnnn nnnn
	Slit16 value is resolved by the linker from the supplied label, absolute address or expression. If the branch is taken, the new address will be (PC+2) + 2*Slit16, since the PC will have incremented to fetch the next instruction. The instruction ther becomes a two-cycle instruction, with a NOP executed in the second cycle
	The 'n' bits are a signed literal that specifies the number of instructions offset from (PC+2).
Words:	1
Cycles:	1 (2 if branch taken)
Example 1	002000 HERE: BRA NOV, BYPASS ; If NOV, branch to BYPAS 002002 NO_NOV: ; Otherwise continue 002004 002006 002008 00200C BYPASS: 00200E
	Before Instruction After Instruction PC 00 2000 SR PC 00 2000 0008 (N=1) SR 0008

BRA N	Ζ 🛛	Branch if No	ot Zero					
Syntax:	{label:} E	BRA	NZ,	Expr				
Operands:	Expr may be Expr is resol Slit16 ∈ [-32	ved by the li	nker to a Sli	ss or express t16, where	ion.			
Operation:								
Status Affecte	d: None							
Encoding:	0011	1010	nnnn	nnnn	nnnn	nnnn		
	offset of the ports branch value is reso expression.	If the Z flag is '0', then the program will branch relative to the next PC. The offset of the branch is the 2's complement number '2*Slit16', which supports branches up to 32K instructions forward or backward. The Slit16 value is resolved by the linker from the supplied label, absolute address or expression.						
	PC will have	incremente	d to fetch the	ss will be (PC- e next instruct n, with a NOP (tion. The inst	ruction		
	The 'n' bits a offset from (F		literal that s	pecifies the n	umber of inst	ructions		
Words:	1							
Cycles:	1 (2 if branch	n taken)						
Example 1	002000 HERE: 002002 NO_NZ: 002004 002006 002008 00200A 00200C BYPASS: 00200E	BRA NZ, GOTO TH 	BYPASS ERE		, branch t wise co			
	Before Instruction PC 00 2000 SR 0002	(Z=1)	Ins	After truction 0 2002 0002 (Z=1	1)			

BRA (Branch if Overflow Accumulator A			
Syntax:	{label:} BRA OA, Expr			
Operands:	Expr may be a label, absolute address or expression. Expr is resolved by the linker to a Slit16, where Slit16 \in [-32768 +32767].			
Operation:	Condition = OA If (Condition) (PC+2) + 2*Slit16 \rightarrow PC NOP \rightarrow Instruction Register			
Status Affect	ed: None			
Encoding:	0000 1100 nnnn nnnn nnnn nnnn			
Description: If the Overflow Accumulator A flag is '1', then the program will branch relative to the next PC. The offset of the branch is the 2's complemer number '2*Slit16', which supports branches up to 32K instructions forward or backward. The Slit16 value is resolved by the linker from t supplied label, absolute address or expression.				
If the branch is taken, the new address will be (PC+2) + 2*Slit1 the PC will have incremented to fetch the next instruction. The then becomes a two-cycle instruction, with a NOP executed in t cycle.				
	The 'n' bits are a signed literal that specifies the number of instructions offset from (PC+2).			
	Note: The assembler will convert the specified label into the offset be used.			
Words:	1			
Cycles:	1 (2 if branch taken)			
Example 1	002000 HERE: BRA OA, BYPASS ; If OA, branch to BYPA 002002 NO_OA: ; Otherwise continue 002004 ; 002006 ; 002008 ; 00200C BYPASS: ; 00200E ;			
	Before After Instruction Instruction PC 00 2000 SR 8800 (OA, OAB=1)			

BRA O	В в	ranch if Ov	erflow Acc	umulator B		
Syntax:	{label:} B	RA (OB,	Expr		
Operands:	Expr may be Expr is resolv Slit16 ∈ [-327	ed by the lir	nker to a Sli		ion.	
Operation:		B *Slit16 \rightarrow P struction Reg				
Status Affected	None					
Encoding:	0000	1101	nnnn	nnnn	nnnn	nnnn
	ative to the no ber '2*Slit16', backward. Th label, absolut If the branch the PC will ha then become	which supp e Slit16 value e address o is taken, the ive increment	orts branch ue is resolv r expressio new addre nted to fetcl	es up to 32K ed by the link n. ess will be (PC n the next inst	instructions f er from the s C+2) + 2*Slit1 truction. The	forward or upplied 6, since instruction
	cycle. The 'n' bits ar offset from (P		iteral that s	pecifies the n	umber of inst	ructions
Words:	1	,				
Cycles:	1 (2 if branch	taken)				
Example 1	002000 HERE: 002002 NO_OB: 002004 002006 002008 00200A 00200C BYPASS: 00200E	BRA OB, GOTO THE 			8, branch f	
	Before Instruction PC 00 2000 SR 8800 (0	DA, OAB=1)	PC	After struction	A, OAB=1)	

BRA O	V	Branch i	f Overflow			
Syntax:	{label:}	BRA	OV,	Expr		
Operands:		olved by th	e linker to a	ress or express Slit16, where	ion.	
Operation:						
Status Affected	None					
Encoding:	0011	0000	nnnn	nnnn	nnnn	nnnn
Description:	PC. The of which supp	fset of the ports branc e is resolve	branch is the hes up to 32 ad by the link	program will bra 2's complemer (instructions fo er from the supp	nt number '2*S rward or back	Slit16', ward. The
	PC will hav	e incremei	nted to fetch t	ress will be (PC he next instruct vith a NOP exect	ion. The instru	uction then
	The 'n' bits offset from	-	ed literal that	specifies the n	umber of inst	ructions
Words:	1					
Cycles:	1 (2 if bran	ch taken)				
	002000 HERE: 002002 NO_OV 002004 002006 002008 002008 00200A 00200C BYPASS 00200E	 GOTC	OV, BYPASS		V, branch t rwise co	
	Before Instruction PC 00 2000 SR 0000		PC SR	After Instruction 00 2002 0002 (Z=	-1)	

BRA S	SA		Branch if S	Saturation	Accumulator A	A	
Syntax:		{label:}	BRA	SA,	Expr		
Operands:	l	Expr is res	be a label, ab colved by the 32768 +32	linker to a	ress or express Slit16, where	ion.	
Operation:							
Status Affecte	ed: I	None					
Encoding:	Γ	0000	1110	nnnn	nnnn	nnnn	nnnn
Description:		relative to number '2' or backwa	the next PC. 'Slit16', which	The offset n supports b o value is re	g is '1', then the of the branch is pranches up to 3 prolved by the li ion.	the 2's comp 32K instructio	olement ns forward
	1	PC will ha	ve incremente	ed to fetch	ress will be (PC the next instruc ion, with a NOP	tion. The inst	ruction
		The 'n' bits	-	d literal that	specifies the n	umber of inst	ructions
Words:		1					
Cycles:		1 (2 if brar	ich taken)				
Example 1	00200 00200 00200 00200 00200	6 8 A C BYPASS	 GOTO T	., BYPASS HERE	-	A, branch rwise c	
	PC SR	Before nstruction 00 2000 2400		PC	After Instruction 00 200C 2400 (SA	A, SAB=1)	

BRA S	B Branch if Saturation Accumulator B							
Syntax:	{label:} BRA SB, Expr							
Operands:	Expr may be a label, absolute address or expression. Expr is resolved by the linker to a Slit16, where Slit16 \in [-32768 +32767].							
Operation:	Condition = SB if (Condition) (PC+2) + 2*Slit16 \rightarrow PC NOP \rightarrow Instruction Register	f (Condition) (PC+2) + 2*Slit16 \rightarrow PC						
Status Affecte	d: None							
Encoding:	0000 1111 nnnn nnnn nnnn	n						
Description:	number '2*Slit16', which supports branches up to 32K instructions forward	relative to the next PC. The offset of the branch is the 2's complement number '2*Slit16', which supports branches up to 32K instructions forward or backward. The Slit16 value is resolved by the linker from the supplied						
	If the branch is taken, the new address will be (PC+2) + 2*Slit16, since the PC will have incremented to fetch the next instruction. The instruct then becomes a two-cycle instruction, with a NOP executed in the second cycle.	ion ond						
	The 'n' bits are a signed literal that specifies the number of instructions offset from (PC+2).	5						
Words:	1							
Cycles:	1 (2 if branch taken)							
Example 1	002000 HERE: BRA SB, BYPASS ; If SB, branch to BYPAS 002002 NO_SB: ; Otherwise continue 002004 ; 002006 ; 002008 ; 00200C BYPASS: ; 00200E ;	SS						
	Before After Instruction Instruction PC 00 2000 SR 0000 SR 0000							

BRA Z		Branch if Ze	ero					
Syntax:	{label:}	BRA	Ζ,	Expr				
Operands:	Expr is reso	e a label, abs blved by the li 2768 +327	nker to a Slit		on.			
Operation:		_						
Status Affected	: None					-		
Encoding:	0011	0010	nnnn	nnnn	nnnn	nnnn		
	supports br value is res expression.	The offset of the branch is the 2's complement number '2*Slit16', which supports branches up to 32K instructions forward or backward. The Slit16 value is resolved by the linker from the supplied label, absolute address or expression.						
	PC will hav	If the branch is taken, the new address will be $(PC+2) + 2*Slit16$, since the PC will have incremented to fetch the next instruction. The instruction then becomes a two-cycle instruction, with a NOP executed in the second cycle.						
	The 'n' bits offset from	are a signed (PC+2).	literal that sp	ecifies the nu	Imber of instr	uctions		
Words:	1							
Cycles:	1 (2 if brand	ch taken)						
Example 1	002000 HERE: 002002 NO_Z: 002004 002006 002008 00200A 00200A 00200C BYPAS 00200E	 GOTO 1	, BYPASS FHERE		Z, branch herwise			
	Before			After				
	Instruction	-	· · · · · · · · · · · · · · · · · · ·	nstruction				
	PC 00 200		PC	00 200C	7 4)			
	SR 000	2 (Z=1)	SR	0002 (2	Z=1)			

BSET		Bit Set f				
Syntax:	{label:}	BSET{.B}	f,	#bit4		
Operands:	f ∈ [0 8′ bit4 ∈ [0	191] for byte (190] (even on . 7] for byte o . 15] for word	ly) for word operation	peration		
Operation:	$1 \rightarrow f \le bit4$	>				
Status Affected:	None	_				
Encoding:	1010	1000	bbbf	ffff	ffff	fffb
Description:	with the Le	east Significa	gister f specifi nt bit (bit 0) a ions, bit 15 fo	nd advances	to the Most	
			bit4 of the bi		be set.	
		denote a wo When this ir address mu	a word opera ord operation, nstruction ope st be word ali nstruction op and 7.	but it is not i erates in Wor gned.	required. d mode, the	file register
Words:	1					
Cycles:	1					
Example 1 BSI	ET.B 0x60	01, #0x3	; Set b	it 3 in Ox	601	
Data 0600 SF		Data 06	After Instruction 600 FA34 SR 0000			
Example 2 BSI	ET 0x44	4, #0xF	; Set b	it 15 in 0	x444	
Data 0444 SF		Data 04	After Instruction 44 D604 SR 0000			

BSET		Bit Set in V	Vs			
Syntax:	{label:}	BSET{.B}	Ws,	#bit4		
			[Ws],			
			[Ws++],			
			[Ws],			
			[++Ws],			
			[Ws],			
Operands:		W15] 7] for byte o 15] for word				
Operation:	$1 \rightarrow Ws \le b$	it4>				
Status Affected:	None					
Encoding:	1010	0000	bbbb	0B00	0ppp	SSSS
Description:	Least Sign for byte op	ificant bit (bit	s specified by 0) and advan 15 for word op d for Ws.	ces to the Mo	ost Significan	t bit (bit 7
	The 'B' bit The 'p' bits	selects byte of select the so	bit4 of the bit or word opera ource Address ddress of the s	tion (0 for wo mode.	rd, 1 for byte	
	2:	rather than a denote a wo When this register add	on .B in the a word operati rd operation, instruction op ress must be nstruction ope nd 7.	ion. You may but it is not re perates in W word aligned.	y use a .w e equired. /ord_mode, 1	xtension t
Words:	1					
Cycles:	1					
Example 1	BSET.B W3,	#0x7	; Set b	it 7 in W3		
	Before Instruction W3 0026 SR 0000		After Instruction W3 00A6 SR 0000			
Example 2	BSET [W4++]	, #0x0		t 0 in [W4 ncrement W		
Data 6	Before Instruction W4 6700 3700 1734 SR 0000	Data 67	After Instruction W4 6702 700 1735 SR 0000			

Instruction Descriptions

BSW		Bit Write	in Ws			
Syntax:	{label:}	BSW.C BSW.Z	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wb		
Operands:	Ws ∈ [W0 Wb ∈ [W0					
Operation:	For ".C" o C → W <u>For</u> ".Z" o		ault):			
Status Affected						
Encoding:	1010	1101	Zwww	w000	0ppp	SSSS
Description:	the Status 0) and ad Only the f nation bit either reg The 'Z' bit	register. Bit vances to the our Least Sig number. Reg ister direct, c	numbering be Most Signific gnificant bits o gister direct ad r indirect add C or Z flag as		Least Signific of the workin to determine the used for e used for W	cant bit (bit ng register. e the desti- Wb, and
	The 'p' bit	s select the s s select the a	source Addres address of the	source registe	er.	
	Note:			rates in Word r tion is assume		xtension is
Words:	1	-	-			
Cycles:	1					
Example 1	BSW.C W2, W	13	-	bit W3 in W the C bit	V2 to the	value
	Before Instruction W2 F234 W3 111F SR 0002 (Z=1, C=0)	After Instructi W2 7234 W3 1118 SR 0002	7		
Example 2	BSW.Z W2, W	3		bit W3 in W he Z bit	2 to the c	complement
	Before Instruction W2 E235 W3 0550 SR 0002 (a)	Z=1, C=0)	After Instructio W2 E234 W3 0550 SR 0002			

Example 3 BSW.	C [++W	0], W6	; Set bit W6 in [W0++] to the value ; of the C bit
	Before		After
li	nstruction		Instruction
W0	1000		W0 1002
W6	34A3		W6 34A3
Data 1002	2380	Data 1	002 2388
SR	0001	(Z=0, C=1)	SR 0001 (Z=0, C=1)
Example 4 BSW	[W1-	-], W5	; Set bit W5 in [W1] to the ; complement of the Z bit ; Post-decrement W1
	Before		After
I	nstruction		Instruction
W1	1000	W	1 OFFE
W5	888B	W	5 888B
Data 1000	C4DD	Data 100	0 CCDD
SR	0001	(C=1) SI	R 0001 (C=1)

© 2003 Microchip Technology Inc.

BTG		Bit Toggle	f			
Syntax:	{label:}	BTG{.B}	f,	#bit4		
Operands:	f ∈ [0 8 bit4 ∈ [0	191] for byte (190] (even on 7] for byte o 15] for word	ly) for word o peration	peration		
Operation:	(f) <bit4></bit4>	\rightarrow (f) <bit4></bit4>				
Status Affected:	None					
Encoding:	1010	1010	bbbf	ffff	ffff	fffb
Description:	bit numbe	n file register f ring begins wi Significant bit (e.	th the Least S	Significant bit	(bit 0) and ad	dvances to
		s select value s select the ac			ggle.	
	2:	denote a wo When this ir	a word operation, rd operation, istruction ope st be word ali istruction op	tion. You may but it is not r rates in Wor gned.	y use a .w e required. d mode, the t	xtension to file register
Words:	1					
Cycles:	1					
Example 1 BTG	.B 0x1	001, #0x4	; Toggle	bit 4 in	0x1001	
Data 1000 SR	Before nstruction F234 0000	Data 100 S				
Example 2 BTG	0x16	560, #0x8	; Toggle	bit 8 in 1	RAM660	
ا Data 1660 SR	Before nstruction 5606 0000	Data 166 Si				

BTG		Bit Toggle	e in Ws			
Syntax:	{label:}	BTG{.B}	Ws,	#bit4		
			[Ws],			
			[Ws++],			
			[Ws],			
			[++Ws],			
			[Ws],			
Operands:		W15] . 7] for byte . 15] for wor				
Operation:	(Ws) <bit4></bit4>	\rightarrow Ws <bit< td=""><td>4></td><td></td><td></td><td></td></bit<>	4>			
Status Affected:	None					
Encoding:	1010	0010	bbbb	0800	0ppp	SSSS
	the Most S operations	ignificant bi	t (bit 7 for byte	operations, l ct addressing	: (bit 0) and ad bit 15 for word may be used	
	The 'B' bit The 's' bits	selects byte s select the a	or word oper	ation (0 for wo source/destir	ord, 1 for byte) nation register	
	2:	rather than denote a w When this register add	a word opera ord operation, instruction o dress must be instruction op	tion. You ma but it is not r perates in V word aligned	Vord mode, t	ttension t
Words:	1					
Cycles:	1					
Example 1	BTG W2, #0x	0	; Toggl	e bit 0 in	W2	
	Before		After			
	Instruction		Instruction			
	W2 F234 SR 0000		W2 F235 SR 0000			
Example 2	BTG [W0++],	#0x0	. 55	e bit 0 in increment N		
Data 2	Before Instruction W0 2300 300 5606 SR 0000	Data 2	After Instruction W0 2302 300 5607 SR 0000			

BTSC	Bit Test f, Skip if Clear
Syntax:	{label:} BTSC{.B} f, #bit4
Operands:	$f \in [0 \dots 8191]$ for byte operation $f \in [0 \dots 8190]$ (even only) for word operation bit4 $\in [0 \dots 7]$ for byte operation bit4 $\in [0 \dots 15]$ for word operation
Operation:	Test (f) <bit4>, skip if clear</bit4>
Status Affected:	None
Encoding:	1010 1111 bbbf ffff ffff fffb
Description:	Bit 'bit4' in the file register is tested. If the tested bit is '0', the next instruction (fetched during the current instruction execution) is discarded and on the next cycle, a NOP is executed instead. If the tested bit is '1', the next instruction is executed as normal. In either case, the contents of the file register are not changed. For the bit4 operand, bit numbering begins with the Least Significant bit (bit 0) and advances to the Most Significant bit (bit 7 for byte operations, bit 15 for word operations).
	The 'b' bits select value bit4, the bit position to test. The 'f' bits select the address of the file register.
	 Note 1: The extension .B in the instruction denotes a byte operation rather than a word operation. You may use a .W extension to denote a word operation, but it is not required. 2: When this instruction operates in Word mode, the file register address must be word aligned. 3: When this instruction operates in Byte mode, 'bit4' must be between 0 and 7.
Words:	1
Cycles:	1 (2 or 3)
00 00 00 00	2000 HERE: BTSC.B 0x1201, #2 ; If bit 2 of 0x1201 is 0, 2002 GOTO BYPASS ; skip the GOTO 2004 2006 2008 BYPASS: 200A
P(Data 120 SF	0 264F Data 1200 264F

()	002000 HERE: 002002 002004 002006 002008 BYPASS: 00200A	BTSC 0x804, GOTO BYPASS 	#14 ; If bit 14 of 0x804 is 0, ; skip the GOTO
Before Instruction			After Instruction
	PC 00 2000	PC	00 2004
Data 08	304 2647	Data 0804	2647
	SR 0000	SR	0000

BTSC		Bit Test W	/s, Skip if Cle	ar		
Syntax:	{label:}	BTSC	Ws,	#bit4		
			[Ws],			
			[Ws++],			
			[Ws],			
			[++Ws],			
			[Ws],			
Operands:	Ws ∈ [W0 bit4 ∈ [0	-				
Operation:	Test (Ws)	<bit4>, skip i</bit4>	f clear			
Status Affecte	d: None					
Encoding:	1010	0111	bbbb	0000	0ppp	SSSS
	is execute changed. Significan the word. The 'b' bit The 'p' bit	ed as normal For the bit4 t bit (bit 0) ar Either registe s select valu s select the s s select the a	In either case operand, bit n nd advances t er direct or inc e bit4, the bit source Addres address of the	source regist	s of Ws are n ins with the L gnificant bit (b ng may be us t. er.	ot east bit 15) of
Words:	1		clion operates	in Word mod	e only.	
Cycles:	-	f the next ins	truction is skip	oped)		
Example 1	002000 HERE: 002002 002004 002006 002008 BYPAS 00200A	BTSC GOTO S:	W0, #0x0 BYPASS		bit 0 of N ip the GOTO	
	Before Instruction PC 00 200 W0 264 SR 000	0 F	PC W0 SR	After Instruction 00 2002 264F 0000		

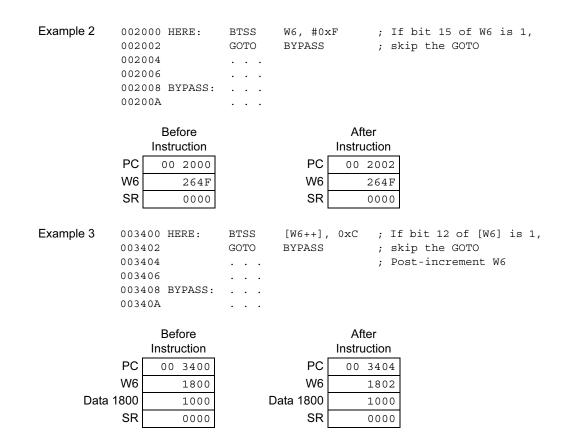
002 002 002	004 006 008 BYPASS:		W6, #0xF BYPASS	; If bit 15 of W6 is 0, ; skip the GOTO
PC W6 SR	Before Instruction 00 2000 264F 0000			After struction 00 2004 264F 0000
0034 0034 0034	404 406 408 BYPASS:	BTSC GOTO 	[W6++], #0 BYPASS	<pre>xC ; If bit 12 of [W6] is 0, ; skip the GOTO ; Post-increment W6</pre>
	Before			After
	Instruction		Ins	struction
PC	00 3400		PC	00 3402
W6	1800		W6	1802
Data 1800	1000	Da	ata 1800	1000
SR	0000		SR	0000

5

BTSS		Bit Test f, S	kip if Set						
Syntax:	{label:}	BTSS{.B}	f,	#bit4					
Operands:	f ∈ [0 819 bit4 ∈ [0	91] for byte c 90] (even onl 7] for byte of 15] for word	y) for word operation	peration					
Operation:	Test (f) <bit4< td=""><td>>, skip if set</td><td></td><td></td><td></td><td></td></bit4<>	>, skip if set							
Status Affected:	None		1						
Encoding:	1010	1110	bbbf	ffff	ffff	fffb			
Description:	instruction (and on the i next instruc file register with the Lea	Bit 'bit4' in the file register f is tested. If the tested bit is '1', the next instruction (fetched during the current instruction execution) is discarded and on the next cycle, a NOP is executed instead. If the tested bit is '0', the next instruction is executed as normal. In either case, the contents of the file register are not changed. For the bit4 operand, bit numbering begins with the Least Significant bit (bit 0) and advances to the Most Significant bit (bit 7 for byte operation, bit 15 for word operation).							
			bit4, the bit p dress of the f		t.				
	2:	rather than a denote a wo When this in address mus	on . B in the a word operation, struction ope to be word align struction ope nd 7.	tion. You may but it is not re rates in Word gned.	y use a .w e equired. d mode, the	xtension to file register			
Words:	1								
Cycles:	1 (2 or 3 if t	he next instr	uction is skip	ped)					
. 00	7100 HERE: 7102 7104	BTSS.B CLR · · ·	0x1401, WREG	#0x1; If k ; don4	oit 1 of 0x t clear WI	-			
Pi Data 140 Si	0 0280	D	In PC ata 1400 SR	After Istruction 00 7104 0280 0000					
00	7100 HERE: 7102 7104 7106 BYPASS	BTSS GOTO 	0x890, BYPASS	#0x9 ; If] ; skij	bit 9 of 0: p the GOTO	x890 is 1,			
P(Data 089 SI	0 00FE		Ir PC Data 0890 SR	After Instruction 00 7102 00FE 0000					

BTSS			Bit Test Ws	, Skip if Set			
Syntax:	{Ia	ibel:}	BTSS	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	#bit4		
Operands:		s ∈ [W0 . t4 ∈ [0	-				
Operation:	Те	st (Ws)<	bit4>, skip if	set.			
Status Affecte	ed: No	one					
Encoding:		1010	0110	bbbb	0000	0ppp	SSSS
	ch Si th Tr Tr	anged. F gnificant e word. E ne 'b' bits ne 's' bits	or the bit4 or bit (bit 0) and ither register select the va select the ad	berand, bit nu d advances to direct or indi alue bit4, the	e, the contents imbering beg the Most Sig rect addressi bit position to source regist s mode.	ins with the L gnificant bit (b ng may be us test.	east it 15) of
		Note:	This instruct	ion operates	in Word mode	e only.	
Words: Cycles:	1 1	(2 or 3 if 1	the next instr	uction is skip	ped)		
Example 1	002000 002002 002004 002006 002008 00200A	HERE: BYPASS	BTSS GOTO 	W0, #0x0 BYPASS	-	bit 0 of W ip the GOTC	-
	In	Before struction 00 2000 264F 0000	-	PC W0 SR	After nstruction 00 2004 264F 0000		

dsPIC30F Programmer's Reference Manual



BTST		Bit Test f				
Syntax:	{label:}	BTST{.B}	f,	#bit4		
Operands:	f ∈ [0 8′ bit4 ∈ [0	191] for byte o 190] (even on . 7] for byte o . 15] for word	ly) for word c peration	operation		
Operation:	(f) <bit4> -</bit4>	×Z				
Status Affected:	Z					
Encoding:	1010	1011	bbbf	ffff	ffff	fffb
Description:	stored to th are not ch Least Sigr for byte op	ne Z flag in th anged. For th ificant bit (bit peration, bit 1	e Status Reg e bit4 operar 0) and adva 5 for word op	ister. The con ad, bit numbe nces to the M eration).	nent of the te ntents of the fi ring begins w lost Significar	ile registe rith the
		s select value select the ad			e tested.	
		denote a wo When this ir address mus	rd operation, istruction ope st be word all nstruction op	but it is not i erates in Wor gned.	y use a .wex required. 'd mode, the f te mode, 'bit₄	ile registe
Words:	1					
Cycles:	1					
Example 1 BT	ST.B 0x12	201, #0x3		= complem in 0x1201		
Data 1200 SF		Data 12	After Instruction 00 F7FF SR 0002			
Example 2 BT	ST 0x1	302, #0x7		Z = complen / in 0x1302		
Data 130 Sl		Data 13 (Z=1)	After Instructior 302 F7FF SR 0000	I		

BTST		Bit Test in	Ws			
Syntax:	{label:}	BTST.C BTST.Z	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	#bit4		
Operands:	Ws ∈ [W0 bit4 ∈ [0					
Operation:	<u>For ".C" o</u> (Ws) <b <u>For ".Z" o</u></b 	-	<u>ult):</u>			
Status Affected:	Z or C					
Encoding:	1010	0011	bbbb	Z000	0ppp	SSSS
	Status reg the tested case, the For the bit (bit 0) and register di The 'b' bit The 'Z' bit The 'p' bit	ister. If the ". bit is stored contents of W 4 operand, b advances to rect or indirec s select value selects the C s select the s	ent of the test c" option of the to the Carry fla /s are not chan it numbering b the Most Sign ct addressing r b bit4, the bit p c or Z flag as d ource Address ddress of the s	e instruction ag in the State nged. egins with th ificant bit (bit nay be used osition to tes lestination.	is specified, thus register. In e Least Signin t 15) of the wo for Ws. t.	ne value of either ficant bit
	Note:		tion only operation only operation only operation of the content o			extension is
Words:	1					
Cycles:	1					
Example 1 BT	ST.C [W04	-+], #0x3		= bit 3 in ncrement W		
W(Data 120(SF) FFF7	Data 12	After Instruction W0 1202 200 FFF7 SR 0000			
Example 2 BI	ST.Z WO,	#0x7	; Set Z	= compleme	nt of bit ?	7 in W0
W			After Instruction V0 F234 SR 0002 (2	Z=1)		

BTST		Bit Test in	Ws					
Syntax:	{label:}	BTST.C	Ws,	Wb				
		BTST.Z	[Ws],					
			[Ws++],					
			[Ws],					
			[++Ws],					
			[Ws],					
Operands:	Ws ∈ [W0 Wb ∈ [W0	-						
Operation:		peration: Wb)> \rightarrow C peration (def	ault):					
	(Ws)<($Wb)> \rightarrow Z$						
Status Affected:	Z or C			1	1	1		
Encoding:	1010	0101	Zwww r Ws is tested	w000	0ppp	SSSS		
	the tested the conter Only the f	register. If the ". Z" option of the instruction is specified, the complement of the tested bit is stored to the Zero flag in the Status register. In either case, the contents of Ws are not changed.Only the four Least Significant bits of Wb are used to determine the bit number. Bit numbering begins with the Least Significant bit (bit 0) and						
			ect addressing			3161.		
	The 'w' bi The 'p' bit	ts select the s s select the s	C or Z flag as address of the source Addres address of the	e bit select reg s mode.				
	Note:		ction only oper ne ". z" operat			extension is		
Words:	1							
Cycles:	1							
Example 1	BTST.C W2	, W3	; Set	C = bit W3	of W2			
	Before Instruction W2 F234 W3 2368 SR 0001	(C=1)	After Instructio W2 F234 W3 2368 SR 0000	n 				

Instruction Descriptions

Example 2 BT	ST.Z [W	0++],	Wl	; bit W			of		
	Before				After				
	Instruction	n	Instruction						
W	1200		W0	1202					
W	CCC0		W1	CCC0					
Data 1200	6243		Data 1200	6243					
SF	R 0002	(Z=1)	SR	0000					
		-	L						

BTSTS	Bit Test/Set f
Syntax:	{label:} BTSTS{.B} f, #bit4
Operands:	$\begin{array}{l} f \in [0 \ \ 8191] \ \text{for byte operation} \\ f \in [0 \ \ 8190] \ (\text{even only}) \ \text{for word operation} \\ \text{bit4} \in [0 \ \ 7] \ \text{for byte operation} \\ \text{bit4} \in [0 \ \ 15] \ \text{for word operation} \end{array}$
Operation:	$\overline{(f) < bit4>} \rightarrow Z$ 1 \rightarrow (f) < bit4>
Status Affected:	Z
Encoding:	1010 1100 bbbf ffff ffff fffb
Description:	Bit 'bit4' in file register f is tested and the complement of the tested bit is stored to the Zero flag in the Status register. The tested bit is then set to "1" in the file register. For the bit4 operand, bit numbering begins with the Least Significant bit (bit 0) and advances to the Most Significant bit (bit 7 for byte operations, bit 15 for word operations).
	The 'b' bits select value bit4, the bit position to test/set. The 'f' bits select the address of the file register.
	 Note 1: The extension . B in the instruction denotes a byte operation rather than a word operation. You may use a .w extension to denote a word operation, but it is not required. 2: When this instruction operates in Word mode, the file register address must be word aligned. 3: When this instruction operates in Byte mode, 'bit4' must be between 0 and 7.
Words:	1
Cycles:	1
Example 1 BTS	TS.B 0x1201, #0x3 ; Set Z = complement of bit 3 in 0x1201, ; then set bit 3 of 0x1201 = 1
I Data 1200 SR Example 2 BTS	
RAM300 SR	BeforeAfterInstructionInstruction8050RAM30080508050

BTSTS		Bit Test/Se	et in Ws			
Syntax:	{label:}	BTSTS.C BTSTS.Z	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	#bit4		
Operands:	Ws ∈ [W0 bit4 ∈ [0	-				
Operation:	$1 \rightarrow W$ For ".Z" of	it4> → C s <bit4> <u>peration (defa</u> it4> → Z</bit4>	uult):			
Status Affected:	Z or C					
Encoding:	1010	0100	bbbb	Z000	0ppp	SSSS
Description:	specified, Status reg the tested cases, the The 'b' bit The 'Z' bit The 'p' bit	the complem ister. If the ". bit is stored to tested bit in s select the v selects the C s select the s	is tested. If th ent of the test C [°] option of th to the Carry fl Ws is set to " alue bit4, the or Z flag as o ource Addres ddress of the	ed bit is store e instruction ag in the Stat 1". bit position to destination. s mode.	ed to the Zero is specified, th us register. Ir o test/set.	flag in the he value of
	Note:		tion only oper e ".z" operati			extension is
Words:	1	•				
Cycles:	1					
Example 1 BTS	IS.C [WO	++], #0x3	; Set b	= bit 3 i it 3 in [W increment	0] = 1	
ا W0 Data 1200 SR	Before nstruction 1200 FFF7 0001 (C	Data 1	After Instruction W0 1202 200 FFFF SR 0000	1		
Example 2 BT:	STS.Z WO	, #0x7		-	ent of bit oit 7 in WC	
W0 SF			After Instruction W0 F2BC SR 0002	(Z=1)		

CALL		Call Subr	outine			
Syntax:	{label:}	CALL	Expr			
Operands:			r expression (b e linker to a lit2			8606].
Operation:	$(PC)+4 \rightarrow I$ $(PC<15:0>$ $(W15)+2 \rightarrow I$ $(PC<23:16)$ $(W15)+2 \rightarrow I$ $Iit23 \rightarrow PC$	PC) → (TOS) • W15 >) → (TOS		,	<u>.</u>	
01 1 4 5 1	$NOP \rightarrow Ins$	truction Re	gister			
Status Affected:	None					
Encoding: 1st word	0000	0010	nnnn	nnnn	nnnn	nnn0
2nd word	0000	0000	0000	0000	0nnn	nnnn
Description:	Direct su	broutine ca	all over the enti	re 4 Mbyte ins	struction pro	gram
	(PC+4) is	s pushed o	ore the call is n nto the stack. A 3' is loaded into	fter the return		
	The 'n' bits	form the ta	arget address.			
	Note:	The linker be used.	will resolve the	specified exp	ression into	the lit23 to
Words:	2					
Cycles:	2					
	6000 6004	CALL MOV 	_FIR WO, W1	; Cal:	l _FIR sub	proutine
	6844 _FIR: 6846	MOV =	#0x400, ₩2	; _ ^{FI1}	R subrouti	ine star
	Before			After		
	Instruction	-		nstruction		
P			PC	02 6844		
W1			W15	A26C		
Data A26		' 	Data A268	6004		
Data A26		-	Data A26A	0002		
SI	R 0000		SR	0000		
	72000 72004	CALL MOV	_G66 ₩0, ₩1	; call	routine _(G66
07	77A28 _G66: 77A2A 77A2C	 INC 	W6, [W7++]	; routi	ne start	
	Before			After		
	Instruction	-		nstruction		
P			PC	07 7A28		
	5 9004		W15	9008		
W1						
Data 900	4 FFFF		Data 9004	2004		
	04 FFFF 06 FFFF	-	Data 9004 Data 9006 SR	2004 0007		

CALL		Call Indirect Su	ıbroutir	ie		
Syntax:	{label:}	CALL Wr	ו			
Operands:	Wn ∈ [W0 .	W15]				
Operation:) → TOS • W15 >) → TOS • W15				
Status Affected:	None	Ū				
Encoding:	0000	0001	0000	0000	0000	SSSS
Description:	Before the the stack. A PC<15:1> a is ignored.	oroutine call over call is made, the After the return ac and PC<22:16> i select the addres	24-bit re ddress is s cleare	eturn address s stacked, Wn d. Since PC<	(PC+2) is pus <15:1> is load D> is always '	shed onto ded into
Words:	1			Ū		
Cycles:	2					
Example 1 0010		CALL W0		Call BOOT : using WO	subroutine	indirectly
0016		MOV #0x400, MOV #0x300, MOV #0x300, MOV		_BOOT star	ts here	
	Before			After		
	Instruction	1		Instruction		
PC	00 1002		PC	00 1600		
WO	1600		W0	1600		
W15	6F00		W15	6F04		
Data 6F00	FFFF	Data		1004		
Data 6F02 SR	FFFF 0000	Data	SR	0000		
	200 202	CALL W7		Call TEST s using W7	subroutine :	indirectly
	500 _TEST: 502	INC W1, W2 DEC W1, W3		_TEST start	s here	
	Before			After		
	Instruction	_	_	Instruction		
PC	00 4200		PC	00 5500		
W7	5500		W7	5500		
W15	6F00		W15	6F04		
Data 6F00	FFFF	Data 6		4202		
Data 6F02	FFFF	Data 6		0000		
SR	0000		SR	0000		

CLR		Clear f or W	REG			
Syntax:	{label:}	CLR{.B}	f			
			WREG			
Operands:	f ∈ [0 81	191]				
Operation:	$0 \rightarrow \text{destin}$	ation designation	ted by D			
Status Affected:	None					
Encoding:	1110	1111	OBDf	ffff	ffff	ffff
Description:		contents of a fi s specified, the s cleared.				
	The 'D' bit	selects byte o selects the de select the add	stination (0 f	or WREG, 1		
		The extension rather than a denote a worn The WREG is	word operat d operation,	ion. You may but it is not re	y use a .w ex equired.	
Words:	1					
Cycles:	1					
Example 1 CL	R.B RAM2	00 ;	Clear RAM2	00 (Byte r	mode)	
RAM200 SF		RAM20 SF				
Example 2 C	LR WRE	G;	Clear WRE	G (Word mc	ode)	
WRE(SI		WRE	After Instruction G 0000 R 0000			

CLR		Clear Wd				
Syntax:	{label:}	CLR{.B}	Wd			
			[Wd]			
			[Wd++]			
			[Wd]			
			[++Wd]			
			[Wd]			
Operands:	Wd ∈ [W0) W15]				
Operation:	$0 \rightarrow Wd$					
Status Affected:	None					
Encoding:	1110	1011	0Bqq	qddd	d000	0000
Description:		contents of re		her register c	lirect or indire	ect
	addressin	g may be use	d for Wd.			
	The 'd' bit Note: 7	s select the de s select the ac he extension ather than a w	dress of the o . B in the instr ord operation	destination re ruction denot You may us	es a byte ope se a .w exter	
Words:		lenote a word	operation, bu	t it is not req	uired.	
	1					
Cycles:	1					
Example 1	CLR.B W2	;	Clear W2 (E	3yte mode)		
	Before		After			
	Instructio		Instruction			
	W2 3333		V2 3300			
	SR 0000		SR 0000			
Example 2	CLR [W		Clear [W0] Post-increm	nent WO		
	Before Instructio W0 2300	V	After Instruction			
Data	2300 5607 SR 0000	Data 23	00 0000 SR 0000			

		Clear A	ccumulator, P	е-гесспоре	ranus	
Syntax:	[label:} CLR	Acc	{,[Wx],Wxd}	{,[Wy],		{,AWB}
				Vxd} {,[Wy]·		
			{,[Wx]-=kx,V	√xd} {,[Wy]·	=ky,Wyd}	
			{,[W9+W12]	,Wxd} {,[W11	+W12],Wyd}	
Operands:	Wy ∈ [V	N8, W9]; kx ∈	:: [-6, -4, -2, 2, 4 / ∈ [-6, -4, -2, 2 +=2]			
Operation:	([Wx])— ([Wy])—	c(A or B) → Wxd; (Wx)- → Wyd; (Wy)- or A)) rounde	⊦/-ky→Wy			
Status Affected	OA, OB	s, SA, SB				
Encoding:	110	0011	A0xx	yyii	iijj	jjaa
Description:	operand the non	ds in preparat -specified acc	e specified acc ion for a MAC to cumulator resu and saturate fla	ype instruction Its. This instru	n and optiona liction clears t	illy store he
	Sectior optiona content	4.14.1 "MA I register dire	et and register C Pre-Fetches et or indirect st r" accumulator	". Operand A ore of the con	WB specifies	the Inded
	The 'x' The 'y' The 'i' b The 'j' b	bits select the bits select the bits select the bits select the	e other accumu pre-fetch Wxc pre-fetch Wyc Wx pre-fetch c Wy pre-fetch c e accumulator v	d destination. d destination. operation. operation.		
Words:	1					
Cycles:	1					
Example 1	CLR A, [W8	3]+=2, W4,	; Load	r ACCA W4 with [W e ACCB to W	-	inc W8
	Befo			After		
	Instruc		14/4	Instruction	7	
	W4	F001	W4	1221		
	W8	2000	VVA I			
	W8 W13	2000 C623	W8	2002 5420		
		C623	W13)	
A A	W13 CCA 00 0067 CCB 00 5420	C623 2345	W13 ACCA 00 ACCB 00	5420)	
A	W13 CCA 00 0067 CCB 00 5420	C623 2345	W13 ACCA 00	5420 0 0000 0000)) -	

Example 2	CLR	B, [W8]+=2, W	6, [W10]+=2, W7,	[W13]+=2 ;	Clear ACCB
·				;	Load W6 with [W8]
				;	Load W7 with [W10]
				;	Save ACCA to [W13]
				;	Post-inc W8,W10,W13
		Before		After	
		Instruction		Instruction	

	I	nstruct	tion		I	nstruct	ion
W6			F001	W6			1221
W7			C783	W7			FF80
W8			2000	W8			2002
W10			3000	W10			3002
W13			4000	W13			4002
ACCA	00	0067	2345	ACCA	00	0067	2345
ACCB	00	5420	ABDD	ACCB	00	0000	0000
Data 2000			1221	Data 2000			1221
Data 3000			FF80	Data 3000			FF80
Data 4000			FFC3	Data 4000			0067
SR			0000	SR			0000

CLRWD	Г	Clear Watch	ndog Timer			
Syntax:	{label:}	CLRWDT				
Operands:	None					
Operation:	$0 \rightarrow WDT$	count register prescaler A co prescaler B co	ount			
Status Affected:	None					
Encoding:	1111	1110	0110	0000	0000	0000
Description:	prescaler o	contents of the count registers et by configura	s. The Watch	dog Prescal	er A and Pres	scaler B
Words:	1					
Cycles:	1					
Example 1	CLRWDT	; Clear Wat	chdog Time	er		
	Before Instruction SR 0000	SI	After Instruction			

© 2003 Microchip Technology Inc.

COM		Complemer	nt f			
Syntax:	{label:}	COM{.B}	f	{,WREG}		
Operands:	f ∈ [0 81	-				
Operation:		nation designa	ated by D			
Status Affected:	N, Z	1			1	1
Encoding:	1110	1110 he 1's comple	1BDf	ffff	ffff	ffff
Description:	the result determines stored in V register. The 'B' bit The 'D' bit	in the dest the destinat VREG. If WRE selects byte o selects the de select the add	ination regis ion register. EG is not spe r word opera estination (0 f	ter. The op If WREG is ecified, the re tion (0 for wo or WREG, 1	tional WREC specified, th esult is stored ord, 1 for byte	G operand le result is d in the file e).
	2:	The extension rather than a denote a wore The WREG is	word operat	tion. You may but it is not re	y use a .w e equired.	
Words:	1					
Cycles:	1					
Example 1	COM.b RAM	200 ; CC	OM RAM200	(Byte mode))	
	R 0000	RAM20 S	SR 0002 (Z)		
Example 2 c	OM RAM4	100, WREG		AM400 and mode)	store to W	REG
WRE RAM4(S	-	WRE RAM40	00 0823	N=1)		

		Compleme	ent Ws			
Syntax:	{label:}	COM{.B}	Ws,	Wd		
			[Ws],	[Wd]		
			[Ws++],	[Wd++]		
			[Ws],	[Wd]		
			[++Ws],	[++Wd]		
			[Ws],	[Wd]		
Operands:	Ws ∈ [W0 Wd ∈ [W0					
Operation:	$\overline{(Ws)} \rightarrow W$	/d				
Status Affected:	N, Z					
Encoding:	1110	1010	1Bqq	qddd	dppp	SSSS
Description:	and place	the result in	the destinatio	contents of th on register Wd. both Ws and	Either regist	
	The 'q' bit The 'd' bit The 'p' bit	s select the c s select the a s select the s	lestination Ad address of the source Addres	ration (0 for wo ddress mode. e destination re ss mode. e source regist	egister.	9).
	Note:	rather than	a word oper	e instruction de ation. You ma , but it is not re	yusea.we	
Words:	1					
	4					
Cycles:	1					
Cycles: Example 1 COM				and store rement W0,		yte mode
Example 1 COM	.B [W0++] Before					yte mode
Example 1 COM.	.B [W0++] Before Instruction		; Post-inc: After Instructio	rement W0, '		yte mode
Example 1 COM	.B [W0++] Before Instruction 2301		; Post-inc: After Instructio W0 2302	n		yte mode
Example 1 COM W0 W1	.B [W0++] Before Instruction 2301 2400		; Post-inc: After Instructio W0 2302 W1 2401	n		yte mode
Example 1 COM	.B [W0++] Before Instruction 2301		; Post-inc: After Instructio W0 2302 W1 2401 300 5607	n n		yte mode
Example 1 COM W0 W1 Data 2300	.B [W0++] Before Instruction 2301 2400 5607	Data 23 Data 24	; Post-inc: After Instructio W0 2302 W1 2401 300 5607	n n		yte mode
Example 1 COM W0 W1 Data 2300 Data 2400	.B [W0++] Before Instruction 2301 2400 5607 ABCD	Data 2: Data 24 ++1] ;	; Post-inc: After Instructio W0 2302 W1 2401 300 5607 400 ABA9 SR 0008	n (N=1)	W1	-
Example 1 COM W0 W1 Data 2300 Data 2400 SR Example 2 COM	B [W0++] Before Instruction 2301 2400 5607 ABCD 0000 W0, [W1 Before	Data 2: Data 24 ++1] ;	; Post-inc: After Instructio WO 2302 W1 2401 300 5607 400 ABA9 SR 0008 COM W0 and Post-incre After	n (N=1) store to [ment W1	W1	-
Example 1 COM W0 W1 Data 2300 Data 2400 SR Example 2 COM	.B [W0++] Before Instruction 2301 2400 5607 ABCD 0000 W0, [W1 Before Instruction	Data 23 Data 24 ++] ; ;	; Post-inc: After Instructio W0 2302 W1 2401 300 5607 400 ABA9 SR 0008 COM W0 and Post-incre After Instruction	n (N=1) store to [ment W1	W1	-
Example 1 COM W0 W1 Data 2300 Data 2400 SR Example 2 COM	B [W0++] Before Instruction 2301 2400 5607 ABCD 0000 W0, [W1 Before	Data 2: Data 24 ++] ; ; V	; Post-inc: After Instructio WO 2302 W1 2401 300 5607 400 ABA9 SR 0008 COM W0 and Post-incre After	n (N=1) store to [ment W1	W1	-
Example 1 COM W0 W1 Data 2300 Data 2400 SR Example 2 COM	.B [W0++] Before Instruction 2301 2400 5607 ABCD 0000 W0, [W1 Before Instruction D004	Data 2: Data 24 ++] ; ; V	; Post-inc: After Instructio W0 2302 W1 2401 300 5607 400 ABA9 SR 0008 COM W0 and Post-incre After Instruction V0 D004 V1 1002	n (N=1) store to [ment W1	W1	-

СР		Compare f	with WREG,	Set Status F	lags	
Syntax:	{label:}	CP{.B}	f			
Operands:	f ∈ [0819	91]				
Operation:	(f) – (WRE	-				
Status Affected:	DC, N, OV	, Z, C				
Encoding:	1110	0011	OBOf	ffff	ffff	ffff
Description:	Compute (f) – (WREG) and update the Status register. This instruction equivalent to the SUBWF instruction, but the result of the subtraction is stored.					
		selects byte o select the ado			ord, 1 for byte	e).
		The extension rather than a denote a wor The WREG is	word operation,	ion. You may but it is not re	y use a .w ex equired.	
Words:	1					
Cycles:	1					
Example 1 CP	.B RAM	100 ; Co	mpare RAM4	00 with WR	EG (Byte m	ode)
WREG RAM40 SI	0 0823	WRE RAM4		(Z=1)		
Example 2 CP	0x12	200 ; Co	mpare (0x1	200) with	WREG (Word	mode)
WRE0 Data 120 SI	0 2277	WRE Data 12		(N=1)		

Suptor	(lobal·)		Wb,	#lit5		
Syntax:	{label:}	CP{.B}	VVD,	#115		
Operands:	Wb ∈ [W0 lit5 ∈ [0					
Operation:	(Wb) – lit	5				
Status Affected:	DC, N, O	V, Z, C				
Encoding:	1110	0001	0www	wB00	011k	kkkk
Description:	equivalen	(Wb) – lit5, ar It to the SUB ir egister direct a	struction, but	the result of	the subtraction	
	The 'B' bi	ts select the a t selects byte ts provide the	or word opera	ation (0 for wo	ord, 1 for byte	
	Note:	rather than	on .B in the a word opera ord operation,	tion. You may	yusea.we	
Words:	1					
Cycles:	1					
Example 1	CP.B W4, #	0x12	; Compare	W4 with 0x	12 (Byte m	ode)
	Before Instruction W4 7711 SR 0000	V	After Instruction V4 7711 SR 0008 (N=1)		
Example 2	CP W4, ‡	#0x12	; Compare	W4 with 0x	12 (Word m	ode)
	Before Instruction W4 7713 SR 0000		After Instruction W4 7713 SR 0000	1		

© 2003 Microchip Technology Inc.

CP		Compare	Wb with Ws,	Set Status Fla	ags	
Syntax:	{label:}	CP{.B}	Wb,	Ws		
				[Ws]		
				[Ws++]		
				[Ws]		
				[++Ws]		
				[Ws]		
Operands:	$Wb \in [W0]$ $Ws \in [W0]$					
Operation:	(Wb) – (W	/s)				
Status Affected:	DC, N, O\	/, Z, C				
Encoding:	1110	0001	0www	wB00	0ppp	SSSS
Description:	equivalen stored. Re	t to the SUB i	nstruction, bu	he Status regis t the result of t lust be used fo Ws.	he subtractio	n is not
	The 'B' bit The 'p' bit	selects byte s select the s	or word oper source Addres	Wb source re ation (0 for wo ss mode. Ws source ree	rd, 1 for byte)).
	Note:	rather than	a word operation	instruction de ation. You may but it is not re	/usea.we	-
Words:	1				1	
Cycles:	1					
Example 1 C	P.B WO,	[W1++]	; Compare ; Post-inci	[W1] with WC cement W1	(Byte mod	e)
	Before		After			
	Instructio		Instructio			
	VO ABA9	-	WO ABA			
v Data 20	V1 2000 00 D004	-	W1 200 2000 D00	_		
	SR 0000	-		8 (N=1)		
Example 2 C	CP W5,	W6	; Compare W	V6 with W5	(Word mode)	
	Before	_	After			
	Instructio	n	Instruction	on		
١	N5 0004		W6 000			
	V5 2334 V6 8001		W5 2334 W6 8002	_		

CP0		Compare f	with 0x0, Set	Status Flag	S	
Syntax:	{label:}	CP0{.B}	f			
Operands:	f ∈ [0 81	91]				
Operation:	(f) – 0x0					
Status Affected:	DC, N, OV	, Z, C	_			_
Encoding:	1110	0010	OBOf	ffff	ffff	ffff
Description:		f) – 0×0 and units not stored.	•	atus register.	The result of	the
		selects byte o select the add			ord, 1 for byte).
	Note:	The extension rather than a denote a wor	word operat	ion. You may	yuse a .we	•
Words:	1					
Cycles:	1					
Example 1 C	PO.B RA	M100 ; (Compare RAM	1100 with ()x0 (Byte m	iode)
	Before		After			
	Instruction		Instruction	ו		
RAM10	_	RAM		(N = 1)		
3	R 0000		SR 0008	(N=1)		
Example 2 CF	0 0x1FF	E ; Comp	are (0x1FF	E) with 0x	0 (Word mo	de)
	Before		After			
	Instruction	D (/=	Instruction			
Data 1FF		Data 1F				
SI	R 0000	:	SR 0000			

CP0		Compare	Ws with 0x0, \$	Set Status F	lags	
Syntax:	{label:}	CP0{.B}	Ws [Ws] [Ws++] [Ws]			
			[++Ws] [Ws]			
Operands:	Ws ∈ [W0) W15]				
Operation:	(Ws) – 0x	0000				
Status Affected:	DC, N, O	/, Z, C		-		
Encoding:	1110	0000	0000	0B00	0ppp	SSSS
Description:		n is not store	00 and update d. Register dir		-	
	The 'p' bit	s select the s	or word opera source Address address of the	s mode.	-	e).
	Note:	rather than	ion . B in the a word operation,	tion. You ma	yusea.we	-
Words:	1		· · ·		·	
Cycles:	1					
Example 1 CI	90.В [W4-		compare [W4] Post-decreme		Byte mode)	
W			After Instruction W4 1000	I		
Data 100 S		Data 1		(Z=1)		
Example 2 CI	90 [W	15] ; C	Compare [W	5] with 0	(Word mode	e)
W Data 23F		Data 2	After Instruction W5 23FE 3FE 9000	I		

СРВ	Compare	f with WREG	using Borro	ow, Set Statu	is Flags	
Syntax:	{label:}	CPB{.B}	f			
Operands:	f ∈ [081	-				
Operation:	(f) – (WRE	, , ,				
Status Affected:	DC, N, OV	/, Z, C				
Encoding:	1110	0011	1B0f	ffff	ffff	ffff
Description:	instruction	(f) – (WREG) is equivalent n is not stored	to the SUBB i	date the Stat nstruction, b	us register. T ut the result o	This of the
		selects byte of select the ad			ord, 1 for byte	e).
	2:	denote a wo The WREG i The Z flag is	a word operat rd operation, is set to work	tion. You may but it is not re ing register V	y use a .w e equired. V0.	extension to
Words:	1		-			
Cycles:	1					
Example 1 CPF	3.B RAM4	00 ; Compa	re RAM400	with WREG	using \overline{C} (B)	yte mode
WREG RAM400 SR	0823	WRE RAM4 S	00 0823	(N=1)		
Example 2 CPB	0x120)0 ; Compai	ce (0x1200)	with WREG	using \overline{C} (Word mod
WREG Data 1200 SR	Before Instruction 2377 2377 0001 (1	WRE Data 120 C=1) S	00 2377	(C=1)		

© 2003 Microchip Technology Inc.

СРВ	Compare	Wb with lit	5 using Borro	v, Set Status	Flags	
Syntax:	{label:}	CPB{.B}	Wb,	#lit5		
Operands:	Wb ∈ [W0 lit5 ∈ [0	-				
Operation:	(Wb) – lit	$\overline{o} - (\overline{C})$				
Status Affected:	DC, N, O	V, Z, C				
Encoding:	1110	0001	lwww	wB00	011k	kkkk
Description:	is equival	ent to the SU	(C), and updat BB instruction, addressing mu	but the result	of the subtrac	
	The 'B' bi	t selects byte	address of the e or word opera e literal operanc	tion (0 for wo	rd, 1 for byte	
		rather than denote a w The Z flag i	sion .B in the a word operation, ord operation, is "sticky" for AI is can only clear	tion. You may but it is not re	/ use a .w ex quired.	ktension to
Words:	1					
Cycles:	1					
	Before Instruction		Compare W4 After Instruction W4 7711 SR 0008	with 0x12 (N=1)	using C (By	te mode)
Example 2 CP	B.B W4, ‡	‡0x12 ;	Compare W4	with 0x12	using \overline{C} (By	rte mode)
W4 SF	Before Instruction 4 7711 R 0000		After Instruction W4 7711 SR 0008 (N=1)		
Example 3 CP	B W12,	#0x1F ;	Compare W12	with 0x1F	using \overline{C} (We	ord mode)
W12 SF			After Instruction /12 0020 SR 0003 (Z, C=1)		
Example 4 CP	B W12,	#0x1F ;	Compare W12	with 0x1F	using \overline{C} (Wo	ord mode)
W12 SF			After Instruction /12 0020 SR 0001 (C=1)		

Syntax:	{label:}	CPB{.B}	Wb,	Ws		
				[Ws]		
				[Ws++]		
				[Ws]		
				[++Ws]		
				[Ws]		
Operands:	Wb ∈ [W0 Ws ∈ [W0					
Operation:	(Wb) – (W	′s) – (C)				
Status Affected:	DC, N, O\	/, Z, C				
Encoding:	1110	0001	lwww	wB00	0ppp	SSSS
Description:	is equivale stored. Re	ent to the SUB	- (C), and upda BB instruction, I addressing mu y be used for V	but the result st be used for	of the subtrac	tion is not
	The 'B' bit The 'p' bit	selects byte s select the se	ddress of the or word opera ource Address ddress of the V	tion (0 for wor mode.	d, 1 for byte)	
	2:	denote a wo The Z flag is	a word opera ord operation, l s "sticky" for Al can only clear	DDC, CPB,	quired.	
Words:	1					
	1					
Cycles:	I					
Cycles:			Compare [W1] Post-increme		sing C (Byt	e mode)
Cycles:	PB.B W0, Before	; P	Post-increme After	nt Wl	sing C (Byt	e mode)
Cycles: Example 1 C	PB.B W0, Before Instructio	; P	Post-increme After Instruction	nt Wl	sing C̄ (Byt	e mode)
Cycles: Example 1 C	PB.B W0, Before Instruction V0 ABA9	; P	Post-increme After Instruction W0 ABA9	nt Wl	sing C (Byt	e mode)
Cycles: Example 1 C	PB.B W0, Before Instructio V0 ABA9 V1 1000	; P	After Instruction W0 ABA9 W1 1001	nt Wl	sing C̄ (Byt	e mode)
Cycles: Example 1 C V V Data 100	PB.B W0, Before Instruction V0 ABA9 V1 1000 D0 D0A9	; ₽ n]	After Instruction W0 ABA9 W1 1001	nt W1	sing C (Byt	e mode)
Cycles: Example 1 C V Data 100 S	PB.B W0, Before Instruction V0 ABA9 V1 1000 00 D0A9 SR 0002	; P Data ^ (Z=1)	After Instruction W0 ABA9 W1 1001 1000 D0A9	nt W1 (N=1) with W0 us		
Cycles: Example 1 C V Data 100 S	PB.B W0, Before Instruction V0 ABA9 V1 1000 D0 D0A9 SR 0002 PB.B W0, Before	; P Data ^ (Z=1) [W1++] ; Cd ; Pd	After Instruction W0 ABA9 W1 1001 1000 D0A9 SR 0008 ompare [W1] ost-increment After	nt W1 (N=1) with W0 us nt W1		
Cycles: Example 1 C V Data 100 S Example 2 C	PB.B W0, Before Instruction VO ABA9 V1 1000 DO D0A9 SR 0002 PB.B W0, Before Instruction	; P Data ^ (Z=1) [W1++] ; Cd ; Pd	After Instruction WO ABA9 W1 1001 1000 D0A9 SR 0008 ompare [W1] ost-increment After Instruction	nt W1 (N=1) with W0 us nt W1		
Cycles: Example 1 C V Data 100 S	PB.B W0, Before Instruction V0 ABA9 V1 1000 00 D0A9 SR 0002 PB.B W0, Before Instruction V0 ABA9	; P Data ^ (Z=1) [W1++] ; Cd ; Pd	After Instruction W0 ABA9 W1 1001 1000 D0A9 SR 0008 ompare [W1] ost-increment After	nt W1 (N=1) with W0 us nt W1		

Example 3 CPB W4, W5

; Compare W5 with W4 using \overline{C} (Word mode)

(C=1)

I	Before nstructio	า	I	After nstructior	า
W4	4000		W4	4000	
W5	3000		W5	3000	
SR	0001	(C=1)	SR	0001	(

CPSEQ	Compare Wb with Wn, Skip if Equal (Wb = Wn)
Syntax:	{label:} CPSEQ{.B} Wb, Wn
Operands:	Wb ∈ [W0 W15] Wn ∈ [W0 W15]
Operation:	(Wb) – (Wn) Skip if (Wb) = (Wn)
Status Affected:	None
Encoding:	1110 0111 1www wB00 0000 ssss
Description:	Compare the contents of Wb with the contents of Wn by performing the subtraction (Wb) – (Wn), but do not store the result. If (Wb) = (Wn), the next instruction (fetched during the current instruction execution) is discarded and on the next cycle, a NOP is executed instead. If (Wb) \neq (Wn), the next instruction is executed as normal.
	The 'w' bits select the address of the Wb source register. The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 's' bits select the address of the Ws source register.
	Note: The extension .B in the instruction denotes a byte operation rather than a word operation. You may use a .w extension to denote a word operation, but it is not required.
Words:	1
Cycles:	1 (2 or 3 if skip taken)
	02000 HERE: CPSEQ.B W0, W1 ; If W0 = W1 (Byte mode), 02002 GOTO BYPASS ; skip the GOTO 02004 02006 02008 BYPASS: 0200A
N N	Before After Instruction Instruction PC 00 2000 PC 00 2002 V0 1001 W0 1001 V1 1000 W1 1000 SR 0000 SR 0000
0 0	18000 HERE: CPSEQ W4, W8 ; If W4 = W8 (Word mode), 18002 CALL _FIR ; skip the subroutine call 18006 18008
v v	Before Instruction After Instruction PC 01 8000 V4 3344 PC 01 8006 V4 3344 V8 3344 W8 3344 W8 3344 V8 0002 (Z=1) SR 0002 (Z=1)

CPSGT	Siç	gned Compare	e wo with W	n, Skip if Gr	eater Than	(vvd > vvn
Syntax:	{label:}	CPSGT{.B}	Wb,	Wn		
Operands:	$Wb \in [W0]$ $Wn \in [W0]$					
Operation:	(Wb) – (W Skip if (Wt	,				
Status Affected:	None					
Encoding:	1110	0110	0www	wB00	0000	SSSS
Description:	subtraction next instru discarded	he contents of n (Wb) – (Wn), ction (fetched and on the ne: struction is exe	but do not st during the cu xt cycle, a NO	ore the resul rrent instruct P is executed	t. If (Wb) > (ion executio	Wn), the n) is
	The 'B' bit	s select the ad selects byte o s select the add	r word operat	ion (0 for wo	rd, 1 for byte	e).
	Note:	The extensio rather than a denote a wor	word operati	on. You may	use a .we	•
Words:	1		. operation, .		4	
Cycles:	-	skip taken)				
Example 1	002000 HERE 002002 002006 002008 00200A BYPA 00200C	GOTO 		If WO > W1 skip the G	-	le),
	Before Instruction PC 0.0 2.0 W0 0.0 0.0 W1 2.6 SR 0.0	00 FF	PC W0 W1 SR	After Instruction 00 2006 00FF 26FE 0009	(N, C=1)	
Example 2	018000 HERE: 018002 018006 018008	CPSGT CALL 	W4, W5; If _FIR ; sk	W4 > W5 (ip the sub		
	Before Instruction PC 01 800 W4 260 W5 260 SR 000	0 0	PC W4 W5 SR	After nstruction 2600 2600 0004 (0	OV=1)	

CPSL1	Signed Compare Wb with Wn, Skip if Less Than (Wb < W
Syntax:	{label:} CPSLT{.B} Wb, Wn
Operands:	Wb ∈ [W0 W15] Wn ∈ [W0 W15]
Operation:	(Wb) – (Wn) Skip if (Wb) < (Wn)
Status Affecte	d: None
Encoding:	1110 0110 1www wB00 0000 ssss
Description:	Compare the contents of Wb with the contents of Wn by performing the subtraction (Wb) – (Wn), but do not store the result. If (Wb) < (Wn), the next instruction (fetched during the current instruction execution) is discarded and on the next cycle, a NOP is executed instead. Otherwise, th next instruction is executed as normal.
	The 'w' bits select the address of the Wb source register. The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 's' bits select the address of the Ws source register.
	Note: The extension .B in the instruction denotes a byte operation rather than a word operation. You may use a .W extension denote a word operation, but it is not required.
Words:	1
Cycles:	1 (2 or 3 if skip taken)
Example 1	002000 HERE: CPSLT.B W8, W9; If W8 < W9 (Byte mode),
	Before After Instruction Instruction PC 00 2000 PC 00 2002 W8 00FF W8 00FF W9 26FE W9 26FE SR 0008 SR 0008
Example 2	018000 HERE: CPSLT W3, W6 ; If W3 < W6 (Word mode), 018002 CALL _FIR ; skip the subroutine call 018006 018008
	Before Instruction After Instruction PC 01 8000 PC 01 8006 W3 2600 W3 2600 W6 3000 W6 3000 SR 0000 SR 0000

CPSNE		Signed Con	npare Wb wi	th Wn, Skip i	f Not Equal (Wb ≠ Wn)
Syntax:	{label:}	CPSNE{.B}	Wb,	Wn		
Operands:	$Wb \in [W0 . Wn \in [W0 .$	-				
Operation:	(Wb) – (Wn Skip if (Wb)	•				
Status Affected	: None					
Encoding:	1110	0111	0www	wB00	0000	SSSS
Description:	subtraction instruction (and on the	(Wb) – (Wn), (fetched durin	but do not st g the current NOP is execu	contents of W ore the result. instruction ex ted instead. C	If (Wb) ≠ (Wr ecution) is di	n), the next scarded
	The 'B' bit s	elects byte o	r word operat	Wb source reg tion (0 for wor Vs source reg	d, 1 for byte).	
		rather than a	word operation	instruction de tion. You may out it is not rec	/usea.we	
Words:	1		•			
Cycles:	1 (2 or 3 if s	skip taken)				
	002000 HERE: 002002 002006 002008 00200A BYPAS 00200C	GOTO 		If W2 != W skip the G	-	de),
	Before Instruction PC 0.0 2.00 W2 0.0F W3 2.6F SR 0.00	00 'F 'E	PC W2 W3 SR	After Instruction 00 2006 00FF 26FE 0001 (C=1)	
·	018000 HERE: 018002 018006 018008			If WO != W8 skip the su		
	Before Instruction PC 01 800 W0 300 W8 300 SR 000	000000	PC W0 W8 SR	After Instruction 01 8002 3000 3000 0000		

DAW.B		Decimal Au	ljust Wn			
Syntax:	{label:}	DAW.B	Wn			
Operands:	Wn ∈ [W0) W15]				
Operation:	•)> > 9) or (DC	,			
	(vvn<3: Else	$(0>) + 6 \rightarrow Wn$	<3:0>			
	(Wn<3:	$0>) \rightarrow Wn<3:0$)>			
		4> > 9) or (C = 4>) + 6 → Wn				
	(Wn<7:	4>) → Wn<7:4	1>			
Status Affected:	С		1			
Encoding:	1111	1101	0100	0000	0000	SSSS
	addressin	is used to ind g must be use s select the ac	d for Wn.		-	
	The 's' bit	s select the ac	Idress of the	source/destir	nation registe	r.
	Note 1:	This instruct	ion is used t	to correct the	e data forma	t after two
	2:	This instruc		s in Byte m		nd the .B
Words:	1	extension m	ust be include	ed with the op	ocode.	
Cycles:	1					
Cyclos.	·					
Example 1	DAW.B WO	; Dec	imal adjus	t WO		
	Before		After			
	Instructio		Instruction	1		
	W0 771A		WO 7720	(DC-1)		
	SR 0002	(DC=1)	SR 0002	(DC=1)		
Example 2	DAW.B W3	; Dec	imal adjus	t W3		
	Before		After			
	W3 77AA SR 0000		V3 7710 SR 0001 (0	C=1)		

5

DEC		Decrement	f			
Syntax:	{label:}	DEC{.B}	f	{,WREG}		
Operands:	f ∈ [0 81	-				
Operation:		estination des	ignated by D			
Status Affected:	DC, N, OV,	Z, C				
Encoding:	1110	1101	OBDf	ffff	ffff	ffff
Description:	destination tion registe	e from the co register. The r. If WREG is fied, the resul	optional WR specified, th	EG operand e result is sto	determines the red in WREC	he destina-
	The 'D' bit	selects byte o selects the de select the ado	stination (0 f	or WREG, 1 f		
		The extensio rather than a denote a wor The WREG is	word operat d operation, l	tion. You may but it is not re	/ use a .w ex quired.	•
Words:	1					
Cycles:	1					
Example 1	DEC.B 0x20	0	; Decrem	ent (0x200) (Byte mod	le)
Data 2	Before Instruction 200 80FF SR 0000	Data 2	After Instruction 200 80FE SR 0009	n (N,C=1)		
Example 2	DEC RAM4	00, WREG ; ;	Decrement (Word mod		nd store to	WREG
WRE RAM4		WRE RAM40 S				

DEC		Decrement	Ws			
Syntax:	{label:}	DEC{.B}	Ws,	Wd		
			[Ws],	[Wd]		
			[Ws++],	[Wd++]		
			[Ws],	[Wd]		
			[++Ws],	[++Wd]		
			[Ws],	[Wd]		
Operands:	Ws ∈ [W0 Wd ∈ [W0	-				
Operation:	(Ws) – 1 –	→ Wd				
Status Affected:	DC, N, OV	, Z, C				
Encoding:	1110	1001	0Bqq	qddd	dppp	SSSS
Description:	result in th	e destination	ontents of the register Wd. d by Ws and V	Either register		
	The 'q' bits The 'd' bits The 'p' bits	s select the de s select the ac s select the so	or word opera estination Add ddress of the ource Address ddress of the	lress mode. destination re s mode.	gister.	
	ra	ather than a v	. B in the inst vord operatior operation, bu	n. You may us	e a .w extens	
Words:	1		,			
Cycles:	1					
Example 1 DEC	.B [W7++]], [W8++]		and store crement W7		yte mod
	Before		After			
	nstruction		Instruction			
W7	nstruction 2301		Instruction			
W7 W8	nstruction 2301 2400	V	Instruction V7 2302 V8 2401			
W7 W8 Data 2300	nstruction 2301 2400 5607	W Data 230	Instruction V7 2302 V8 2401 00 5607			
W7 W8	nstruction 2301 2400	W Data 230 Data 240	Instruction V7 2302 V8 2401 00 5607			
W7 W8 Data 2300 Data 2400	nstruction 2301 2400 5607 ABCD 0000	W Data 23(Data 24(S +++] ; Dec	Instruction V7 2302 V8 2401 00 5607 00 AB55		o [W6] (Wo	rd mode)
W7 W8 Data 2300 Data 2400 SR Example 2 DEC	nstruction 2301 2400 5607 ABCD 0000 W5, [W6 Before	W Data 23(Data 24(S +++] ; Dec	Instruction V7 2302 V8 2401 00 5607 00 AB55 5R 0000 crement W5 a st-increment After		o [W6] (Wo	rd mode)
W7 W8 Data 2300 Data 2400 SR Example 2 DEC	nstruction 2301 2400 5607 ABCD 0000 W5, [W6 Before nstruction	W Data 230 Data 240 S ++] ; Dec ; Pos	Instruction V7 2302 V8 2401 00 5607 00 AB55 SR 0000 erement W5 ast-increment After Instruction		.o [W6] (Wo	rd mode)
W7 W8 Data 2300 Data 2400 SR Example 2 DEC	nstruction 2301 2400 5607 ABCD 0000 W5, [W6 Before nstruction D004	W Data 23(Data 24(S +++] ; Dec ; Pos	Instruction V7 2302 V8 2401 00 5607 00 AB55 SR 0000 crement W5 After Instruction V5 D004		.o [W6] (Wo	rd mode)
W7 W8 Data 2300 Data 2400 SR Example 2 DEC	nstruction 2301 2400 5607 ABCD 0000 W5, [W6 Before nstruction	W Data 23(Data 24(S +++] ; Dec ; Pos	Instruction V7 2302 V8 2401 00 5607 00 AB55 SR 0000 crement W5 St-increment After Instruction V5 D004 V6 2002		o [W6] (Wo	rd mode)

DEC2		Decrement	f by 2			
Syntax:	{label:}	DEC2{.B}	f	{,WREG}		
Oporanda:	f ∈ [0 81	011				
Operands:	-	-	invested by D			
Operation:	.,	estination des	Ignated by D			
Status Affected:	DC, N, OV		_			
Encoding:	1110	1101	1BDf	ffff	ffff	ffff
Description:	destination tion registe	o from the co register. The r. If WREG is ified, the resu	optional WR specified, th	EG operand e result is sto	determines tl pred in WREC	he destina-
	The 'D' bit	selects byte o selects the de select the ado	stination (0 f	or WREG, 1 f		
	ra	ne extension ther than a we enote a word o	ord operation	. You may us	e a .w exten	
Words:	1					
Cycles:	1					
Example 1 DI	EC2.B 0x	200	; Decreme	ent (0x200)	by 2 (Byt	e mode)
Data 20 S		Data 2 S		(N, C=1)		
Example 2 D	EC2 RA	M400, WREG		ent RAM400 to WREG (Wo	-	
WRE RAM40 S		WR RAM4		n		

DEC2		Decrement	t Ws by 2			
Syntax:	{label:}	DEC2{.B}	Ws,	Wd		
			[Ws],	[Wd]		
			[Ws++],	[Wd++]		
			[Ws],	[Wd]		
			[++Ws],	[++Wd]		
			[Ws],	[Wd]		
Operands:	Ws ∈ [W0 Wd ∈ [W0	-				
Operation:	(Ws) – 2 -	→ Wd				
Status Affected:	DC, N, O	/, Z, C				
Encoding:	1110	1001	1Bqq	qddd	dppp	SSSS
Description:	result in th	ne destination	ontents of the register Wd. d by Ws and V	Either registe		
	The 'q' bit The 'd' bit The 'p' bit	s select the des select the action of the select the action of the select the	or word opera estination Add ddress of the ource Address ddress of the	lress mode. destination re s mode.	egister.	·).
	r	ather than a v	. B in the inst vord operatior operation, bu	n. You may us	se a .w exter	
Words:	1					
Cycles:	1					
Example 1 DEC	2.B [W7		DEC [W7] by Post-decre	2, store ment W7, W8		yte mode
			robe accre			
	Before		After			
	Instruction	14	After Instruction			
W7	Instruction 2301		After Instruction			
W7 W8	Instruction 2301 2400	W	After Instruction /7 2300 /8 23FF			
W7	Instruction 2301 2400 0107		After Instruction /7 2300 /8 23FF 00 0107			
W7 W8 Data 2300	Instruction 2301 2400 0107 ABCD	W Data 230 Data 240	After Instruction /7 2300 /8 23FF 00 0107	V=1)		
W7 W8 Data 2300 Data 2400	Instruction 2301 2400 0107 ABCD 00000	W Data 230 Data 240 S	After Instruction /7 2300 /8 23FF 00 0107 00 ABFF	store to [mode)
W7 W8 Data 2300 Data 2400 SR	Instruction 2301 2400 0107 ABCD 0000 3C2 W5, Before	W Data 230 Data 240 S	After Instruction /7 2300 /8 23FF 00 0107 00 ABFF 00 ABFF 00 0008 (N C W5 by 2, st-incremen After	store to [mode)
W7 W8 Data 2300 Data 2400 SR Example 2 DE	Instruction 2301 2400 0107 ABCD 0000 3C2 Before Instruction	W Data 230 Data 240 S N6++] ; DE0 ; Pos	After Instruction /7 2300 /8 23FF 00 0107 00 ABFF 00 ABFF 00 008 (N C W5 by 2, st-incremen After Instruction	store to [mode)
W7 W8 Data 2300 Data 2400 SR	Instruction 2301 2400 0107 ABCD 0000 3C2 W5, [W Before Instruction 5 D004	W Data 230 Data 240 S N6++] ; DE0 ; Pos	After Instruction /7 2300 /8 23FF 00 0107 00 ABFF 00 ABFF 00 0008 (N C W5 by 2, st-incremen After	store to [mode)
W7 W8 Data 2300 Data 2400 SR Example 2 DE	Instruction 2301 2400 0107 ABCD 0000 C2 W5, [W Before Instruction 5 D004 6 1000	W Data 230 Data 240 S N6++] ; DE0 ; Pos	After Instruction /7 2300 /8 23FF 00 0107 00 ABFF 00 ABFF 00 0008 (N C W5 by 2, st-incremen After Instruction V5 D004 V6 1002	store to [mode)

DISI		Disable Inte	errupts Tem	oorarily		
Syntax:	{label:}	DISI	#lit14			
Operands:	lit14 ∈ [0	-				
Operation:	lit14 \rightarrow DIS 1 \rightarrow DISI Disable inte	ICNT errupts for (lit	14+1) cycles			
Status Affected:	None					
Encoding:	1111	1100	00kk	kkkk	kkkk	kkkk
Description:	cycles after DISICNT re instruction effects of in	the instruction egister, and the can be used aterrupts.	on executes. ne DISI flag (before execu	The lit14 val INTCON2<14 Iting time criti	r (lit14+1) ins ue is written t 4>) is set to ' ical code, to l	to the 1'. This imit the
				•	ty 7 interrupts Sheet for de	
Words:	1					
Cycles:	1					
0	02000 HERE: 02002 02004	DISI #1			upts for 10 ime critica	-
	Before			After		
-	Instruction	7		nstruction		
H DISICI	C 00 2000	_	PC DISICNT	00 2002		
INTCO			INTCON2		DISI=1)	
S	SR 0000	,	SR	0000	. ,	

DIV.S		Signed Int	eger Divide			
Syntax:	{label:}	DIV.S{W}	Wm, Wn			
		DIV.SD	Wm, Wn			
Operands:	Wm ∈ [W		word operatic W14] for doul			
Operation:	Wm — <u>If (Wm</u> 0xFl <u>Else:</u> 0x0 W1:W0	operation (de → W0 <15> = 1): FFF → W1 → W1 0 / Wn → W0 nder → W1	<u>fault):</u>			
	Wm+1 W1:W0	le operation ([:Wm → W1:W) / Wn → W0 nder → W1	•			
Status Affected:	N, OV, Z,	С				
Encoding:	1101	1000	Ottt	tvvv	vW00	SSSS
	operation divide ope	, Wm+1:Wm i eration is store	through W1 to s first copied t ed in W0, and	to W1:W0. The the 16-bit rer	he 16-bit quot mainder is sto	tient of tl red in W
	(with an it remainde otherwise overflow a and clear	teration count r. The N flag v e. The OV flag and cleared of ed otherwise.	e executed 18 of 17) to gene will be set if th will be set if t therwise. The The C flag is value should n	erate the corr e remainder he divide ope Z flag will be used to imple	ect quotient a is negative ar eration resulte set if the rem	and nd cleare ed in an ainder is
	(with an it remainde otherwise overflow a and clear algorithm The 't' bit operation The 'v' bit The 'W' b The 's' bit	teration count r. The N flag v e. The OV flag and cleared of ed otherwise. and its final v s select the M . These bits a ts select the L it selects the ts select the d	of 17) to gene will be set if th will be set if t therwise. The The C flag is	erate the corr e remainder he divide ope Z flag will be used to imple ot be used. t Word of the e word opera nt Word of th (0 for 16-bit,	rect quotient a is negative ar eration resulte set if the rem ement the div dividend for tion. e dividend. 1 for 32-bit).	and nd cleare ed in an ainder is ide the doub

DIV.S	Si	gned Integer Divide
Words:	1	
Cycles:	18 (plus 1 for F	REPEAT execution)
Example 1	REPEAT #17 DIV.S W3, W4	; Execute DIV.S 18 times ; Divide W3 by W4 ; Store quotient to W0, remainder to W1
	Before Instruction W0 5555 W1 1234 W3 3000 W4 0027 SR 0000	After Instruction W0 013B W1 0003 W3 3000 W4 0027 SR 0000
Example 2	REPEAT #17 DIV.SD W0, W12	; Execute DIV.SD 18 times ; Divide W1:W0 by W12 ; Store quotient to W0, remainder to W1
	Before Instruction W0 2500 W1 FF42 W12 2200 SR 0000	After Instruction W0 FA6B W1 EF00 W12 2200 SR 0008 (N=1)

Syntax:	{label:}	DIV.U{W}	Wm, Wn			
- ,	()	DIV.UD	Wm, Wn			
Operands:		0, W2, W4	word operatic W14] for doul			
Operation:	$Wm \rightarrow 0x0 \rightarrow W1:WC$		fault):			
	Wm+1: W1:W0	e operation ([Wm → W1:V) / Wns → W0 nder → W1	VO			
Status Affected:	N, OV, Z,	С				
Encoding:	1101	1000	1ttt	tvvv	vW00	SSSS
	operation This instru (with an it remainde divide ope will be set	is stored in W uction must be eration count r. The N flag v eration resulte t if the remain	ed to W1:W0. V0, and the 16 e executed 18 of 17) to gene vill always be ed in an overflo der is 0 and c algorithm and	-bit remainde times using erate the corr cleared. The bw and cleare leared otherv	er is stored in the REPEAT ect quotient a OV flag will b ed otherwise. vise. The C fl	W1. instructic and be set if th The Z fla ag is use
	The 't' bits	s select the M	1.01 .0			be used
	The 'v' bit The 'W' b	. These bits a s select the L it selects the c	ost Significan re clear for the east Significa dividend size ivisor register.	nt Word of the (0 for 16-bit,	tion. e dividend.	
	The 'v' bit The 'W' b The 's' bit Note 1: 2: 3:	These bits a s select the L it selects the d s select the d The extens (32-bit) divid extension to Unexpected operation (I bit will be s used. Dividing by first cycle of This instruct	re clear for the east Significat dividend size ivisor register. ion . D in the dend rather tha denote a wor d results will d in 16 bits. DIV.UD). Whe set and the qu	e word opera nt Word of the (0 for 16-bit, an a word divi rd operation, occur if the This may on en an overflo iotient and re te an arithme	tion. e dividend. 1 for 32-bit). denotes a d dend. You m but it is not r e quotient c ly occur for w occurs, the emainder sho	ouble-wo ay use a equired. an not the dout e OV stat ould not
Words:	The 'v' bit The 'W' b The 's' bit Note 1: 2: 3:	These bits a s select the L it selects the d s select the d The extens (32-bit) divid extension to Unexpected represented operation (I bit will be s used. Dividing by first cycle of	re clear for the east Significat dividend size of ivisor register. ion . D in the dend rather that denote a word results will d in 16 bits. DIV.UD). Who set and the qu zero will initia f execution.	e word opera nt Word of the (0 for 16-bit, an a word divi rd operation, occur if the This may on en an overflo iotient and re te an arithme	tion. e dividend. 1 for 32-bit). denotes a d dend. You m but it is not r e quotient c ly occur for w occurs, the emainder sho	ouble-wo ay use a equired. an not the dout e OV stat ould not

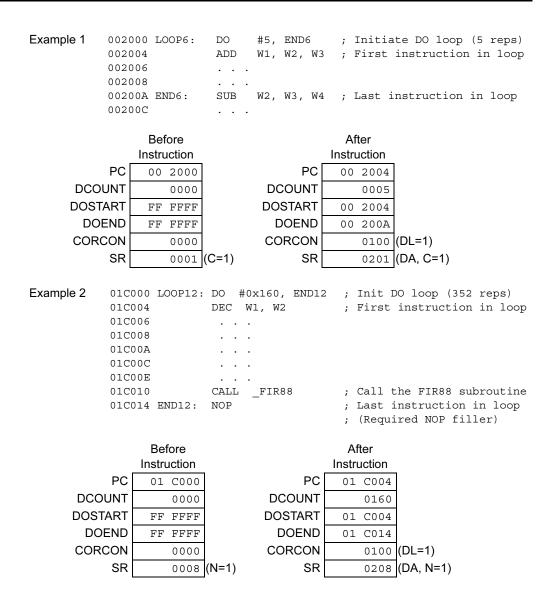
	REPEAT #17 DIV.U W2, W4	; Execute DIV.U 18 times ; Divide W2 by W4 ; Store quotient to W0, remainder to W1
	Before Instruction W0 5555 W1 1234 W2 8000 W4 0200 SR 0000	After Instruction W0 0040 W1 0000 W2 8000 W4 0200 SR 0002 (Z=1)
	REPEAT #17	; Execute DIV.UD 18 times
1	DIV.UD W10, W12	; Divide W11:W10 by W12 ; Store quotient to W0, remainder to W1
	Before Instruction	After Instruction
١	W0 5555	W0 01F2
١	W1 1234	W1 0100
W	/10 2500	W10 2500
	/11 0042	W11 0042
	/12 2200	W12 2200
	SR 0000	SR 0000

DIVF		Fractiona	Divide			
Syntax:	{label:}	DIVF	Wm, Wn			
Operands:	Wm ∈ [W0 Wn ∈ [W2					
Operation:	$0x0 \rightarrow W0$ $Wm \rightarrow W1$ W1:W0 / V Remainde	$Vn \rightarrow W0$				
Status Affected:	N, OV, Z, (С				
Encoding:	1101	1001	Ottt	t000	0000	SSSS
Description:	stored in V W0 is first divide ope The sign c	Vm and the cleared and ration is stor f the remain	nal 16-bit by 1 divisor is stored Wm is copied ed in W0, and der will be the	d in Wn. To p to W1. The 1 the 16-bit rer same as the	erform the op 16-bit quotien nainder is sto sign of the d	peration, it of the pred in W1 ividend.
	(with an ite remainder otherwise. overflow a and cleare	eration coun . The N flag . The OV flag nd cleared o d otherwise	e executed 18 t of 17) to gene will be set if the will be set if the will be set if the therwise. The The C flag is value should no	erate the corr e remainder i he divide ope Z flag will be used to imple	ect quotient a is negative an eration resulte set if the rem	and nd cleared ed in an nainder is
			ividend registe livisor register.			
	2:	or equal to will occur b When this of and remain Dividing by first cycle of	ctional divide to Wn. If Wm is ecause the fra- occurs, the OV der should not zero will initia f execution. Inction is inter	greater than ctional result status bit wil be used. te an arithme	Wn, unexpea will be great I be set and t etic error trap	cted resul er than 1.0 he quotien during th
Words:	1	,				
Cycles:	18 (plus 1	for REPEAT	execution)			
	REPEAT #17 DIVF W8,	W9 ; D	xecute DIVF ivide W8 by tore quotien	W9	remainder	to W1
V V V	Before Instruction V0 8000 V1 1234 V8 1000 V9 4000 SR 0000		After Instruction W0 2000 W1 0000 W8 1000 W9 4000 SR 0002 (2	Z=1)		

Example 2	REPEAT #17 DIVF W8, W9	; Execute DIVF 18 times ; Divide W8 by W9 ; Store quotient to W0, remainder to W1
	Before Instruction W0 8000 W1 1234 W8 1000 W9 8000 SR 0000	After Instruction W0 F000 W1 0000 W8 1000 W9 8000 SR 0002 (Z=1)
Example 3	REPEAT #17 DIVF W0, W1	; Execute DIVF 18 times ; Divide W0 by W1 ; Store quotient to W0, remainder to W1
	Before Instruction W0 8002 W1 8001 SR 0000	After Instruction W0 7FFE W1 8002 SR 0008 (N=1)

DO	Initialize Hardware Loop Literal						
Syntax:	{label:}	DO	#lit14,	Expr			
Operands:		16383] e an absolute lved by the lir				+32767	
Operation:	$\begin{array}{l} (lit14) \to DC \\ (PC) + 4 \to F \\ (PC) \to DO \\ (PC) + (2^*S) \end{array}$	РС	ND)		
Status Affected:	DA						
Encoding:	0000	1000	00kk	kkkk	kkkk	kkkk	
	0000	0000	nnnn	nnnn	nnnn	nnnn	
	the address supports a i	p begins at the 2*Slit16 instr maximum loop ports offsets o	uction words count value	away. The 14 of 16384, an	4-bit count va d the 16-bit o	llue (lit14) ffset value	
	pushed into new DO loc DL<2:0> (C completes e	nstruction exe their respect p parameters ORCON<8:10 execution, the e restored, an	ive shadow r specified by 0>), is then ir pushed DC0	egisters, and the instructio ncremented. A DUNT, DOST	then updated n. The DO le After the DO l ART and DOI	l with the vel count, oop	
	The 'k' bits specify the loop count. The 'n' bits are a signed literal that specifies the number of instructions offset from the PC to the last instruction executed in the loop.						
	 Special Features, Restrictions: The following features and restrictions apply to the DO instruction. Using a loop count of 0 will result in the loop being executed one time. Using a loop size of -2, -1 or 0 is invalid. Unexpected results may occur if these offsets are used. The very last two instructions of the DO loop can NOT be: an instruction which changes program control flow a DO or REPEAT instruction 						
	Unexpected results may occur if any of these instructions are used.						
	Note 1:	The DO instr hardware nes	ruction is in	terruptible ar	nd supports	1 level o	
	2:	provided in s Reference Ma The linker will be used.	oftware by anual for deta	the user. See ails.	e the dsPIC3	30F Fami	
Words:	2:	provided in s Reference Ma The linker will	oftware by anual for deta	the user. See ails.	e the dsPIC3	30F Famil	

dsPIC30F Programmer's Reference Manual



		Initialize Ha		Wn		
Syntax:	{label:}	DO	Wn,	Expr		
Operands:		W15] e an absolute blved by the lir				+32767
Operation:	$\begin{array}{l} (Wn) \rightarrow DC \\ (PC)+4 \rightarrow I \\ (PC) \rightarrow DC \\ (PC) + (2^{*}S) \end{array}$	PC	ND	·		
Status Affected:	DA					
	0000	1000	1000	0000	0000	SSSS
Encoding:	0000	0000	nnnn	nnnn	nnnn	nnnn
	When this i	ports offsets on nstruction exe their respection	cutes, DCOL	JNT, DOSTAF	RT and DOEN	D are first
	new DO loc DL<2:0> (C completes of	 their respection p parameters ORCON<8:10 execution, the 	specified by)>), is then in	the instruction cremented. A	n. The DO lev fter the DO lo	vel count, oop
						ND
	The 's' bits The 'n' bits	e restored, an specify the re are a signed I (PC+4), which	d DL<2:0> is gister Wn tha iteral that spe	decremented t contains the ecifies the nu	l. loop count. mber of instru	ctions
	The 's' bits The 'n' bits offset from Special Fe The followin 1. Using a 2. Using a if these 3. The ve • an ir	specify the rea	d DL<2:0> is gister Wn tha iteral that spe- n is the last in ictions: d restrictions f 0 will result i -1 or 0 is inv sed. structions of t ch changes po	decremented t contains the ecifies the nur struction exe apply to the n the loop bei alid. Unexpect	d. e loop count. mber of instruction cuted in the lo DO instruction ng executed o cted results m an NOT be:	ctions bop. one time.
	The 's' bits The 'n' bits offset from Special Fe The followin 1. Using a 2. Using a if these 3. The ve • an ir • a DO	specify the re- are a signed I (PC+4), which atures, Restr ng features an a loop count of an offset of -2, offsets are us ry last two in- struction which	d DL<2:0> is gister Wn tha iteral that spe- n is the last in ictions : d restrictions f 0 will result i -1 or 0 is inv sed. structions of t ch changes po- struction	decremented t contains the ecifies the nur struction exe apply to the n the loop bei ralid. Unexpect the DO loop of rogram contro	d. e loop count. mber of instruction cuted in the lo DO instruction ng executed of cted results m an NOT be: ol flow	ctions bop. one time. ay occur
	The 's' bits The 'n' bits offset from Special Fe The followin 1. Using a 2. Using a if these 3. The ve • an ir • a DO Unexp Note 1:	specify the re- are a signed I (PC+4), which atures, Restr ng features an a loop count of an offset of -2, e offsets are us ry last two ins nstruction which or REPEAT in	d DL<2:0> is gister Wn tha iteral that spe- n is the last in ictions : d restrictions f 0 will result i -1 or 0 is inv sed. structions of t struction nay occur if t ction is interru an additional See the dsP	decremented t contains the ecifies the nur struction exe apply to the n the loop bei ralid. Unexpect the DO loop of rogram control hese last inst uptible and su 5 levels may IC30F Family	d. a loop count. mber of instru- cuted in the lo DO instruction ng executed of cted results m an NOT be: an NOT be: of flow ructions are u pports 1 level be provided y Reference	ctions bop. one time. ay occur used. of nesting in softwar Manual fo
Words:	The 's' bits The 'n' bits offset from Special Fe The followin 1. Using a 2. Using a if these 3. The ve • an ir • a DO Unexp Note 1:	specify the re- are a signed I (PC+4), which atures, Restr ng features an a loop count of an offset of -2, e offsets are us ry last two ins astruction which or REPEAT in ected results r The DO instruct Nesting up to by the user. details. The linker will	d DL<2:0> is gister Wn tha iteral that spe- n is the last in ictions : d restrictions f 0 will result i -1 or 0 is inv sed. structions of t struction nay occur if t ction is interru an additional See the dsP	decremented t contains the ecifies the nur struction exe apply to the n the loop bei ralid. Unexpect the DO loop of rogram control hese last inst uptible and su 5 levels may IC30F Family	d. a loop count. mber of instru- cuted in the lo DO instruction ng executed of cted results m an NOT be: an NOT be: of flow ructions are u pports 1 level be provided y Reference	ctions bop. one time. ay occur used. of nesting in softwar Manual fo

Instruction Descriptions

dsPIC30F Programmer's Reference Manual

0020 0020 0020 0020 0020 0020 0020	006 008 00A 00C	DO W0, END6 ; Initiate DO loop (W0 reps) ADD W1, W2, W3 ; First instruction in loop REPEAT #6 SUB W2, W3, W4 NOP ; Last instruction in loop ; (Required NOP filler)
PC	Before Instruction	After Instruction PC 00 2004
W0	0012	W0 0012
DCOUNT	0000	DCOUNT 0012
DOSTART	FF FFFF	DOSTART 00 2004
DOEND	FF FFFF	DOEND 00 2010
CORCON	0000	CORCON 0100 (DL=1)
SR	0000	SR 0080 (DA=1)
0020 0020 0020 0020	006 008	DO W7, ENDA ; Initiate DO loop (W7 reps) SWAP W0 ; First instruction in loop MOV W1, [W2++] ; Last instruction in loop
	Before	After
	Instruction	Instruction
PC	00 2000	PC 00 2004
W7	EOOF	W7 EOOF
DCOUNT	0000	DCOUNT 200F
DOSTART	FF FFFF	DOSTART 00 2004
DOEND	FF FFFF	
CORCON	0000	CORCON 0100 (DL=1)
SR	0000	SR 0080 (DA=1)

		Euclidean	Distance (N	o Accumula	ite)	
Syntax: {	abel:} ED	Wm*Wm,	Acc,	[Wx], [Wx]+=kx, [Wx]-=kx, [W9+W12],	[Wy], [Wy]+=ky, [Wy]-=ky, [W11+W12]	Wxd I,
Operands:	Wx ∈ [N Wy ∈ [N	A,B] n ∈ [W4*W4, W /8, W9]; kx ∈ [- /10, W11]; ky ∈ W4 W7]	6, -4, -2, 2, 4	, 6]		
Operation:			or B)			
Status Affected	OA, OB,	OAB	-			
Encoding:	1111	0 0 mm	Alxx	00ii	iijj	jj11
	sign-externe results o Operanc support	n values specifi ended to 40-bit f [Wx] – [Wy] a ds Wx, Wxd an indirect and reg 4.14.1 "MAC	s and stored ire stored in V d Wyd specif gister offset a	in the specif Vxd, which n y the pre-feto ddressing as	ed accumulated accumula	ator. The ame as Wr s which
	The 'A' b The 'x' b The 'i' bi	bits select the o bit selects the a bits select the p tits select the W tits select the W	ccumulator for re-fetch differ x pre-fetch o	or the result. ence Wxd d peration.		
Words:	1					
Cycles:	1					
Example 1	ED W4*W4, 2	A, [W8]+=2,	[W10]-=2,	; [W8]-[; Post-i	e W4 to ACC W10] to W4 ncrement W ecrement W	18
	Befo	re		After		
	Instruc		14/4	Instruct		
	W4 W8	009A 1100	W4 W8		0057 1102	
	V10	2300	W10		22FE	
V	-			00 0000		
	CA 00 3DOA	0000	ACCA	00 0000	5CA4	
		0000 007F	Data 1100		007F	
AC	100					

Example 2 ED W5*W5, B, [W9]+=2, [W11+W12], W5 ; Square W5 to ACCB

; [W11+W12]-[W9] to W5 ; Post-increment W9

	Before Instruction
W5	43C2
W9	1200
W11	2500
W12	0008
ACCB	00 28E3 F14C
Data 1200	6A7C
Data 2508	2B3D
SR	0000

	I	After nstruct	
W5			3F3F
W9			1202
W11			2500
W12			0008
ACCB	00	11EF	1F04
Data 1200			6A7C
Data 2508			2B3D
SR			0000

EDAC		Euclidean	Distance			
Syntax: {label:}	EDAC	Wm*Wm,	Acc,	[Wx], [Wx]+=kx, [Wx]-=kx, [W9+W12],		Wxd
Operands:	Wx ∈ [W8,	: [W4*W4, W W9]; kx ∈ [·), W11]; ky ∈	-6, -4, -2, 2,	· •		
Operation:	(Acc(A or E ([Wx]–[Wy] (Wx)+kx→ (Wy)+ky→	Wx	Wm) → Acc	(A or B)		
Status Affected:	OA, OB, O	AB, SA, SB,	SAB			
Encoding:	1111	0 0 mm	Alxx	00ii	iijj	jj10
Description:	values spe sign-exten results of [Operands	cified by [W: ded to 40-bit Wx] – [Wy] a Wx, Wxd an	x] and [Wy]. s and addeo re stored in d Wyd speci	lso the differe The results o to the speci Wxd, which ify the pre-fet	of Wm*Wm a fied accumul may be the s inch operation	re ator. The ame as Wr s which
	Section 4. The 'm' bit The 'A' bit The 'x' bits The 'i' bits	14.1 "MAC s select the a selects the a	Pre-Fetches operand reg occumulator re-fetch diffe /x pre-fetch	ister Wm for for the result erence Wxd o operation.	the square.	
Words:	1					
Cycles:	1					
Example 1 EDAC	C W4*W4,	A, [W8]+=:	2, [w10]-=	2, W4	Square W add to A [W10]-[W Post-inc: Post-deci	CCA 8] to W4 rement W8
	Before Instructi			After Instructi	on	
W4	monucu	009A	W4		0057	
W8		1100	W8		1102	
W10		2300	W10		22FE	
ACCA	00 3D0A	2007	ACCA	00 3D0A	0075	
ЛООЛ	00 JD0A	SDUA	ACCA	UU SDUA	99AL	
Data 1100 Data 2300	UU JDUA	007F	Data 1100 Data 2300	00 SDOA	007F	

SR

0000

SR

0000

Example 2 EDAC W5*W5, B, [w9]+=2, [W11+W12], W5 ; Square W5 and

; Square W5 and ; add to ACCB ; [W9]-[W11+W12] to W5 ; Post-increment W9

	Before Instruction				
W5			43C2		
W9			1200		
W11			2500		
W12			0008		
ACCB	00	28E3	F14C		
Data 1200			6A7C		
Data 2508			2B3D		
SR			0000		

	After Instruction				
W5			3F3F		
W9			1202		
W11			2500		
W12			0008		
ACCB	00	3AD3	1050		
Data 1200			6A7C		
Data 2508			2B3D		
SR			0000		

EXCH		Exchange V	Nns and W	nd		
Syntax:	{label:}	EXCH	Wns,	Wnd		
Operands:	Wns ∈ [W Wnd ∈ [W					
Operation:	$(Wns) \leftrightarrow ($	Wnd)				
Status Affected:	None					
Encoding:	1111	1101	0000	0ddd	d000	SSS
Description:		the word cont must be use		working regis nd Wnd.	ters. Register	direct
				e first register. e second regis	ter.	
	Note: ⊤	his instruction	only execu	tes in Word m	ode.	
Words:	1					
Cycles:	1					
Example 1	EXCH W1, W9) ; Exc	hange the	contents o	of W1 and W	9
	Before		A	fter		
	Instructio	n		ruction		
	W1 55F		W1	A3A3		
	W9 A3A		W9	55FF		
	SR 000	0	SR	0000		
Example 2	EXCH W4, W5	5 ; Exc	hange the	contents o	of W4 and W	5
	Before	-	-	After		
	Unstructio		W4	ruction		
	W4 ABC	U	vv 4	4321		
	W5 432	1	W5	ABCD		

FBCL		Find Firs	t Bit Change f	rom Left		
Syntax:	{label:}	FBCL	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wnd		
Operands:	Ws ∈ [W0 . Wnd ∈ [W0					
Operation:	Max_Shift = Sign = (Ws Temp = (W Shift = 0 While ((Sh	= 15) & 0x8000 s) << 1 ift < Max_S Temp << 1 hift + 1	Shift) && ((Ten	np & 0x8000)	== Sign))	
Status Affected:	С	T				
Encoding:	1101	1111	0000	0ddd	dppp	SSSS
Description:	negative va Ws and wo bit number The next M the Least S allows for th values up. flag is set.	alue), startir rking towar result is sig ost Signific significant b he immedia If a bit char When a bit	ce of a one (fo ng from the Mo ds the Least S gn-extended to ant bit after the it is allocated b ate use of Wd nge is not found change is four address of the	ost Significant ignificant bit of 16-bits and p sign bit is allo bit number -14 with the SFTA d, a result of -1 nd, the C flag	bit after the f the word op laced in Wn ocated bit nu f. This bit ord c instruction 15 is returned is cleared.	sign bit of berand. Th d. mber 0 ar dering for scalin
	The 'p' bits	select the	source Addres	s mode.	-	
	Note:	This instru	ction operates	in Word mode	e only.	
Words:	1					
Cycles:	1					
Example 1	FBCL W1, W9			bit change e result to		in W1
	Before Instruction W1 55FF W9 FFFF SR 0000		After Instruction W1 55FF W9 0000 SR 0000	I		

Example 2	FBCL W1, W9	; Find 1st bit change from left in W1 ; and store result to W9
	Before	After
	Instruction	Instruction
	W1 FFFF	W1 FFFF
	W9 BBBB	W9 FFF1
	SR 0000	SR 0001 (C=1)
Example 3	FBCL [W1++], W9	; Find 1st bit change from left in [W1] ; and store result to W9 ; Post-increment W1
	Before	After
	Instruction	Instruction
	W1 2000	W1 2002
	W9 BBBB	W9 FFF9
Data	a 2000 FF0A	Data 2000 FF0A
	SR 0000	SR 0000

5

FF1L	Find First One from Left					
Syntax:	{label:}	FF1L	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wnd		
Operands:	Ws ∈ [W0 Wnd ∈ [W	-				
Operation:		/s) hift < Max_ Temp << 1 Shift + 1 Max_Shift nd)		mp & 0x8000))		
Status Affected:	С					
Encoding:	1100	1111	1000	0ddd	dppp	SSSS
Description:	Ws and wo The bit nu Bit numbe	orking towa mber result ring begins	rds the Least is zero-exter with the Mos	tarting from the Significant bit ded to 16-bits t Significant bit ant bit (allocate	of the word of and placed in (allocated nu	perand. Wnd. mber 1)
		icates a '1'	was not foun	d, and the C fla		
	The 'p' bits	select the	source Addre	e destination re ess mode. e source regist	0	
	Note:	This instru	uction operate	s in Word mod	e only.	
Words:	1					
Cycles:	1					
Example 1	FF1L W2, W5			one from t esult to W5	he left in	W2
	Before Instruction V2 000A V5 BBBB		Afte Instruc W2 000 W5 000	tion)A		

dsPIC30F Programmer's Reference Manual

Example 2 FF1L [W2++], W5 ; Find the 1st one from the left in [W2]; and store the result to W5 ; Post-increment W2 Before After Instruction Instruction W2 2000 W2 2002 W5 BBBB W5 0000

SR

0000

0001 (C=1)

Data 2000

Data 2000

SR

0000

0000

5

FF1R	Find First One from Right
Syntax:	{label:} FF1R Ws, Wnd [Ws], [Ws++], [Ws], [++Ws], [Ws],
Operands:	Ws ∈ [W0 W15] Wnd ∈ [W0 W15]
Operation:	$\begin{array}{l} Max_Shift = 17\\ Temp = (Ws)\\ Shift = 1\\ While ((Shift < Max_Shift) && !(Temp & 0x1))\\ Temp = Temp >> 1\\ Shift = Shift + 1\\ If (Shift == Max_Shift)\\ 0 \rightarrow (Wnd)\\ Else\\ Shift \rightarrow (Wnd) \end{array}$
Status Affected:	: C
Encoding:	1100 1111 0000 0ddd dppp ssss
Description:	Finds the first occurrence of a '1' starting from the Least Significant bit of Ws and working towards the Most Significant bit of the word operand. The bit number result is zero-extended to 16-bits and placed in Wnd. Bit numbering begins with the Least Significant bit (allocated number 1) and advances to the Most Significant bit (allocated number 16). A result o
	zero indicates a '1' was not found, and the C flag will be set. If a '1' is found, the C flag is cleared.
	The 'd' bits select the address of the destination register. The 'p' bits select the source Address mode. The 's' bits select the address of the source register.
	Note: This instruction operates in Word mode only.
Words:	1
Cycles:	1
Example 1	FF1R W1, W9 ; Find the 1st one from the right in W1 ; and store the result to W9
	Before After Instruction Instruction W1 000A W1 000A W9 BBBB W9 0002 SR 0000 SR 0000

Example 2 FF1R [W1++], W9 ; Find the 1st one from the right in [W1] ; and store the result to W9 ; Post-increment W1
Before After

I	nstructior	n l	nstruction
W1	2000	W1	2002
W9	BBBB	W9	0010
Data 2000	8000	Data 2000	8000
SR	0000	SR	0000

GOTO		Unconditio	nal Jump			
Syntax:	{label:}	GOTO	Expr			
Operands:			pression (but n nker to a lit23,		፪ [0 838860)6].
Operation:	lit23 \rightarrow PC NOP \rightarrow Inst	ruction Regis	ster			
Status Affected:	None					
Encoding:						
1st word	0000	0100	nnnn	nnnn	nnnn	nnn0
2nd word	0000	0000	0000	0000	0nnn	nnnn
Description:	memory ran	ge. The PC i Since the PC	ywhere within s loaded with must always	the 23-bit lite	al specified ir	n the
	The 'n' bits f	form the targe	et address.			
		The linker will used.	I resolve the s	pecified expre	ession into the	e lit23 to be
Words:	2					
Cycles:	2					
Example 1	026000 026004	GOTO MOV	_THERE W0, W1	; J	ump to _THE	IRE
	027844 _THE 027846	RE: MOV	#0x400, W2	-	ode executi esumes here	
	Before)		After		
	Instructi			Instruction	1	
	PC 02 60 SR 00		PC SR	02 7844		
	SK 00	000	эк	0000		
Example 2	000100 _code	e:		; st	art of cod	e
	026000 026004	GOTO 	_code+2	; Jı	ump to _cod	e+2
	Before			After		
				Instruction		
	PC 02 60 SR 00		PC SR	00 0102 0000		

GOTO		Unconditio	nal Indirect	Jump		
Syntax:	{label:}	GOTO	Wn			
Operands:	$Wn \in [W0]$.	W15]				
Operation:	$0 \rightarrow PC < 0 >$	\rightarrow PC<15:1				
Status Affected	None					
Encoding:	0000	0001	0100	0000	0000	SSSS
Description:	Zero is load into PC<15: boundary, V	led into PC<2 :1>. Since the Vn<0> is igno	22:16> and the PC must alword the procession of the procesion of the procession of the procession of the procession of t	ne value spec ways reside o	rds of progran cified in (Wn) on an even ac	is loaded
147 1		select the ad	dress of the s	source regist	er.	
Words:	1					
Cycles:	2					
Example 1	006000 006002		∛4 ∛0, W1		mp uncondit 16-bit val	-
	007844 _THERE 007846	 E: MOV #0)x400, W2		de executio sumes here	n
	Before Instruction		I	After Instruction		
	W4 7844	4	W4	7844		
	PC 00 6000		PC	00 7844		
	SR 0000	0	SR	0000		

© 2003 Microchip Technology Inc.

INC		Increment f				
Syntax:	{label:}	INC{.B}	f	{,WREG}		
Operands:	f ∈ [0 81	91]				
Operation:	(f) + 1 \rightarrow de	estination des	ignated by D)		
Status Affected:	DC, N, OV,	Z, C				
Encoding:	1110	1100	OBDf	ffff	ffff	ffff
Description:	destination destination	the contents register. Tl register. If W ot specified, t	he optional REG is spec	WREG op	erand deter ult is stored in	mines the
	The 'D' bit s	selects byte o selects the de select the ado	stination (0 f	or WREG, 1		,
		The extensio rather than a denote a wor The WREG is	word operat d operation,	ion. You may but it is not re	/ use a .w e equired.	
Words:	1					
Cycles:	1					
Example 1 INC	C.B 0x100	0;	Increment	0x1000 (By	yte mode)	
Data 1000 SR		Data 10	After Instruction 00 8F00 GR 0101) (DC, C=1)		
Example 2 INC	C 0x1000,	WREG ; ;	Increment (Word mode	0x1000 and e)	d store to	WREG
WREG Data 1000 SR		WRE Data 100 S		(DC, N=1)		

{label:} Ws ∈ [W0 Wd ∈ [W0	INC{.B}	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd]		
	14/4 51	[Ws++], [Ws], [++Ws],	[Wd++] [Wd]		
	14/4 51	[Ws], [++Ws],	[Wd]		
	14/4 51	[++Ws],			
			[++Wd]		
		[Ws],			
			[Wd]		
(Ws) + 1 -	→ Wd				
DC, N, O\	/, Z, C				
1110	1000	0Bqq	qddd	dppp	SSSS
the destination	ation register		-		
The 'q' bit The 'd' bit The 'p' bit	s select the de s select the ac s select the so	estination Add ddress of the ource Address	lress mode. destination re s mode.	gister.).
Note:	rather than a	a word opera	tion. You may	use a .w ex	
1					
1					
C.B W1,	[++W2]	; Incre	ment W1 an		W2
Before		After			
			n 1		
			-		
			1		
			(DC, N, OV	=1)	
C W1, W2		,		ore to W2	
1 FF7F 2 2000		W1 FF7F W2 FF80			
	Add one to the destina used for W The 'B' bit The 'q' bits The 'd' bits The 's' bits Note: 1 1 1 4 C.B W1, Before Instruction 1 FF7F 2 2000 0 ABCD 0 0000 C W1, W2 Before Instruction 1 FF7F	Add one to the contents the destination register used for Ws and Wd. The 'B' bit selects byte The 'q' bits select the di The 'd' bits select the at The 'p' bits select the at The 's' bits select the at Note: The extensi rather than denote a wo 1 1 1 %C.B W1, [++W2] Before Instruction 1 FF7F 2 2000 D ABCD Data 2 0000 C W1, W2 Before Instruction 1 FF7F 2 2000	Add one to the contents of the source the destination register Wd. Register used for Ws and Wd. The 'B' bit selects byte or word opera The 'a' bits select the destination Add The 'a' bits select the address of the 'a' bits select the address of the source Address The 'b' bits select the address of the 'a' bits select the address of the source Address The 'b' bits select the address of the 'a' bits select the address of the source Address The 's' bits select the address of the source Address of the source a word operation. B in the rather than a word operation, 1 1	Add one to the contents of the source register Ws the destination register Wd. Register direct or indir used for Ws and Wd. The 'B' bit selects byte or word operation (0 for word The 'q' bits select the destination Address mode. The 'd' bits select the address of the destination register Ws is select the address of the destination register Ws is select the address of the source register. Note: The extension .B in the instruction derather than a word operation, but it is not register that a word operation, but it is not register. 1 1 *C.B W1, [++W2] ; Pre-increment W1 and ; (Byte mode) Before After Instruction Instruction 1 1 *C.B W1, [++W2] ; Pre-increment W1 and ; (Byte mode) Before After Instruction Instruction 1 1 *C.B W1, [++W2] ; Increment W1 and ; (Byte mode) Before After Instruction Instruction 1 \$	Add one to the contents of the source register Ws and place the the destination register Wd. Register direct or indirect addressir used for Ws and Wd. The 'B' bit selects byte or word operation (0 for word, 1 for byte The 'q' bits select the address of the destination register. The 'B' bit selects byte or word operation (0 for word, 1 for byte The 'q' bits select the address of the destination register. The 'B' bit select the address of the destination register. The 's' bits select the address of the source register. Note: The extension .B in the instruction denotes a byte rather than a word operation. You may use a .W existent that a word operation, but it is not required. 1 1 1C.B W1, [++W2] ; Pre-increment W2 ; Increment W1 and store to ; (Byte mode) Before After Instruction Instruction 1 1 10 SR 11 1 12 ; Increment W1 and store to W2 ; (Word mode) Before After Instruction SR 10 ; W1 12 ; Increment W1 and store to W2 ; (Word mode) Before After Instruction Instruction 1 instruction 1 ; W1 12

© 2003 Microchip Technology Inc.

Instruction Descriptions

INC2		Increment f	by 2			
Syntax:	{label:}	INC2{.B}	f	{,WREG}		
Operands:	f ∈ [0 81	91]				
Operation:	(f) + 2 \rightarrow de	estination des	ignated by D			
Status Affected:	DC, N, OV,	Z, C				
Encoding:	1110	1100	1BDf	ffff	fff	ffff
Description:	Add two to the contents of the file register and place the result in the destination register. The optional WREG operand determines the destination register. If WREG is specified, the result is stored in WREG. WREG is not specified, the result is stored in the file register.					
	The 'D' bit :	selects byte o selects the de select the ado	estination (0 f	or WREG, 11		
		The extensio rather than a denote a wor	word operat	ion. You may	/usea.we	
Words:	1					
Cycles:	1					
Example 1 IN	IC2.B 0x10		rement 0x1 te mode)	000 by 2		
Data 1000 SF		Data 10	After Instruction 000 8F01 SR 0101) (DC, C=1)		
Example 2 INC	C2 0x1000,		increment (Word mode)	x1000 by 2	and store	to WREG
WREG Data 1000 SR	8FFF	WRE Data 100 S		(DC, N=1)		

INC2		Increment V	/s by 2			
Syntax:	{label:}	INC2{.B}	Ws,	Wd		
			[Ws],	[Wd]		
			[Ws++],	[Wd++]		
			[Ws],	[Wd]		
			[++Ws],	[++Wd]		
			[Ws],	[Wd]		
Operands:	$Ws \in [W0 . Wd \in [W0 .$					
Operation:	(Ws) + 2 \rightarrow	Wd				
Status Affected:	DC, N, OV,	Z, C				
Encoding:	1110	1000	1Bqq	qddd	dppp	SSSS
Description:		register Wd. I		register Ws ar ct or indirect a		
	The 'q' bits The 'd' bits The 'p' bits	select the des select the add select the sou	stination Add dress of the c urce Address	lestination reg	ister.	
	I	rather than a	word operat	instruction de ion. You may out it is not rec	use a .we	
Words:	1					
Cycles:	1					
Example 1 IN	IC2.B W1,	[++W2] ;;	Pre-incre Increment (Byte mod	by 2 and	store to W	1
	Before		After			
1.47				-		
W ² W2			W1 FF7F W2 2001	-		
Data 2000		Data 2	-	-		
SF	۲ 0000		SR 010C	(DC, N, OV	=1)	
Example 2 IN	IC2 W1, W		crement W1 ord mode)	by 2 and s	tore to W2	
	Before Instruction		After Instructio	n		

Instruction Descriptions

IOR		Inclusive O	R f and WRE	G		
	{label:}	IOR{.B}	f	{,WREG}		
Operands:	f ∈ [0 8′	191]				
Operation:	(f).IOR.(W	$REG) \rightarrow desti$	nation desigr	nated by D		
Status Affected:	N, Z					
Encoding:	1011	0111	OBDf	ffff	ffff	ffff
Description:	register W the destina destination WREG is i The 'B' bit	he logical inclu REG and the ation register. If W not specified, the selects byte of selects the de	contents of th The optional REG is spec the result is s or word opera	e file registe WREG oper- fied, the resu- tored in the f tion (0 for we	r and place th and determinult is stored in ile register. ord, 1 for byte	ne result i es the n WREG. e).
		select the add			ior nio region	
Words: Cycles:	1 1	rather than a denote a wor The WREG i	rd operation, s set to work	but it is not r ng register V	equired. V0.	
Example 1 IOP	R.B 0x100		(Byte mode		(Byte mod	e)
WREG Data 1000 SR	FF00	WRI Data 10		1		
Example 2 IO	R 0x1000,		IOR (0x100 (Word mode	•		
WREG Data 1000 SR	0FAB	WRI Data 10 (N=1) \$		1		

IOR	I	nclusive O	R Literal and	d Wn		
Syntax:	{label:} I	OR{.B}	#lit10,	Wn		
Operands:	lit10 ∈ [0 2 lit10 ∈ [0 5 Wn ∈ [W0	1023] for wo				
Operation:	lit10.IOR.(W	n) \rightarrow Wn				
Status Affected:	N, Z					
Encoding:	1011	0011	0Bkk	kkkk	kkkk	dddd
Description:	and the cont the working The 'B' bit se	ents of the v register Wn elects byte o	working regis	ter Wn and p tion (0 for wo	10-bit literal lace the resul ord, 1 for byte	t back int
			dress of the		ster.	
	a n		-		ng 10-bit Lit literal operar	
Words:	1					
Cycles:	1					
Example 1	IOR.B #0xAA,	W9 ; ;	IOR 0xAA (Byte mod			
	Before Instruction W9 1234 SR 0000		After Instruction W9 12BE SR 0008	n (N=1)		
Example 2	IOR #0x2AA,	W4 ; ;	IOR 0x2AA (Word mod			

IOR		Inclusive C	OR Wb and SI	nort Literal		
Syntax:	{label:}	IOR{.B}	Wb,	#lit5,	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 lit5 ∈ [0 Wd ∈ [W0	31]				
Operation:	(Wb).IOR.	lit5 \rightarrow Wd				
Status Affected:	N, Z					
Encoding:	0111	0www	wBqq	qddd	d11k	kkkk
Description:	register W destinatior	b and the 5-b n register Wd.	lusive OR ope it literal opera . Register dire indirect addre	nd and place ct addressine	the result in t g must be use	the d for Wb.
	The 'B' bit The 'q' bits The 'd' bits	selects byte of select the do select the ac provide the l The extension	ddress of the or word opera estination Add ddress of the literal operanc on .B in the	tion (0 for wo ress mode. destination re l, a five-bit in instruction d	ord, 1 for byte egister. teger number enotes a byte	e operatio
			a word operation,			xtension
Words:	1		•		•	
Cycles:	1					
Example 1 I	OR.B W1,	#0x5, [W9+	; Store	1 and 0x5 to [W9] increment	(Byte mode W9)
W Data 200 S	R 0000	Data 2	After Instructio W1 AAAA W9 2001 2000 00AF SR 0008 ; IOR W1 wi	(N=1)	ord mode)	
	,		; Store to		,	
V	Before Instruction /1 0000 /9 A34D R 0000	I	After Instructio W1 0000 W9 0000 SR 0002	n (Z=1)		

		Inclusive OR Wb and Ws							
Syntax:	{label:}	IOR{.B}	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]				
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wd ∈ [W0) W15]							
Operation:	(Wb).IOR	$(Ws) \rightarrow Wd$							
Status Affected:	N, Z								
Encoding:	0111	0www	wBqq	qddd	dppp	SSSS			
	The 'B' bi The 'q' bit The 'd' bit The 'p' bit	t selects byte s select the c s select the a s select the s	or word op lestination A address of th source Addr	Address mode ne destination ess mode.	vord, 1 for byte register.) .			
			laaress of tr	ne source regi	ster.				
	Note:	The extens rather than	ion.вint aword.op	he instruction	denotes a by nay use a .₩				
		The extens rather than	ion.вint aword.op	he instruction eration. You r	denotes a by nay use a .₩				
	Note:	The extens rather than	ion.вint aword.op	he instruction eration. You r	denotes a by nay use a .₩				
Words: Cycles: Example 1	Note: 1 1	The extens rather than	ion .B in t a word op ord operatic ^{W9++]} ; ;	he instruction eration. You r n, but it is not IOR W1 and Store resul	denotes a by nay use a .w required. [W5] (Byte	extension 1 mode)			

Example 2

IOR W1, W5, W9

Before Instruction					
W1	AAAA				
W5	5555				
W9	A34D				
SR	0000				

; IOR W1 and W5 (Word mode) ; Store the result to W9

In	After struction	
W1	AAAA	
W5	5555	
W9	FFFF	
SR	0008	(N=1)

LAC		Load Acc	umulator			
Syntax:	{label:}	LAC	Ws,	{#Slit4,}	Acc	
			[Ws],			
			[Ws++],			
			[Ws],			
			[Ws],			
			[++Ws],			
			[Ws+Wb],			
Operands:	Ws ∈ [W0 Wb ∈ [W0 Slit4 ∈ [-8 Acc ∈ [A,E) W15] +7]				
Operation:	Shift _{Slit4} (E	- (Ws))	\rightarrow Acc(A or B)			
Status Affected:	OA, OB, C	DAB, SA, SB,	, SAB			
Encoding:	1100	1010	Awww	wrrr	rggg	SSSS
	operand ir	ndicates an a	and indicates an rithmetic right sl be 1.15 fractiona	nift. The data	stored in the s	source
	operand ir register is sign-exten shifting. The 'A' bit The 'w' bit The 'r' bits The 'g' bits	ndicates an a assumed to aded (through specifies the ts specify the s encode the s select the s	rithmetic right sł	nift. The data al data and is o-backfilled (I cumulator. Wb. e-shift. mode.	stored in the sautomatically	source
	operand ir register is sign-exten shifting. The 'A' bit The 'w' bit The 'r' bits The 'g' bits	ndicates an a assumed to inded (through specifies the s specify the s encode the s select the s s specify the If the opera upper Accu	rithmetic right sh be 1.15 fractiona bit 39) and zero e destination acc offset register N accumulator pre- source Address source register ation moves mo- mulator register	hift. The data al data and is p-backfilled (l cumulator. Wb. e-shift. mode. Ws. ore than sign (AccxU), or	stored in the s automatically bits [15:0]), priv -extension da causes a satu	source or to ta into the
Words:	operand ir register is sign-exten shifting. The 'A' bit The 'A' bit The 'r' bits The 'g' bits The 's' bits Note:	ndicates an a assumed to inded (through specifies the s specify the s encode the s select the s s specify the If the opera upper Accu	rithmetic right sh be 1.15 fractiona bit 39) and zero e destination acc offset register N accumulator pre- source Address source register ation moves mo	hift. The data al data and is p-backfilled (l cumulator. Wb. e-shift. mode. Ws. ore than sign (AccxU), or	stored in the s automatically bits [15:0]), priv -extension da causes a satu	source or to ta into the
Words: Cycles:	operand ir register is sign-exten shifting. The 'A' bit The 'A' bit The 'Y bits The 'g' bits The 's' bits	ndicates an a assumed to inded (through specifies the s specify the s encode the s select the s s specify the If the opera upper Accu	rithmetic right sh be 1.15 fractiona bit 39) and zero e destination acc offset register N accumulator pre- source Address source register ation moves mo- mulator register	hift. The data al data and is p-backfilled (l cumulator. Wb. e-shift. mode. Ws. ore than sign (AccxU), or	stored in the s automatically bits [15:0]), priv -extension da causes a satu	source or to ta into the
Cycles:	operand ir register is sign-exten shifting. The 'A' bit The 'A' bit The 'r' bits The 'g' bit The 's' bits Note: 1	ndicates an a assumed to inded (through specifies the s specify the s encode the s select the s s specify the If the opera upper Accu	rithmetic right sh be 1.15 fractiona bit 39) and zero e destination acc offset register N accumulator pre- source Address source register ation moves mo- mulator register overflow and sa ; Load AC ; Content ; Post in	hift. The data al data and is p-backfilled (I cumulator. Wb. e-shift. mode. Ws. ore than sign (AccxU), or aturation bits CCB with [V is of [W4] iccrement W4 saturation	stored in the s automatically bits [15:0]), priv -extension da causes a satu will be set.	source or to ta into the uration, the
Cycles:	operand ir register is sign-exten shifting. The 'A' bit The 'A' bit The 'r' bits The 'g' bits The 'g' bits Note: 1 1 AC [W4++	hdicates an a assumed to l aded (through specifies the specify the s encode the s select the s s specify the If the opera upper Accu appropriate	rithmetic right sh be 1.15 fractiona bit 39) and zero e destination acc offset register N accumulator pre- source Address source register ation moves mo- mulator register overflow and sa ; Load AC ; Content ; Post ir ; Assume	hift. The data al data and is p-backfilled (I cumulator. Wb. e-shift. mode. Ws. ore than sign (AccxU), or aturation bits CCB with [V is of [W4] icrement W4 saturation : 0)	stored in the s automatically bits [15:0]), priv -extension da causes a satu will be set.	source or to ta into the uration, the
Cycles:	operand ir register is sign-exten shifting. The 'A' bit The 'A' bit The 'Y bits The 'g' bits The 's' bits Note: 1 1 AC [W4++	ndicates an a assumed to I aded (through specifies the specify the s encode the s select the s s specify the If the opera upper Accu appropriate	rithmetic right sh be 1.15 fractiona bit 39) and zero e destination acc offset register N accumulator pre- source Address source register ation moves mo- mulator register overflow and sa ; Load AC ; Content ; Post ir ; Assume	hift. The data al data and is p-backfilled (I cumulator. Wb. e-shift. mode. Ws. ore than sign (AccxU), or aturation bits CCB with [V is of [W4] iccrement W4 saturation	stored in the s automatically bits [15:0]), prid -extension da causes a satu will be set.	source or to ta into the uration, the
Cycles: Example 1 L	operand ir register is sign-exten shifting. The 'A' bit The 'w' bit The 'r' bits The 'g' bit: The 's' bits Note: 1 1 AC [W4++ Be Instr /4	hdicates an a assumed to l aded (through specifies the specify the sencode the s select the s s specify the lf the opera upper Accu appropriate	rithmetic right sh be 1.15 fractiona bit 39) and zero e destination acc offset register N accumulator pre- source Address source register ation moves mo- mulator register overflow and sa ; Load AC ; Content ; Post in ; Assume ; (SATB =	hift. The data al data and is b-backfilled (I cumulator. Wb. e-shift. mode. Ws. ore than sign (AccxU), or aturation bits CCB with [V is of [W4] icrement W4 saturation i o) After Instructi	stored in the s automatically bits [15:0]), prid -extension da causes a satu will be set.	source or to ta into the uration, the
Cycles: Example 1 L	operand ir register is sign-exten shifting. The 'A' bit The 'w' bit The 'r' bits The 'g' bit The 's' bits Note: 1 1 AC [W4++ Be Instr /4	hdicates an a assumed to l aded (through specifies the s specify the s select the s s select the s s specify the If the opera upper Accu appropriate	rithmetic right sh be 1.15 fractiona bit 39) and zero e destination acc offset register N accumulator pre- source Address source register ation moves mo- mulator register overflow and sa ; Load AC ; Content ; Post in ; Assume ; (SATE =	hift. The data al data and is b-backfilled (I cumulator. Wb. e-shift. mode. Ws. ore than sign (AccxU), or aturation bits CCB with [V is of [W4] icrement W4 saturation i o) After Instructi	stored in the s automatically pits [15:0]), priv -extension da causes a satu will be set.	source or to ta into the uration, the

Data 4002

SR

Example 2	LAC	[-	-W2],	#7, A	;	; Pre-decrement W2 ; Load ACCA with [W2] >> 7 ; Contents of [W2] do not cha: ; Assume saturation disabled ; (SATA = 0)				t change
		I	Befor nstruct	-			I	Aftei nstruct		
	W2			4002		W2			4000	
AC	CA	00	5125	ABCD		ACCA	FF	FF22	1000	
Data 4	000			9108	I	Data 4000			9108	

1221

0000

Data 4002

SR

1221

0000

LNK		Allocate St	ack Frame			
Syntax:	{label:}	LNK	#lit14			
Operands:	lit14 ∈ [0 .	16382]				
Operation:	$(W14) \rightarrow ($ (W15) + 2 (W15) $\rightarrow $ (W15) + lit	\rightarrow W15				
Status Affected:	None					
Encoding:	1111	1010	00kk	kkkk	kkkk	kkk0
	stack fram The 'k' bits	e of 16382 by specify the s	size of the sta	ck frame.		kimum
	Note:		ack pointer c	an only resid	la an a ward	
		lit14 must be	e even.	, · · · · ·	ie on a word	boundary,
Words:	1	lit14 must be	e even.		le on a word	boundary,
Words: Cycles:	1 1	lit14 must be	e even.		ie on a word	boundary,
	1		e even. e a stack f			boundary,
Cycles:	1	; Allocat				boundary,
Cycles:	1 K #0xA0	; Allocat		Frame of 16	50 bytes	boundary,
Cycles: Example 1 LN	1 K #0xA0 Befor Instruct	; Allocat	e a stack f W14 [Frame of 16 After Instructio	50 bytes	boundary,
Cycles: Example 1 LN	1 K #0xA0 Befor Instruct	; Allocat e ion	e a stack f	Frame of 16 After Instructio	50 bytes n	boundary,
Cycles: Example 1 LN W14	1 K #0xA0 Befor Instruct	; Allocat e ion 2000	e a stack f W14 [Frame of 16 After Instructio 2	50 bytes n 2002	boundary,

LSR		Logical S	hift Right f			
Syntax:	{label:}	LSR{.B}	f	{,WREG}		
Operands:	f ∈ [0 81	91]				
Operation:	$(f<0>) \rightarrow Des$	st<7> \rightarrow Dest<6:(\Rightarrow C <u>peration:</u> st<15> \Rightarrow Dest<1				
Status Affected:	N, Z, C					
Encoding:	1101	0101	OBDf	ffff	ffff	ffff
	Significant The option is specified result is sto	bit of the de al WREG of I, the result pred in the f	bit of the Status estination regis perand determinis stored in WF ile register.	ter. Ines the destir REG. If WREG	nation registe i is not specif	r. If WREG ied, the
	The 'f' bits Note 1:	select the a The extens rather than denote a w	destination (0 f ddress of the f sion .B in the a word opera ord operation, b is set to work	ile register. instruction d tion. You may but it is not re	enotes a byf y use a .w e quired.	e operation
Words:	1					
Cycles:	1					
Example 1	LSR.B 0x60		gically shif yte mode)	t right (02	x600) by oi	ne
Data 6	Before Instruction 00 55FF SR 0000	1	After Instructi ta 600 5577 SR 0000	on F		
Example 2	LSR 0x600		Logically s Store to WR (Word mode)	EG	(0x600) by	y one
Data 6 WRI		Da	After Instruct ta 600 555 WREG 2AF SR 000	ion F F		

LSR						
Syntax:	{label:}	LSR{.B}	Ws,	Wd		
			[Ws],	[Wd]		
			[Ws++],	[Wd++]		
			[Ws],	[Wd]		
			[++Ws],	[++Wd]		
			[Ws],	[Wd]		
Operands:	Ws ∈ [W0 Wd ∈ [W0	-				
Operation:	(Ws<0> For word $0 \rightarrow Wc$	d < 7 > $1 >) \rightarrow Wd < 6$ $y \rightarrow C$ y = 7 y = 7 > d < 15 >				
	0->	►C				
Status Affected:	N, Z, C	1	1	1	1	1
Encoding:	1101	0001	0Bqq	qddd	dppp	SSSS
Description:	the result shifted inte Most Sign	in the destin o the Carry b	ation register oit of the State Wd. Either re	ister Ws one b Wd. The Leas us register. Ze egister direct of	t Significant b ro is shifted ir	oit of Ws is nto the
	The 'q' bit The 'd' bit The 'p' bit	s select the s select the s select the	destination A address of th source Addre	ration (0 for w ddress mode. e destination r ess mode. e source regist	egister.	e).
	Note:	rather than	a word oper	e instruction de ation. You ma n, but it is not r	yuse a .we	•
Words:	1					
Cycles:	1					
Example 1	LSR.B WO, W		SR W0 (Byt tore resul			
	Before Instruction W0 FF03 W1 2378 SR 0000		After Instruction W0 FF03 W1 2301 SR 0001	-		

Instruction Descriptions Example 2

LSR W0, W1

; LSR W0 (Word mode) ; Store the result to W1

Before							
Instruction							
W0	8000						
W1	2378						
SR	0000						

After Instruction					
W0	8000				
W1	4000				
SR	0000				

LSR			Logical S	hift Right by	Short Literal		
Syntax:		{label:}	LSR	Wb,	#lit4,	Wnd	
Operands:		Wb ∈ [W0 lit4 ∈ [0 Wnd ∈ [W					
Operation:		$0 \rightarrow Wnd$	→ Shift_Val <15:15-Shift nift_Val> →	_Val+1> Wnd<15-Shift	_Val:0>		
Status Affected:		N, Z					
Encoding:		1101	1110	0www	wddd	d100	kkkk
Description:		unsigned	literal and s	contents of the tore the result st be used for	in the destination	ation register	
		The 'd' bit	s select the	address of th address of the e literal opera	e destination i		
		Note:	This instru	ction operates	s in Word mod	le only.	
Words:		1					
Cycles:		1					
Example 1	LSR	W4, ‡	14, W5	; LSR W4 k ; Store re	by 14 esult to W5		
		Before		After			
		nstruction		Instructio	-		
	W4	C800		W4 C800			
	W5 SR	1200 0000		W5 0003 SR 0000			
Example 2	LSR	W4, ‡	1, W5	; LSR W4 k ; Store re	by 1 esult to W5		
		Before		After			
		struction		Instructior	-		
	W4	0505		W4 0505	_		
	W5 SR	F000		W5 0282 SR 0000	4		
	sк	0000		SR 0000			

LSR			Logical	Shift Right b	y Wns		
Syntax:		{label:}	LSR	Wb,	Wns,	Wnd	
Operands:		$Wb \in [WC]$ $Wns \in [W]$ $Wnd \in [W]$	0W15]				
Operation:		Wns<4:0> $0 \rightarrow$ Wnd Wb<15:St	<15:15-Sh		ift_Val:0>		
Status Affected:		N, Z					
Encoding:		1101	1110) Owww	wddd	d000	SSSS
Description:		Significan	t bits of W	ns (only up to	15 positions)	egister Wb by th) and store the ust be used for	result in th
		The 'd' bit	s select th	e address of t e address of t e address of t	he destination	on register.	
				ruction operate greater than		node only. be loaded with	0x0.
Words:		1					
Cycles:		1					
Example 1	LSR	WO, W	1, W2	; LSR W0 ; Store 1	by W1 result to	W2	
	h	Before nstruction		After Instruct			
	wo	COOC		W0 C00	-		
	W1	0001		W1 000	1		
	W2	2390		W2 600	6		
	SR	0000		SR 000	0		
Example 2	LSR	W5, W	4, W3	; LSR W5 ; Store 1	by W4 result to	W3	
		Before		After			
	-	struction					
	W3 W4	DD43 000C		W3 000 W4 000			
	W5	0800		W5 080			
	SR	0000		SR 000			

Section	5.	Instruction	Descri	ptions
---------	----	-------------	--------	--------

MAC			Multiply and				
Syntax:	{label:}	MAC	Wm*Wn, Acc	{,[Wx], Wxd}	{,[Wy],	Wyd}	{,AWB}
				{,[Wx]+=kx, V	/xd} {,[Wy]-	⊦=ky, Wyd}	
				{,[Wx]-=kx, W			
				{,[W9+W12], '	Wxd} {,[W11	+W12], Wy	d}
Operands:		$Acc \in [$				·	*W7]
		Wy ∈ [\	W8, W9]; kx ∈ [W10, W11]; ky ∈ [W13, [W13]+=	∈ [- 6, -4, -2, 2,]
Operation:		([Wx])– ([Wy])–	or B)) + (Wm)*(→ Wxd; (Wx)+ky → Wyd; (Wy)+ky or A)) rounded	x→Wx y→Wy	or B)		
Status Affecte	d:	OA, OE	B, OAB, SA, SB	, SAB			
Encoding:		1100	0 mmm	A0xx	yyii	iijj	jjaa
Description:		Multiply the contents of two working registers, optionally pre-fetch operands in preparation for another MAC type instruction and optionally store the unspecified accumulator results. The 32-bit result of the signe multiply is sign-extended to 40-bits and added to the specified accumulator.					
		which s Section optiona	ds Wx, Wxd, W upport indirect n 4.14.1 "MAC I store of the "o n 4.14.4 "MAC	and register o Pre-Fetches' other" accumul	ffset address '. Operand A	sing, as des WB specifie	cribed in
		 The 'm' bits select the operand registers Wm and Wn for the multiply. The 'A' bit selects the accumulator for the result. The 'x' bits select the pre-fetch Wxd destination. The 'y' bits select the pre-fetch Wyd destination. The 'i' bits select the Wx pre-fetch operation. The 'j' bits select the Wy pre-fetch operation. The 'a' bits select the accumulator write back destination. 					
		Note		CORCON<0> or an integer.	, determines	if the multip	oly is
		4					
Words:		1					

Example 1

MAC W4*W5, A, [W8]+=6, W4, [W10]+=2, W5 ; Multiply W4*W5 and add to ACCA

- ; Fetch [W8] to W4, Post-increment W8 by 6
- ; Fetch [W10] to W5, Post-increment W10 by 2
- ; CORCON = 0x00C0 (fractional multiply, normal saturation)

	Before Instruction	After Instruction		
W4	A022	W4	2567	
W5	B900	W5	909C	
W8	0A00	W8	0A06	
W10	1800	W10	1802	
ACCA	00 1200 0000	ACCA	00 472D 2400	
Data 0A00	2567	Data 0A00	2567	
Data 1800	909C	Data 1800	909C	
CORCON	00C0	CORCON	00C0	
SR	0000	SR	0000	

Example 2

MAC W4*W5, A, [W8]-=2, W4, [W10]+=2, W5, W13

; Multiply W4*W5 and add to ACCA

; Fetch [W8] to W4, Post-decrement W8 by 2

; Fetch [W10] to W5, Post-increment W10 by 2

- ; Write Back ACCB to W13
- ; CORCON = 0x00D0 (fractional multiply, super saturation)

	Befe Instru		After Instruction				
W4		1000	W4			5BBE	
W5		3000	W5			C967	
W8		0A00	W8			09FE	
W10		1800	W10			1802	
W13		2000	W13			0001	
ACCA	23 500	0 2000	ACCA	23	5600	2000	
ACCB	00 000	0 8F4C	ACCB	00	0000	1F4C	
Data 0A00		5BBE	Data 0A00			5BBE	
Data 1800		C967	Data 1800			C967	
CORCON		00D0	CORCON			00D0	
SR		0000	SR			8800	(OA, OAB=1)

MAC			Square and	Accumulate					
Syntax:	{label:}	MAC V	Wm*Wm, Acc	{,[Wx]+=kx, \ {,[Wx]-=kx, V	Wxd} Vxd}	{,[Wy] {,[Wy]			
Operands:		$\begin{array}{l} Acc \in [A, B] \\ Wx \in [W8] \end{array}$	∈ [W4*W4, W 3] , W9]; kx ∈ [-(0, W11]; ky ∈	6, -4, -2, 2, 4,	6]; Wx	- kd ∈ [N			
Operation:		$([Wx]) \rightarrow V$	$(Acc(A \text{ or } B)) + (Wm)^*(Wm) \rightarrow Acc(A \text{ or } B)$ $([Wx]) \rightarrow Wxd; (Wx)+kx \rightarrow Wx$ $([Wy]) \rightarrow Wyd; (Wy)+ky \rightarrow Wy$						
Status Affecte	ed:	OA, OB, C	DAB, SA, SB,	SAB					
Encoding:		1111	0 0 mm	A0xx	УУ	rii	iijj	jj00	
Description:		preparatio unspecifie	n for another d accumulato	MAC type inst r results. The	ruction 32-bit	and o result	ly pre-fetch op ptionally store of the signed r d accumulato	the nultiply is	
		Operands Wx, Wxd, Wy and Wyd specify optional pre-fetch operations, which support indirect and register offset addressing, as described in Section 4.14.1 "MAC Pre-Fetches" .							
		The 'm' bits select the operand register Wm for the square. The 'A' bit selects the accumulator for the result. The 'x' bits select the pre-fetch Wxd destination. The 'y' bits select the pre-fetch Wyd destination. The 'i' bits select the Wx pre-fetch operation. The 'j' bits select the Wy pre-fetch operation.							
		Note:	The IF bit, C or an intege		detern	nines if	the multiply is	fractiona	
Words:		1							
Cycles:		1							

Example 1

MAC W4*W4, B, [W9+W12], W4, [W10]-=2, W5

- ; Square W4 and add to ACCB $% \left({\left({{{\rm{ACCB}}} \right)} \right)$
- ; Fetch [W9+W12] to W4
- ; Fetch [W10] to W5, Post-decrement W10 by 2
- ; CORCON = 0x00C0 (fractional multiply, normal saturation)

	Befor Instruct	-		I	After nstruct	
W4		A022	W4			A230
W5		B200	W5			650B
W9		0C00	W9			0C00
W10		1900	W10			18FE
W12		0020	W12			0020
ACCB	00 2000	0000	ACCB	00	67CD	0908
Data 0C20		A230	Data 0C20			A230
Data 1900		650B	Data 1900			650B
CORCON		00C0	CORCON			00C0
SR		0000	SR			0000

Example 2

MAC W7*W7, A, [W11]-=2, W7 ; Square W7 and add to ACCA

; Fetch [W11] to W7, Post-decrement W11 by 2

; CORCON = 0x00D0 (fractional multiply, super saturation)

	I	Befor nstruct	-		I	After nstruct		
W7			76AE	W7			23FF	
W11			2000	W11			1FFE	
ACCA	FE	9834	4500	ACCA	FF	063E	0188	
Data 2000			23FF	Data 2000			23FF	
CORCON			00D0	CORCON			00D0	
SR			0000	SR			8800	(OA, OAB=1)
				-				•

MOV		Move f to D	estination				
Syntax:	{label:}	MOV{.B}	f	{,WREG}			
Operands:	f ∈ [0 81	91]					
Operation:	=	nation designa	ated by D				
Status Affected:	N, Z	-	-				
Encoding:	1011	1111	1BDf	ffff	ffff	ffff	
Description:	The option WREG is specified, t	contents of the nal WREG o specified, the he result is stand ne status register	perand dete e result is ored back to	ermines the stored in W	destination REG. If WR	register. EG is n	
	The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 'D' bit selects the destination (0 for WREG, 1 for file register). The 'f' bits select the address of the file register.						
		The extension rather than a denote a wor The WREG is	word operation,	tion. You may but it is not re	y use a .w ex equired.		
Words:	1						
Cycles:	1						
Example 1 MOV	.B TMR0,	WREG ; mc	ove (TMR0)	to WREG (B	Byte mode)		
WREG (W0) TMR0 SR	Before nstruction 9080 2355 0000	WREG (W0 TMR(SF	0 2355				
Example 2 MOV	0x800	; up	date SR ba	used on (02	(800) (Word	d mode)	
Data 0800 SR	Before nstruction B29F 0000	Data 0800 SF		J =1)			

MOV		Move WRE	G to f			
Syntax:	{label:}	MOV{.B}	WREG,	f		
Operands:	f ∈ [0 81	91]				
Operation:	(WREG) –	→ f				
Status Affected:	None					
Encoding:	1011	0111	1B1f	ffff	ffff	ffff
Description:	Move the or specified fi		e default wo	rking register	WREG into t	he
		selects byte o select the ad		ation (0 for wo	ord, 1 for byte	e).
		than a word word move,	move. You but it is not r	nstruction den may use a .v equired. king register V	extension to	
Words:	1					
Cycles:	1					
Example 1 MOV	.B WREG,	0x801	; move WR	EG to 0x801	L (Byte mod	de)
WREG (W0) Data 0800 SR	Before Instruction 98F3 4509 0000	WREG (\ Data 0/		3		
Example 2 MOV	WREG,	DISICNT	; move W	REG to DIS	ICNT	
WREG (W0) DISICNT SR	Before Instruction 00A0 0000 0000	WREG (DISIC	-	ion 0 0		

MOV		Move f to W	/nd			
Syntax:	{label:}	MOV	f,	Wnd		
Operands:	f ∈ [0 65 Wnd ∈ [W					
Operation:	$(f) \rightarrow Wnd$					
Status Affected:	None					
Encoding:	1000	Offf	ffff	ffff	ffff	dddd
Description:	register ma	vord contents ay reside anyv igned. Registe	where in the	32K words of	data memory	, but must
		select the ada select the ac		-	egister.	
		This instructi Since the file upper 15 bits assumed to l	e register add s of the file re		word aligne	
Words:	1					
Cycles:	1					
Example 1 MOV	CORCON	, W12 ;	move COR	CON to W12		
W12 CORCON SR	Before nstruction 78FA 00F0 0000	W CORCC S		n]		
Example 2 MOV	0x27F	E, W3 ;	move (0x2	7FE) to W3		
ا W3 Data 27FE SR	Before Instruction 0035 ABCD 0000	Data 27	After Instruction V3 ABCD FE ABCD GR 0000	n 		

MOV		Move Wns t	o f			
Syntax:	{label:}	MOV	Wns,	f		
Operands:	f ∈ [0 65 Wns ∈ [W0					
Operation:	$(Wns) \rightarrow f$					
Status Affected:	None					
Encoding:	1000	lfff	ffff	ffff	ffff	SSSS
Description:	register. Th	vord contents ne file register ut must be wo n.	may reside	anywhere in t	he 32K words	s of data
		select the add select the add			er.	
		This instruction Since the file upper 15 bits assumed to b	register add of the file r	dress must be		
Words:	1					
Cycles:	1					
Example 1 MO	V W4, XM	IDOSRT	; move W4	to XMODSRT		
W4 XMODSRT SR	1340	XMODS	After Instructio N4 1200 RT 1200 SR 0000	n 		
Example 2 MO	V W8, ()x1222	; move W8	to data ad	dress 0x122	22
W8 Data 1222 SF	2 FD88	Data 12	After Instructio W8 F200 222 F200 SR 0000	n]]		

MOV.B		Move 8-bi	t Literal to W	nd		
Syntax:	{label:}	MOV.B	#lit8,	Wnd		
Operands:	lit8 \in [0 . Wnd \in [V	255] V0 W15]				
Operation:	lit8 \rightarrow Wr	nd				
Status Affected:	None					
Encoding:	1011	0011	1100	kkkk	kkkk	dddd
Description:		te of Wnd is n	ral 'k' is loaded ot changed. R			
			value of the li address of the		ter.	
	Note:	This instruc must be pro	ction operates ovided.	in Byte mod	e and the .≞	extensior
Words:	1					
Cycles:	1					
Example 1	MOV.B #0	x17, W5	; load W5	with #0x17	(Byte mode	e)
	Before		After			
	Instructio	n T	Instructio			
	W5 7899 SR 0000	_	W5 7817 SR 0000	_		
Example 2	MOV.B #0	xFE, W9	; load W9	with #0xFE	(Byte mod	e)
	Before Instructio W9 AB23 SR 0000	n]]	After Instruction W9 ABFE SR 0000]		

MOV		Move 16-bit	Literal to W	/nd				
Syntax:	{label:}	MOV	#lit16,	Wnd				
Operands:	lit16 ∈ [-32 Wnd ∈ [W	2768 65535 0 W15]]					
Operation:	lit16 \rightarrow Wr	nd						
Status Affected:	None							
Encoding:	0010	kkkk	kkkk	kkkk	kkkk	dddd		
Description:	The 16-bit be used for	literal 'k' is loa r Wnd.	aded into Wr	d. Register d	lirect address	ing must		
		The 'k' bits specify the value of the literal. The 'd' bits select the address of the working register.						
		This instructi The literal ma or unsigned	ay be specifi	ed as a signe		68:32767],		
Words:	1							
Cycles:	1							
Example 1	IOV #0x423	31, W13	; load W1	3 with #0x	4231			
	Before Instruction 13 091B GR 0000		After Instructio	n]				
Example 2	10V #0x4,	W2	; load W2	with #0x4				
S	Before Instruction V2 B004 SR 0000	\$	After Instructio V2 0004 SR 0000	n] with #-10	0.0			
v	Before Instruction V8 23FF SR 0000	٧	After Instructio V8 FC18 SR 0000					

MOV		Move [Ws	with offset] to	Wnd		
Syntax:	{label:}	MOV{.B}	[Ws+Slit10],	Wnd		
Operands:		512 511] fo 1024 1022	or byte operatio] (even only) fo		tion	
Operation:	[Ws+Slit10	$] \rightarrow Wnd$				
Status Affected:	None					
Encoding:	1001	0kkk	kBkk	kddd	dkkk	SSSS
Description:	range of S maintain w used for th	lit10 is increa vord address le source, an	lit10] are loade ased to [-1024 alignment. Reg d direct addres value of the lite	1022] and gister indirect ssing must be	Slit10 must addressing	be even t must be
	The 'B' bit The 'd' bits	selects byte s select the a	or word operat ddress of the c ddress of the s	ion (0 for woi lestination reg	gister.	e).
	2:	than a word word move, In Byte mod Section 4.6	on . B in the ins d move. You m but it is not red le, the range of G "Using 10-bit an address offs	ay use a .w quired. Slit10 is not Literal Oper	extension t	o denote specified
Words:	1					
Cycles:	1					
Example 1 MOT	∕.Β [₩8⊣	⊦0x13], W1	0 ; load W1 ; (Byte m	0 with [W8 Node)	+0x13]	
W8 W10		V	After Instruction W8 1008 W10 4033			
Data 101A SR		Data 1	01A 3312 SR 0000			
Example 2 MOT	/ [W4+0	Dx3E8], W2	; load W2 ; (Word mc	with [W4+0 ode)	x3E8]	
W2 W4 Data 0BE8 SR	0800 5634	Data 0I	After Instruction W2 5634 W4 0800 BE8 5634 SR 0000			

MOV		Move Wns	to [Wd with	offset]		
Syntax:	{label:}	MOV{.B}	Wns,	[Wd+Slit10]		
Operands:		512 511] in 024 1022]		in Word mode		
Operation:	$(Wns) \rightarrow [V]$	Vd+Slit10]				
Status Affected:	None					
Encoding:	1001	1kkk	kBkk	kddd	dkkk	SSSS
Description:	of Slit10 is maintain w	increased to ord address	[-1024 10 alignment. R	Vd+Slit10]. In ' 22] and Slit10 egister direct a ng must be use	must be eve addressing n	en to nust be
	The 'B' bit The 'd' bits	selects byte select the a	ddress of the	teral. ation (0 for wo destination re destination re	gister.	e).
	2:	than a word word move, In Byte mod Section 4.6	I move. You but it is not re e, the range "Using 10-b	nstruction den may use a . M equired. of Slit10 is not it Literal Ope fset from Wd.	reduced as	to denote specified
Words:	1					
Cycles:	1					
Example 1 MO	V.B W0, [[W1+0x7]	; store V ; (Byte n	10 to [W1+03 node)	c7]	
W0 W ⁷ Data 1806 SF	1 1800 6 2345		After Instruction W0 9015 W1 1800 806 1545 SR 0000			
Example 2 MO	V W11, [W	11-0x400]	; store V ; (Word n	111 to [W1-0 node))x400]	
W ⁷ W1 ² Data 0C00 SF	1 8813 D FFEA		After Instruction W1 1000 V11 8813 C00 8813 SR 0000			

MOV		Move Ws	to Wd			
Syntax:	{label:}	MOV{.B}	Ws,	Wd		
			[Ws],	[Wd]		
			[Ws++],	[Wd++]		
			[Ws],	[Wd]		
			[Ws],	[Wd]		
			[++Ws],	[++Wd]		
			[Ws+Wb],	[Wd+Wb]		
Operands:	Ws ∈ [W0 Wb ∈ [W0 Wd ∈ [W0) W15]				
Operation:	$(Ws) \rightarrow V$	Vd				
Status Affected:	None					
Encoding:	0111	lwww	wBhh	hddd	dggg	SSSS
Description:			-	ister into the or		-
	The 'g' bit	ts select the s	ource Addres	e destination re ss mode. e source regist	-	
	Note 1			nstruction den may use a . v	-	
		word move	, but it is not r	equired.		
	2:	-		ddressing mo the offset mus		
		the 'w' enco	oding bits are	shared by Ws	and Wd.	
				s" translates to " translates to		
Words:	1				HOV [WI	5], wa.
Cycles:	1					
			Move [W0] Post-decre	to W4 (Byte ement W0	e mode)	
	Before		After			
	Instruction		Instructio	on		
W			00A0	4		
W		– , –	W4 2989	4		
Data 0A0	0 8988	Data 0.	A00 8988	4		

SR

0000

Section 5. Instruction Descriptions

SR

0000

Instruction Descriptions

; Move [W6] to [W2+W3] (Word mode) ; Post-increment W6 Example 2 MOV [W6++], [W2+W3] Before After Instruction Instruction W2 0800 W2 0800 W3 0040 W3 0040 W6 1228 W6 122A Data 0840 9870 Data 0840 0690 Data 1228 Data 1228 0690 0690 SR 0000 SR 0000

MOV.D		Double-W	/ord Move fro	m Source to	Wnd	
Syntax:	{label:}	MOV.D	Wns, [Ws], [Ws++], [Ws],	Wnd		
			[++Ws], [Ws],			
Operands:	Ws ∈ [W0	/0, W2, W4 .) W15] /0, W2, W4 .	-			
Operation:	Wns → Wns+1 For indire	addressing Wnd \rightarrow Wnd+1 ct addressing escription				
Status Affected:	None				_	
Encoding:	1011	1110	0000	0ddd	0ppp	SSSS
	specifies t double-wo by 4 bytes The 'd' bit The 'p' bit	the effective ord. Any pre/ s to accomm s select the s select the	If indirect ad address for th post-incremen odate for the c address of the source Address address of the	e Least Signifi t or pre/post-d louble-word. destination re s mode.	cant Word of ecrement wil gister.	the
	2:	for informa Wnd must	ction only oper tion on how do be an even wo ction "POP.D	ouble-words an orking register.	e aligned in	memory.
Words:	1					
Cycles:	2					
Example 1	MOV.D W2	,W6 ;	Move W2 to	W6 (Double	mode)	
	Before Instruction W2 12FB W3 9877 W6 9833 W7 FCC6 SR 0000		After Instructi W2 12F1 W3 987' W6 12F1 W7 987' SR 0000	3 7 3 7		

Before After Instruction Instruction W4 B012 W4 A319 W5 W5 FD89 9927 W7 W7 0900 08FC Data 0900 Data 0900 A319 A319 Data 0902 9927 Data 0902 9927 SR 0000 SR 0000

Example 2 MOV.D [W7--], W4 ; Move [W7] to W4 (Double mode) ; Post-decrement W7

MOV.D		Double-W	ord Move fro	om Wns to De	estination	
Syntax:	{label:}	MOV.D	Wns,	Wnd		
				[Wd]		
				[Wd++]		
				[Wd]		
				[++Wd]		
				[Wd]		
Operands:	-), W2, W4), W2, W4 W15]	-			
Operation:	Wns \rightarrow Wns+1 -	Wnd → Wnd+1	of destination g of destinatio			
	See Des	-	y or destination	41.		
Status Affected:	None					
Encoding:	1011	1110	10qq	qddd	d000	sss0
	will adjust \ The 'q' bits	Wd by 4 byt select the c	es to accomr	ost-increment nodate for the ddress mode.	double-word	
			address of the address of the	e destination r e source regis	-	
	The 's' bits Note 1: 2:	select the a This instru Figure 4-2 f memory. Wnd must I	address of the uction only for informatio be an even w		ter pair. n double-wo ble-words are r.	e aligned
	The 's' bits Note 1: 2:	select the a This instru Figure 4-2 f memory. Wnd must I	address of the uction only for informatio be an even w	e source regis operates ou n on how doul orking registe	ter pair. n double-wo ble-words are r.	e aligned
Words:	The 's' bits Note 1: 2: 3: 1	select the a This instru Figure 4-2 f memory. Wnd must I The instruct	address of the uction only for informatio be an even w	e source regis operates ou n on how doul orking registe	ter pair. n double-wo ble-words are r.	e aligned
	The 's' bits Note 1: 2: 3:	select the a This instru Figure 4-2 f memory. Wnd must I The instruct	address of the uction only for informatio be an even w	e source regis operates ou n on how doul orking registe	ter pair. n double-wo ble-words are r.	e aligned
Words: Cycles: Example 1	The 's' bits Note 1: 2: 3: 1	select the a This instru Figure 4-21 memory. Wnd must I The instruct [W15++].	address of the uction only for informatio be an even w ction PUSH.1	e source regis operates ou n on how doul orking registe	ter pair. n double-wo ble-words are r. ates to MOV	e aligned
Cycles:	The 's' bits Note 1: 2: 3: 1 2 MOV.D W10, W Before	select the a This instru Figure 4-21 memory. Wnd must I The instruct [W15++].	address of the uction only for informatio be an even w ction PUSH.1 ove W10 to After	e source regis operates of n on how doul orking registe O Ws transk	ter pair. n double-wo ble-words are r. ates to MOV	e aligned
Cycles:	The 's' bits Note 1: 2: 3: 1 2 MOV.D W10, W Before Instruction	select the a This instru Figure 4-21 memory. Wnd must I The instruc [W15++].	address of the uction only for informatio be an even w ction PUSH.1 ove W10 to After Instructio	e source regis operates of n on how doul orking registe O Ws transk	ter pair. n double-wo ble-words are r. ates to MOV	e aligned
Cycles:	The 's' bits Note 1: 2: 3: 1 2 MOV.D W10, W Before	select the a This instru Figure 4-21 memory. Wnd must I The instruc [W15++].	address of the uction only for informatio be an even w ction PUSH.1 ove W10 to After	e source regis operates of n on how doul orking registe O Ws transk	ter pair. n double-wo ble-words are r. ates to MOV	e aligned
Cycles: Example 1	The 's' bits Note 1: 2: 3: 1 2 MOV.D W10, W Before Instruction W0 9000	select the a This instru Figure 4-21 memory. Wnd must I The instruct [W15++].	address of the uction only for informatio be an even w ction PUSH.1 ove W10 to After Instructio W0 CCFB	e source regis operates of n on how doul orking registe O Ws transk	ter pair. n double-wo ble-words are r. ates to MOV	e aligned
Cycles: Example 1	The 's' bits Note 1: 2: 3: 1 2 MOV.D W10, W Before Instruction W0 9000 W1 4322	select the a This instru Figure 4-21 memory. Wnd must I The instruc [W15++].	address of the uction only for informatio be an even w ction PUSH.1 ove W10 to After Instructic W0 CCFB W1 0091	e source regis operates of n on how doul orking registe O Ws transk	ter pair. n double-wo ble-words are r. ates to MOV	e aligned

; Pre-decrement W6 (Double mode) Example 2 MOV.D W4, [--W6] ; Move W4 to [W6] Before After Instruction Instruction W4 100A W4 100A W5 W5 CF12 CF12 W6 W6 0804 0800 Data 0800 A319 Data 0800 100A Data 0802 Data 0802 9927 CF12 SR 0000 SR 0000

Syntax:	{label:}	MOVSAC	Acc	{,[Wx], Wxd}		{,[Wy],	Wyd}	{,AWB}	
				{,[Wx]+=kx, V	Vxd}	{,[Wy]+	⊦=ky, Wyd}		
				{,[Wx]-=kx, W	/xd}	{,[Wy]-	=ky, Wyd}		
				{,[W9+W12],	Wxd}	{,[W11	+W12], Wyd	}	
Operands:			W11]; k	∈ [-6, -4, -2, 2, 4 y ∈ [-6, -4, -2, 2 +=2]					
Operation:		$([Wx]) \rightarrow Wx$ $([Wy]) \rightarrow Wy$ (Acc(B or A))	d; (Wy)+	-ky→Wy					
Status Affected	l:	None							
Encoding:		1100	0111	A0xx	УУ	ii	iijj	jjaa	
Description:		Optionally pre-fetch operands in preparation for another MAC type instruction and optionally store the unspecified accumulator results. Even though an accumulator operation is not performed in this instruction, an accumulator must be specified to designate which accumulator to write back.							
		which support Section 4.1 optional store	ort indire 4.1 "MA e of the	Wy and Wyd s ct and register C Pre-Fetches "other" accumu C Write Back"	offset a ". Ope lator, a	address erand A	sing, as desc WB specifies	ribed in	
		The 'x' bits s The 'y' bits s The 'i' bits s The 'j' bits s	select the select the elect the elect the	e other accumu e pre-fetch Wxc e pre-fetch Wyc e Wx pre-fetch c e Wy pre-fetch c e accumulator y	d destir d destir operatio operatio	nation. nation. on. on.			
Words:		1							
Cycles:		1							
Example 1	; Fe ; Fe	tch [W9] t	o W6 to W7,	[W11]+=4, W Post-increm			4		
		Before				After			
		Instruction		14/0	In	structio			
	W6 W7		022	W6 W7			811		
	W9		200 800	W9			800		
	W11		900	W11			.904		
	W13		020	W13			290		
A	CCA	00 3290 5	968	ACCA	00 3	290 5	968		
Data	0800	7	811	Data 0800		7	811		
Data	-	B	2AF	Data 1900		E	2AF		
	SR		000	SR			000		

```
Example 2
```

2	MOVSAC	A, [W9]-=2,	W4, [W11+W12], W6	, [W13]+=2
	; Fetch	[W9] to W4,	Post-decrement WS	by 2
	; Fetch	[W11+W12] to	5 W6	
	; Store	ACCB to [W13	3], Post-increment	W13 by 2

	I	Befor nstruct	-		After Instruction		
W4			76AE	W4			BB00
W6			2000	W6			52CE
W9			1200	W9			11FE
W11			2000	W11			2000
W12			0024	W12			0024
W13			2300	W13			2302
ACCB	00	9834	4500	ACCB	00	9834	4500
Data 1200			BB00	Data 1200			BB00
Data 2024			52CE	Data 2024			52CE
Data 2300			23FF	Data 2300			9834
SR			0000	SR			0000

Syntax: {label:}	MPY	Wm*Wn, Acc	{.[Wx], Wxd}		{,[Wy],	Wvd}		
- ,		,	{,[Wx]+=kx, \					
			{,[Wx]-=kx, V	-				
				-		+W12], Wyd}		
				,	01	1, 1, 1, 1		
Operands:	Wm*Wn Acc ∈ [A,		4*W6, W4*W	7, W5	*W6, W	5*W7, W6*W7	7]	
	Wy ∈ [W	8, W9]; kx ∈ [- 10, W11]; ky ∈ W13, [W13]+=	[-6, -4, -2, 2					
Operation:	([Wx])→	Wm)*(Wn) → Acc(A or B) [Wx])→ Wxd; (Wx)+kx→Wx [Wy])→ Wyd; (Wy)+ky→Wy						
Status Affected:	OA, OB,	OAB, SA, SB,	, SAB					
Encoding:	1100	Ommm	A0xx	УУ	rii	iijj	jj11	
Description:	operands store the	in preparatio unspecified a s sign-extende	n for another	MAC ty esults.	vpe inst The 32	ionally pre-feto ruction and op -bit result of th the specified	tionally	
	Operands Wx, Wxd, Wy and Wyd specify optional pre-fetch operations which support indirect and register offset addressing, as described in Section 4.14.1 "MAC Pre-Fetches" .							
	The 'm' bits select the operand registers Wm and Wn for the multiply: The 'A' bit selects the accumulator for the result. The 'x' bits select the pre-fetch Wxd destination. The 'y' bits select the pre-fetch Wyd destination. The 'i' bits select the Wx pre-fetch operation. The 'j' bits select the Wy pre-fetch operation.							
	Note:	The IF bit, 0 fractional or		, detei	rmines	if the multiply i	S	
Words:	1							
Cycles:	1							

Example 1

MPY W4*W5, A, [W8]+=2, W6, [W10]-=2, W7

- ; Multiply W4*W5 and store to ACCA
- ; Fetch [W8] to W6, Post-increment W8 by 2
- ; Fetch [W10] to W7, Post-decrement W10 by 2 $\,$
- ; CORCON = 0x0000 (fractional multiply, no saturation)

	Before Instruction	After Instruction			
W4	C000	W4		C000	
W5	9000	W5		9000	
W6	0800	W6		671F	
W7	B200	W7		E3DC	
W8	1780	W8		1782	
W10	2400	W10		23FE	
ACCA	FF F780 2087	ACCA	00 3800	0000	
Data 1780	671F	Data 1780		671F	
Data 2400	E3DC	Data 2400		E3DC	
CORCON	0000	CORCON		0000	
SR	0000	SR		0000	

Example 2

MPY W6*W7, B, [W8]+=2, W4, [W10]-=2, W5

; Multiply W6*W7 and store to ACCB $\,$

; Fetch [W8] to W4, Post-increment W8 by 2

; Fetch [W10] to W5, Post-decrement W10 by 2

; CORCON = 0x0000 (fractional multiply, no saturation)

	I	Befor nstruct	-		After Instruction			
W4			C000	W4			8FDC	
W5			9000	W5			0078	
W6			671F	W6			671F	
W7			E3DC	W7			E3DC	
W8			1782	W8			1784	
W10			23FE	W10			23FC	
ACCB	00	9834	4500	ACCB	FF	E954	3748	
Data 1782			8FDC	Data 1782			8FDC	
Data 23FE			0078	Data 23FE			0078	
CORCON			0000	CORCON			0000	
SR			0000	SR			0000	

MPY			Square to Ac	cumulator					
Syntax: {	[label:}	MPY	Wm*Wm, Acc	{,[Wx], Wxd} {,[Wx]+=kx, \	•		Wyd} -=kv, Wvd}		
				{,[Wx]-=kx, V					
				{,[W9+W12],					
				(, [110 1112],	Windj (,		· ••••12], •••90]		
Operands:		Acc ∈ [Wx ∈ [m ∈ [W4*W4, W A,B] W8, W9]; kx ∈ [- W10, W11]; ky ∈	6, -4, -2, 2, 4,	6]; Wxd	- ∈ [₩4			
Operation:		([Wx])-	Wm) → Acc(A c → Wxd; (Wx)+kx → Wyd; (Wy)+ky	→Ŵx					
Status Affected	d:	OA, OE	3, OAB, SA, SB,	SAB				_	
Encoding:		111	1 00mm	A0xx	yyi	i	iijj	jj01	
Description:		in prep unspec	Square the contents of a working register, optionally pre-fetch operands in preparation for another MAC type instruction and optionally store the unspecified accumulator results. The 32-bit result of the signed multiply is sign-extended to 40-bits and stored in the specified accumulator.						
		which s	nds Wx, Wxd, W support indirect a n 4.14.1 "MAC	and register of	fset add	-			
		The 'A' The 'x' The 'y' The 'i' l	' bits select the of bit selects the a bits select the p bits select the p bits select the W bits select the W	ccumulator fo re-fetch Wxd o re-fetch Wyd o x pre-fetch op	r the residestination destination destination	ult. on.	square.		
		Note	: The IF bit, C fractional or	CORCON<0>, an integer.	determir	nes if	the multiply i	S	
Words:		1							
Cycles:		1							
Example 1	; Fe	quare M etch [M	5, A, [W9]+=2 N6 and store N9] to W6, Po = 0x0000 (fra	to ACCA st-incremen	-	•	saturation)		
		Be	efore		Af	ter			
		Inst	ruction		Instru	uction			
	W6 W9		6500 0900	W6 W9			365 902		
Д		00 7C	80 0908	ACCA	00 4FE		002		
	0900		B865	Data 0900			365		
COR	CON		0000	CORCON		0 0	000		

```
Example 2
```

MPY W4*W4, B, [W9+W12], W4, [W10]+=2, W5

- ; Square W4 and store to ACCB
- ; Fetch [W9+W12] to W4
- ; Fetch [W10] to W5, Post-increment W10 by 2
- ; CORCON = 0x0000 (fractional multiply, no saturation)

	Before Instruction			I	After nstruct	
W4	E22	8	W4			8911
W5	900	0	W5			F678
W9	170	0	W9			1700
W10	1B0	0	W10			1B02
W12	FF0	0	W12			FF00
ACCB	00 9834 450	0	ACCB	00	06F5	4C80
Data 1600	891	1	Data 1600			8911
Data 1B00	F67	8	Data 1B00			F678
CORCON	000	0	CORCON			0000
SR	000	0	SR			0000

MPY.N	Multiply	y -Wm by Wn to Ac	cumulator					
Syntax: {label:}	MPY.N Wm*Wr	n, Acc {,[Wx], Wxd}	{,[Wy], Wyd}					
		{,[Wx]+=kx, V	Vxd} {,[Wy]+=ky, V	Vyd}				
		{,[Wx]-=kx, W	/xd} {,[Wy]-=ky, W	/yd}				
		{,[W9+W12],	Wxd} {,[W11+W12]	, Wyd}				
Operands:	Acc ∈ [A,B] Wx ∈ [W8, W9]; I	V5; W4*W6; W4*W7 kx ∈ [-6, -4, -2, 2, 4,]; ky ∈ [-6, -4, -2, 2,	6]; Wxd ∈ [W4 V	V7]				
Operation:	$([Wx]) \rightarrow Wxd; (W)$	Wm)*(Wn) → Acc(A or B) Wx])→ Wxd; (Wx)+kx→Wx Wy])→ Wyd; (Wy)+ky→Wy						
Status Affected:	OA, OB, OAB							
Encoding:	1100 Om	umm Alxx	yyii ii	.jj jj11				
	accumulator resu sign-extended to	ype instruction and lts. The 32-bit resul 40-bits and stored t ct the operand regist	t of the signed multi o the specified accu	iply is umulator.				
	The 'x' bits select The 'y' bits select The 'i' bits select The 'j' bits select Note: The If	s the accumulator fo t the pre-fetch Wxd o t the pre-fetch Wyd o the Wx pre-fetch op the Wy pre-fetch op = bit, CORCON<0>,	destination. destination. peration. peration.	ultiply is fractiona				
		integer.						
Words:	1							
Cycles:	1							
; 1 ; 1	Multiply W4*W5, Fetch [W8] to W Fetch [W10] to	[W8]+=2, W4, [W negate the res M4, Post-increme W5, Post-increm (integer multi	ult and store t nt W8 by 2 ent W10 by 2					
	Before		After					
14/4	Instruction		Instruction					
W4 W5			0054 660A					
W8			0B02					
W10			2002					
ACCA	00 0000 2387	7 ACCA	FF FC82 7650					
Data 0B00			0054					
Data 2000			660A					
CORCON			0001					
SR	0000	SR SR	0000					

5

```
Example 2
```

MPY.N W4*W5, A, [W8]+=2, W4, [W10]+=2, W5
; Multiply W4*W5, negate the result and store to ACCA
; Fetch [W8] to W4, Post-increment W8 by 2
; Fetch [W10] to W5, Post-increment W10 by 2

; CORCON = 0x0000 (fractional multiply, no saturation)

	I	Befor nstruct	-	After Instruction				
W4			3023	W4			0054	
W5			1290	W5			660A	
W8			0B00	W8			0B02	
W10			2000	W10			2002	
ACCA	00	0000	2387	ACCA	FF	F904	ECA0	
Data 0B00			0054	Data 0B00			0054	
Data 2000			660A	Data 2000			660A	
CORCON			0000	CORCON			0000	
SR			0000	SR			0000	

Syntax: {label:}	MSC V	Vm*Wn, Acc	{,[Wx], Wxd}		{,[Wy],	Wyd}	{,AWB}
			{,[Wx]+=kx, W	/xd}	{,[Wy]+	·=ky, Wyd}	
			{,[Wx]-=kx, W	xd}	{,[Wy]-	=ky, Wyd}	
			{,[W9+W12], \	Wxd}	{,[W11∙	+W12], Wyd}	
Operands:	$\begin{array}{l} Acc \in [A \\ Wx \in [N \\ Wy \in [N \end{array} \end{array}$,B] /8, W9]; kx ∈	V4*W6, W4*W7 [-6, -4, -2, 2, 4, ∈ [-6, -4, -2, 2, =2]	6]; W	/xd ∈ [V	V4 W7]	[7]
Operation:	(Acc(A c)) $([Wx]) \rightarrow$ $([Wy]) \rightarrow$		r(Wn) → Acc(A x→Wx y→Wy	or B)			
Status Affected:	OA, OB,	OAB, SA, SE	3, SAB				
Encoding:	1100	0 mmm	Alxx	УУ	'ii	iijj	jjaa
Description:	operand store the	s in preparation of unspecified a is sign-extend	of two working on for another r accumulator re led to 40-bits a	MAC ty sults.	vpe insti The 32	ruction and o -bit result of t	ptionally he signe
	which su Section optional	upport indirect 4.14.1 "MAC store of the "d	Vy and Wyd sp and register o Pre-Fetches " other" accumula Write Back".	ffset a '. Ope	ddressi rand AV	ing as descri VB specifies	bed in
	The 'A' b The 'x' b The 'y' b The 'i' bi The 'j' bi	bit selects the bits select the bits select the bits select the N bits select the N	operand regis accumulator fo pre-fetch Wxd pre-fetch Wyd Vx pre-fetch op Vy pre-fetch op accumulator w	or the destin destin peration peration	result. ation. ation. on. on.		ultiply.
	The 'A' b The 'x' b The 'y' b The 'i' bi The 'j' bi	bit selects the bits select the bits select the bits select the bits select the bits select the The IF bit,	accumulator fo pre-fetch Wxd pre-fetch Wyd Vx pre-fetch op Wy pre-fetch op accumulator w CORCON<0>,	or the destin destin peratio peratio peratio	result. nation. nation. on. on. ack dest	tination.	
Words:	The 'A' t The 'x' b The 'y' b The 'i' bi The 'j' bi The 'a' b	bit selects the bits select the bits select the bits select the bits select the bits select the The IF bit,	accumulator fo pre-fetch Wxd pre-fetch Wyd Wx pre-fetch op Wy pre-fetch op accumulator w	or the destin destin peratio peratio peratio	result. nation. nation. on. on. ack dest	tination.	

```
Example 1
```

MSC W6*W7, A, [W8]-=4, W6, [W10]-=4, W7
; Multiply W6*W7 and subtract the result from ACCA
; Fetch [W8] to W6, Post-decrement W8 by 4
; Fetch [W10] to W7, Post-decrement W10 by 4
; CORCON = 0x0001 (integer multiply, no saturation)

Before Instruction				After Instruction		
W6		9051	W6			D309
W7		7230	W7			100B
W8		0C00	W8			OBFC
W10		1C00	W10			1BFC
ACCA	00 0567	8000	ACCA	00	3738	5ED0
Data 0C00		D309	Data 0C00			D309
Data 1C00		100B	Data 1C00			100B
CORCON		0001	CORCON			0001
SR		0000	SR			0000

Example 2

MSC W4*W5, B, [W11+W12], W5, W13 ; Multiply W4*W5 and subtract the result from ACCA

; Fetch [W11+W12] to W5

; Write Back ACCB to W13

; CORCON = 0x0000 (fractional multiply, no saturation)

	Before Instruction					
W4			0500			
W5			2000			
W11			1800			
W12			0800			
W13			6233			
ACCA	00	3738	5ED0			
ACCB	00	1000	0000			
Data 2000			3579			
CORCON			0000			
SR			0000			

	I	After nstruct	
W4			0500
W5			3579
W11			1800
W12			0800
W13			3738
ACCA	00	3738	5ED0
ACCB	00	0EC0	0000
Data 2000			3579
CORCON			0000
SR			0000

MUL		Integer Uns	igned Multi	ply f and W	REG	
Syntax:	{label:}	MUL{.B}	f			
Operands:	f ∈ [0 8	191]				
Operation:	For byte c	peration:				
	(WREG For word of	6)<7:0> * (f)<7	$:0> \rightarrow W2$			
		$(f) * (f) \rightarrow W2:V$	V3			
Status Affected:	None					
Encoding:	1011	1100	OBOf	ffff	ffff	ffff
Description:	register and and the re executed the Most S Least Sign	e default work ad place the re- sult are interp in Byte mode, Significant Wo aificant Word of selects byte of	esult in the W reted as uns the 16-bit re rd of the 32- of the 32-bit r	V2:W3 regist signed intege sult is stored bit result is s result is store	er pair. Both rs. If this inst l in W2. In W tored in W3, ed in W2.	operands truction is ord mode and the
		select the ad			, ,	,
	2:	denote a wo The WREG				
Words:	3: 4:	The IF bit, C This is the o multiply.	ORCON<0>	, has no effe	ct on this ope	
	3: 4: 1	The IF bit, C This is the o	ORCON<0>	, has no effe	ct on this ope	
	3: 4:	The IF bit, C This is the o	ORCON<0>	, has no effe	ct on this ope	
	3: 4: 1 1	The IF bit, C This is the o multiply.	ORCON<0>	, has no effe n which prov	ct on this ope vides for an 8	
Cycles:	3: 4: 1 1	The IF bit, C This is the o multiply.	ORCON<0> nly instructio	, has no effe n which prov	ct on this ope vides for an 8	
Cycles: Example 1 MUI	3: 4: 1 1 . B 0x800 Before Instruction	The IF bit, C This is the o multiply. ; Multip	ORCON<0> nly instructio	, has no effe n which prov *WREG (Byt	ct on this ope vides for an 8	
Example 1 MUI WREG (W0)	3: 4: 1 1 	The IF bit, C This is the o multiply. ; Multip WREG (W	ORCON<0> nly instructio	, has no effe n which prov *WREG (Byt	ct on this ope vides for an 8	
Cycles: Example 1 MUI WREG (W0) W2	3: 4: 1 1 B 0x800 Before Instruction 9823 FFFF	The IF bit, C This is the o multiply. ; Multip WREG (W W	ORCON<0> nly instructio	, has no effe n which prov *WREG (Byt	ct on this ope vides for an 8	
Cycles: Example 1 MUI WREG (W0) W2 W3	3: 4: 1 1 B 0x800 Before Instruction 9823 FFFF FFFF	The IF bit, C This is the o multiply. ; Multip WREG (W V V	ORCON<0> nly instructio	, has no effe n which prov *WREG (Byt	ct on this ope vides for an 8	
Cycles: Example 1 MUI WREG (W0) W2	3: 4: 1 1 . B 0x800 Before Instruction 9823 FFFF FFFF 2690	The IF bit, C This is the o multiply. ; Multip WREG (W W U Data 080	ORCON<0> nly instructio	, has no effe n which prov *WREG (Byt	ct on this ope vides for an 8	
Cycles: Example 1 MUI WREG (W0) W2 W3 Data 0800	3: 4: 1 1 B 0x800 Before Instruction 9823 FFFF FFFF 2690 0000	The IF bit, C This is the o multiply. ; Multip WREG (W W V Data 080 S	ORCON<0> nly instructio	, has no effe n which prov *WREG (Byth	ct on this ope vides for an 8	
Cycles: Example 1 MUI WREG (W0) W2 W3 Data 0800 SR Example 2 MUI	3: 4: 1 1 B 0x800 Before Instruction 9823 FFFF FFFF 2690 0000 TMR1 Before	The IF bit, C This is the o multiply. ; Multip WREG (W W V Data 080 S	ORCON<0> nly instructio	, has no effe n which prov *WREG (Byt n WREG (Word	ct on this ope vides for an 8	
Cycles: Example 1 MUI WREG (W0) W2 W3 Data 0800 SR Example 2 MUI	3: 4: 1 1 1 . B 0x800 Before Instruction 9823 FFFF FFFF 2690 0000 . TMR1 Before Instruction	The IF bit, C This is the o multiply. ; Multip WREG (W W Data 080 S ; Multip	ORCON<0> nly instructio ly (0x800) After Instructior 0) 9823 V2 13B0 V3 FFFF 00 2690 SR 00000 ly (TMR1)* After Instruction	, has no effe n which prov *WREG (Byt n WREG (Word	ct on this ope vides for an 8	
Cycles: Example 1 MUI WREG (W0) W2 W3 Data 0800 SR Example 2 MUI WREG (W0)	3: 4: 1 1 1 . B 0x800 Before Instruction 9823 FFFF FFFF 2690 0000 . TMR1 Before Instruction F001	The IF bit, C This is the o multiply. ; Multip WREG (W W Data 080 S ; Multip WREG (W	ORCON<0> nly instructio ly (0x800) After Instruction 0) 9823 V2 13B0 V3 FFFF 00 2690 SR 0000 ly (TMR1)* After Instruction 0) F001	, has no effe n which prov *WREG (Byt n WREG (Word	ct on this ope vides for an 8	
Cycles: Example 1 MUI WREG (W0) W2 W3 Data 0800 SR Example 2 MUI WREG (W0) W2	3: 4: 1 1 B 0x800 Before Instruction 9823 FFFF 2690 0000 TMR1 Before Instruction F001 0000	The IF bit, C This is the o multiply. ; Multip WREG (W W Data 080 S ; Multip WREG (W V	ORCON<0> nly instructio ly (0x800) After Instructior 0) 9823 V2 13B0 V3 FFFF 00 2690 SR 0000 ly (TMR1)* After Instruction V0 F001 V2 C287	, has no effe n which prov *WREG (Byt n WREG (Word	ct on this ope vides for an 8	
Cycles: Example 1 MUI WREG (W0) W2 W3 Data 0800 SR Example 2 MUI WREG (W0)	3: 4: 1 1 B 0x800 Before Instruction 9823 FFFF 2690 0000 TMR1 Before Instruction F001 0000	The IF bit, C This is the o multiply. ; Multip WREG (W W Data 080 S ; Multip WREG (W V	ORCON<0> nly instructio ly (0x800) After Instruction 0) 9823 V2 13B0 V3 FFFF 00 2690 CR 0000 ly (TMR1)* After Instruction 0) F001 V2 C287 V3 2F5E	, has no effe n which prov *WREG (Byt n WREG (Word	ct on this ope vides for an 8	

MUL.S	5	Integer 16x16-bit Signed Multiply						
Syntax:	{label:}	MUL.SS	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wnd			
Operands:	$\label{eq:Wb} \begin{array}{l} \mbox{\boldmath \in} [W0 \ . \\ Ws \ \mbox{\boldmath \in} [W0 \ . \\ Wnd \ \mbox{\boldmath \in} [W0 \ . \end{array} \end{array}$	W15]	. W12]					
Operation:	signed (Wb) * signed ($Ws) \rightarrow Wnd:$	Nnd+1				
Status Affected	: None							
Encoding:	1011	1001	lwww	wddd	dppp	SSSS		
	Both source complemen for Wb and used for Ws The 'w' bits The 'd' bits The 'p' bits	e operands it signed int Wnd. Regis s. select the a select the a select the s	Significant W and the resul egers. Regist ster direct or address of the address of the address of the address of the	t Wnd are in er direct add register indir e base regist e lower destin ss mode.	terpreted as Iressing mus ect addressir er. nation registe	two's t be used ng may be		
Words:	2: 3:	Since the p an even wo how double Wnd may n	tion operates roduct of the r orking registe -words are a ot be W14, s CORCON<0>	nultiplication r. See Figure ligned in mer ince W15<0	is 32-bits, W e 4-2 for info mory. > is fixed to z	rmation or ero.		
Cycles:	1							
Example 1	MUL.SS W0, W1 Before Instruction W0 9823		; Multiply ; Store the After Instructior V0 9823	e result to	o W12:W13			
	W1 67DC W12 FFFF W13 FFFF SR 00000	W W	V1 67DC 12 D314 13 D5DC SR 0000					

Example 2	MUL.SS	W2,	[W4]
-----------	--------	-----	------

; Pre-decrement W4 ; Multiply W2*[W4] ; Store the result to W0:W1

I	Before Instructior	ı	After Instruction
W0	FFFF	W0	28F8
W1	FFFF	W1	0000
W2	0045	W2	0045
W4	27FE	W4	27FC
Data 27FC	0098	Data 27FC	0098
SR	0000	SR	0000

, WO

NII

.

MUL.SU	L.SU Integer 16x16-bit Signed-Unsigned Short Liter					
Syntax:	{label:}	MUL.SU	Wb,	#lit5,	Wnd	
Operands:	Wb ∈ [W0 lit5 ∈ [0 Wnd ∈ [M	-	W12]			
Operation:	signed (W	/b) * unsigned	l lit5 \rightarrow Wnd:	Wnd+1		
Status Affected:	None					
Encoding:	1011	1001	0www	wddd	d11k	kkkk
	the result register), The Wb o compleme integer. R	vo successive is stored in W and the Most perand and th ent signed inte egister direct ts select the a	/nd (which m Significant W he result Who eger. The lite addressing n	ust be an eve ord of the res are interpre ral is interpre nust be used	en numbered sult is stored sted as a two sted as an un for Wb and	ł working in Wnd+1. 's isigned
	The 'd' bit	s select the a s define a 5-b	ddress of the	lower destir	nation registe	er.
	2: 3:		oduct of the r rking registe -words are al ot be W14, si	nultiplication r. See Figure igned in mer nce W15<0>	is 32-bits, W e 4-2 for info nory. • is fixed to z	rmation or ero.
Words:	1					
Cycles:	1					
Example 1	MUL.SU W0,	#0x1F, W2		W0 by lit Ne result t		
	Before		After			

I	After Instruction		
W0	C000	W0	C000
W2	1234	W2	4000
W3	C9BA	W3	FFF8
SR	0000	SR	0000

Example 2	MUL.SU	W2,	#0x10,			y W2 by literal 0x10 he result to W0:W1
	Before Instruction				After Instruction	n
	inst	uction			manucio	
	W0 2	ABCD		W0	2400	
	W1 8	39B3		W1	000F	
	W2 1	7240		W2	F240	
	SR	0000		SR	0000	

© 2003 Microchip Technology Inc.

MUL.SU		Integer 16	x16-bit Sign	ed-Unsigne	d Multiply	
Syntax:	{label:}	MUL.SU	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wnd	
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wnd ∈ [W	-	W12]			
Operation:	signed (W	/b) * unsigne	d (Ws) \rightarrow Wr	nd:Wnd+1		
Status Affected:	None					
Encoding:	1011	1001	0www	wddd	dppp	SSSS
	The Wb c compleme unsigned Wnd. Reg The 'w' bi The 'd' bit The 'p' bit The 's' bit Note 1 2:	perand and t ent signed in integer. Regi jister direct o ts select the s select the an even w how double Wnd may r	the result Wn teger. The W ister direct ac r register ind address of th address of th source Addre address of the ction operate roduct of the orking registe e-words are a not be W14, s	e source regi s in Word mo multiplicatior er. See Figur aligned in me since W15<0:	eted as a two interpreted a: st be used fo ing may be use ter. nation registe ster. de only. n is 32-bits, W e 4-2 for info mory. > is fixed to z	's s an r Wb and sed for Ws rr. rnd must be rmation or ero.
Words:	4 : 1	The IF bit,	CORCON<0	>, has no effe	ect on this op	eration.
Cycles:	1					
-			; Multiply ; Store th	W8*[W9] e result t	o W0:W1	
W0 W1 W8 W9 Data 178C SR	AA40 F000 178C F000	\ \ Data 17	After Instructio W0 0000 W1 F100 W8 F000 W9 178C 8C F000 SR 0000	n - - -		

Example 2 MU	L.SU W2,	, [++W3], W4	; Pre-Increment W3 ; Multiply W2*[W3] ; Store the result to W4:W5	
	Before		After	
	Instruction	ו	Instruction	
W2	0040	W2	2 0040	
W3	0280	W3	3 0282	
W4	1819	W4	1A00	
W5	2021	W5	5 0000	
Data 0282	0068	Data 0282	2 0068	
SR	0000	SR	R 0000	

Syntax:	{label:}	MUL.US	Wb,	Ws,	Wnd	
	[,	[Ws],		
				[Ws++],		
				[Ws],		
				[++Ws],		
				[Ws],		
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wnd ∈ [W	-	. W12]			
Operation:	unsigned	(Wb) * signe	d (Ws) $ ightarrow$ V	Vnd:Wnd+1		
Status Affected:	None					
Encoding:	1011	1000	lwww	wddd	dppp	SSSS
	Register direct or register indirect addressing may be used for Ws. The 'w' bits select the address of the base register. The 'd' bits select the address of the lower destination register. The 'p' bits select the source Address mode. The 's' bits select the address of the source register.					
	2:	Since the p an even we	roduct of th orking regis -words are	es in Word mo e multiplication ster. See Figur aligned in me	n is 32-bits, W re 4-2 for info mory.	ormation or ero.
				0>, has no eff		eration.
Words:						eration.
Words: Cycles:	4:					eration.
Cycles:	4 : 1 1	The IF bit, o	CORCON<		ect on this op unsigned-s	
Cycles: Example 1 MUL	4: 1 1 2.US W0, Before	The IF bit, o	; Multipl ; Store t After	0>, has no eff y W0*[W1] (he result t	ect on this op unsigned-s	
Cycles: Example 1 MUI	4 1 1 .US W0, Before Instruction	: The IF bit, ([W1], W2	; Multipl ; Store t After Instructi	0>, has no eff y W0*[W1] (he result t	ect on this op unsigned-s	
Cycles: Example 1 MUL W0	4: 1 1 2.US W0, Before Instruction	: The IF bit, ([W1], W2	; Multipl ; Store t After Instructi	0>, has no eff y W0*[W1] (he result t on	ect on this op unsigned-s	
Cycles: Example 1 MUI	4: 1 1 US W0, Before Instruction C000 2300	: The IF bit, ([W1], W2 V V	; Multipl ; Store t After Instructi	0>, has no eff y W0*[W1] (he result t	ect on this op unsigned-s	
Cycles: Example 1 MUL W0 W1	4: 1 1 US W0, Before Instruction C000 2300 00DA	: The IF bit, ([W1], W2 [W1], W2 [W1], W2 [W1], W2 [W1], W2 [W1], W2 [W1], W2 [W1], W2 [W1], W2 [W1], W2	; Multipl ; Store t After Instructi V0 C000 V1 2300	0>, has no eff y W0*[W1] (he result t	ect on this op unsigned-s	

Example 2	MUL.US	W6,	[W5++],	; 5	Store		(unsigned-signed) ult to W10:W11 t W5
	Befo			Ing	After	n	

I	nstructior	1	Instructio
W5	0C00	W5	0C02
W6	FFFF	W6	FFFF
W10	0908	W10	8001
W11	6EEB	W11	7FFE
Data 0C00	7FFF	Data 0C00	7FFF
SR	0000	SR	0000

MUL.U	U	Integer 16	x16-bit Un	signed Sho	rt Literal Multi	ply			
Syntax:	{label:}	MUL.UU	Wb,	#lit5,	Wnd				
Operands:	lit5 ∈ [0	0 W15] 31] V0, W2, W4 .	W12]						
Operation:	unsigned	(Wb) * unsig	ned lit5 \rightarrow	Wnd:Wnd+1					
Status Affected	d: None								
Encoding:	1011	1000	0www	wddd	d11k	kkkk			
Description:	result in t the result register), Both ope Register	wo successiv is stored in V and the Most rands and the direct addres	e working r Wnd (which Significan e result are sing must l	egisters. The must be an t Word of the interpreted a be used for V		ant Word o d working l in Wnd+1			
	The 'd' bi	The 'w' bits select the address of the base register. The 'd' bits select the address of the lower destination register. The 'k' bits define a 5-bit unsigned integer literal.							
	2 3 4	an even w how double : Wnd may r	roduct of th orking regi e-words are not be W14	ne multiplicat ster. See Fig aligned in n , since W15<	ion is 32-bits, W jure 4-2 for info	ormation o zero.			
Words:	1								
Cycles:	1								
Example 1	MUL.UU WO,	#0xF, W12			literal 0xF t to W12:W13				
	Before Instruction W0 2323 W12 4512 W13 7821 SR 0000	W W	After Instruct M0 232 12 0F0 13 000 SR 000	ion 3 D 2					
Example 2	MUL.UU W7,	#0x1F, W0			literal 0x1F t to W0:W1	,			
	Before Instruction W0 780B W1 3805 W7 F240 SR 00000	١	After Instruct W0 55C W1 001 W7 F24 SR 000	ion 0 D 0					

MUL.UU	J	Integer 16x16-bit Unsigned Multiply							
Syntax:	{label:}	MUL.UU	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wnd				
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wnd ∈ [W		. W12]						
Operation:	unsigned	(Wb) * unsigr	ned (Ws) \rightarrow V	Vnd:Wnd+1					
Status Affected:	None								
Encoding:	1011	1000	0www	wddd	dppp	SSSS			
	Both sourd integers. F Register d The 'w' bit The 'd' bit The 'p' bit	ce operands a Register direct lirect or indirects s select the a s select the a s select the s	Significant W and the resul addressing address of the ddress of the ource Address ddress of the	t are interpre must be use g may be use base regist lower destir s mode.	ted as unsig d for Wb and ed for Ws. er. nation registe	ned I Wnd.			
		Since the pr	tion operates oduct of the r		is 32-bits, Wi	nd must b			
Words:	4 : 1	how double Wnd may ne	-words are al ot be W14, si CORCON<0>	igned in mer nce W15<0>	nory. · is fixed to ze	rmation or ero.			
Words: Cycles: Example 1	4:	how double Wnd may n The IF bit, C	-words are al ot be W14, si	igned in mer nce W15<0> , has no effe	nory. · is fixed to ze ct on this ope	rmation of ero. eration.			

		; Store the result to W4:W ; Post-Increment W1							
I	Before nstructior	1	After nstructior	ı					
W0	1024	W0	1024						
W1	2300	W1	2302						
W4	9654	W4	6D34						
W5	BDBC	W5	0D80						
Data 2300	D625	Data 2300	D625						
SR	0000	SR	0000						

Example 2

MUL.UU W0, [W1++], W4 ; Mult. W0*[W1] (unsigned-unsigned)

NEG		Negate f				
Syntax:	{label:}	NEG{.B}	f	{,WREG}		
Operands:	f ∈ [0 8	191]				
Operation:	-	destination de	signated by	D		
Status Affected:	DC, N, O\		0 ,			
Encoding:	1110	1110	OBDf	ffff	ffff	ffff
Description:	place the determine	the 2's compl result in the d s the destina VREG. If WR	estination regition register.	gister. The o _l If WREG is s	otional WRE	G operand result is
	The 'D' bit	selects byte selects the c select the ac	lestination (0	for WREG, 2		
			a word opera ord operation	tion. You ma , but it is not	y use a . w ex required.	
Words:	1					
Cycles:	1					
Example 1 NEG.	B 0x880,		legate (0x8 tore resul	-	mode)	
lı WREG (W0) Data 0880 SR	Before nstruction 9080 2355 0000	WREG (W0 Data 0880 SF	0 2355	J=1)		
Example 2 NEG	0x1200	; N	legate (0x1	.200) (Word	d mode)	
lı Data 1200 SR	Before nstruction 8923 0000	Data 120 SF				

NEG		Negate W	S			
Syntax:	{label:}	NEG{.B}	Ws,	Wd		
			[Ws],	[Wd]		
			[Ws++],	[Wd++]		
			[Ws],	[Wd]		
			[++Ws],	[++Wd]		
			[Ws],	[Wd]		
Operands:		0 W15] 0 W15]				
Operation:	(Ws) + 1	\rightarrow Wd				
Status Affected:	DC, N, O	V, Z, C				
Encoding:	1110	1010	0Bqq	qddd	dppp	SSSS
Description:	and place	e the result in	the destinati	e contents of on register W I for both Ws	/d. Either reg	
	The 'q' bi The 'd' bi The 'p' bi	ts select the s ts select the s ts select the s	destination A address of th source Addre	eration (0 for ddress mode e destination ess mode. e source regi	register.	/te).
		rather than a	word operati	nstruction de on. You may but it is not re	use a .wex	-
Words:	1		•			
	1					
Cycles:						
Cycles: Example 1 NEG.B	W3, [W4	-	ate W3 and t-incremen	store to t W4	[W4] (Byte	mode)
	W3, [W4 Before	-			[W4] (Byte	mode)
Example 1 NEG.B	Before	; Pos	t-incremen After Instruction		[W4] (Byte	mode)
Example 1 NEG.B	Before Instruction	; Pos	t-incremen After Instruction /3 7839		[W4] (Byte	mode)
Example 1 NEG.B	Before nstruction 7839 1005	; Pos M M	After Instruction /3 7839 /4 1006		[W4] (Byte	mode)
Example 1 NEG.B	Before Instruction	; Pos W Data 100	After Instruction /3 7839 /4 1006	t W4	[W4] (Byte	mode)
Example 1 NEG.B	Before nstruction 7839 1005 2355	; Pos W Data 100 S W4] ; P ; N	After Instruction /3 7839 /4 1006 04 C755 R 0008 (re-decreme	t W4 N=1) nt W4 (Wor and store	d mode)	mode)
Example 1 NEG.B	Before instruction 7839 1005 2355 0000	; Pos W Data 100 S W4] ; P ; N ; P	After Instruction /3 7839 /4 1006 04 C755 /R 0008 (re-decreme egate [W2] ost-increm After Instruction /2 /2 0902 /4 1000 00 870F	t W4 N=1) nt W4 (Wor and store	d mode)	mode)

NEG	Ne	egate Acci	umulator			
Syntax:	{label:} NE	ĒG	Acc			
Operands:	$Acc \in [A,B]$					
Operation:	<u>If (Acc = A):</u> -ACCA → A	CCA				
	<u>Else:</u> -ACCB → A	CCB				
Status Affected:	OA, OB, OAB,	, SA, SB, S	SAB			
Encoding:	1100	1011	A001	0000	0000	0000
Description:	Compute the 2 accumulator. F operates on al	Regardless	s of the Satu	uration mode		ion
	The 'A' bit spe	cifies the s	elected acc	cumulator.		
Words:	1					
Cycles:	1					
			t to ACCA 0000 (no	saturation	1)	
	Before			After		
ACC	Instruction	10	ACCA	Instructio		
CORCO		-	CORCON		000	
SF		-	SR	-	000	
Example 2 N	IEG B ; Nea	ate ACCB	-			
	, ,			3		
	; Sto	re resul	t to ACCE	s mal satura	tion)	
	; Sto ; COR Before	re resul	t to ACCE	rmal satura After		
	; Sto ; COR Before Instruction	re resul CON = 0x	t to ACCE	mal satura After Instructio	n	
ACCI	; Sto ; COR Before Instruction 3 FF F230 10D	Pre resul CON = 0x	t to ACCE 00C0 (nor ACCB	mal satura After Instructio	n SF24	
ACCI CORCON SF	; Sto ; COR Before Instruction B FF F230 10D N 00C	CON = 0x	t to ACCE	mal satura After Instructio 00 0DCF E 0	n	

NOP			No Operatio	on			
Syntax:	{lat	oel:}	NOP				
Operands:	No	ne					
Operation:	No	Operat	ion				
Status Affecte	d: No	ne					
Encoding:	(0000	0000	xxxx	xxxx	xxxx	xxxx
Description:	No	Operat	ion is perform	ed.			1
	The	e 'x' bits	s can take any	value.			
Words:	1						
Cycles:	1						
Example 1	Instru	; ex fore uction 1092 0000	ecute no o <u>r</u>		After struction 00 1094 0000		
Example 2	NOP	; ex	ecute no o <u>p</u>	peration			
	Instru	fore uction 08AE 0000			After struction 00 08B0 0000		

NOPR		No Operation
Syntax:	{label:}	NOPR
Operands:	None	
Operation:	No Operat	tion
Status Affected:	None	
Encoding:	1111	1111 xxxx xxxx xxxx xxxx
Description:	No Operat	tion is performed.
	The 'x' bits	s can take any value.
Words:	1	,
Cycles:	1	
PC	Before Instruction	After Instruction PC 00 2432
SR	0000	SR 0000
Example 2 NOP	PR ; ex	xecute no operation
Example 2 NOF	PR ; ex Before	After
	Before Instruction	After
	Before	After Instruction PC 00 1468

POP		Pop TOS to	f			
Syntax:	{label:}	POP	f			
Operands:	f ∈ [0 65	5534]				
Operation:	(W15)-2 → (TOS) → f	• W15				
Status Affected:	None					
Encoding:	1111	1001	ffff	ffff	ffff	fff0
Description:	(TOS) wor anywhere	pointer (W15 d is written to in the lower 3	the specifie 2K words of	d file register data memor	, which may	
	The 'f' bits	select the ad	dress of the	file register.		
		This instruct The file regis				
Words:	1	Ū			U	
Cycles:	1					
Example 1 POP	0x1230	; Рор ТО	S to 0x123	0		
In W15 Data 1004 Data 1230 SR	Before struction 1006 A401 2355 0000	W15 Data 1004 Data 1230 SF	A401 A401			
Example 2 POP	0x880	; Pop TO	S to 0x880			
In W15 Data 0880 Data 1FFE SR	Before struction 2000 E3E1 A090 0000	W15 Data 0880 Data 1FFE SF	A090 A090			

POP		Pop TOS to	Wd			
Syntax:	{label:}	POP	Wd			
			[Wd]			
			[Wd++]			
			[Wd]			
			[Wd]			
			[++Wd]			
			[Wd+Wb]			
Operands:	Wd ∈ [W0 Wb ∈ [W0					
Operation:	(W15)-2 - (TOS) →					
Status Affected:	None					
Encoding:	0111	lwww	w0hh	hddd	d100	1111
Description:	(TOS) wo	pointer (W15 rd is written to sed for Wd.				
	The 'h' bit	ts define the o s select the de s select the ac	estination Add	lress mode.		
	Note 1:	This instruct	ion operates i	n Word mo	de only.	
	Note 1:	This instruct This instruct	ion operates i	n Word mo ific version	de only. of the "MOV	
Words:	Note 1:	This instruct This instruct instruction (i	ion operates i ion is a spec	n Word mo ific version	de only. of the "MOV	
Words: Cycles:	Note 1: 2:	This instruct This instruct instruction (i	ion operates i ion is a spec	n Word mo ific version	de only. of the "MOV	
	Note 1: 2:	This instruct This instruct instruction (i	ion operates i ion is a spec MOV [W15	n Word mo ific version	de only. of the "MOV	
Cycles:	Note 1: 2: 1 1	This instruct This instruct instruction (1 MOV.	ion operates i ion is a spec MOV [W15 S to W4 After	n Word mo ific version	de only. of the "MOV	
Cycles: Example 1 POP	Note 1: 2: 1 1 W4 Before nstruction	This instruct This instruct instruction (1 MOV. ; Pop TO	ion operates i tion is a spec MOV [W15 S to W4 After Instruction	n Word mo ific version	de only. of the "MOV	
Cycles: Example 1 POP I W4	Note 1: 2: 1 1 W4 Before nstruction EDA8	This instruct This instruct instruction (i MOV. ; Pop TO	ion operates i ion is a spec MOV [W15 S to W4 After Instruction	n Word mo ific version	de only. of the "MOV	
Cycles: Example 1 POP I W4 W15	Note 1: 2: 1 1 W4 Before nstruction EDA8 1008	This instruct This instruct instruction (1 MOV. ; Pop TO W4 W15	ion operates i ion is a spec MOV [W15 S to W4 After Instruction C45A 1006	n Word mo ific version	de only. of the "MOV	
Cycles: Example 1 POP	Note 1: 2: 1 1 W4 Before nstruction EDA8	This instruct This instruct instruction (i MOV. ; Pop TO	ion operates i ion is a spec MOV [W15 S to W4 After Instruction C45A 1006 C45A	n Word mo ific version	de only. of the "MOV	
Cycles: Example 1 POP I W4 W15 Data 1006	Note 1: 2: 1 1 W4 Before nstruction EDA8 1008 C45A	This instruct This instruct instruction (1 MOV. ; Pop TO W4 W15 Data 1006 SF ; Pre-in	ion operates i ion is a spec MOV [W15 S to W4 After Instruction C45A 1006 C45A	n Word moo ific version ;], wd). It	de only. of the "MOV	
Cycles: Example 1 POP I W4 W15 Data 1006 SR Example 2 POP	Note 1: 2: 1 1 1 W4 Before nstruction EDA8 1008 C45A 0000 [++W10] Before	This instruct This instruct instruction (1 MOV. ; Pop TO W4 W15 Data 1006 SF ; Pre-in	ion operates i ion is a spec MOV [W15 S to W4 After Instruction C45A 1006 C45A 0000 crement W10 S to [W10] After	n Word moo ific version ;], wd). It	de only. of the "MOV	
Cycles: Example 1 POP I W4 W15 Data 1006 SR Example 2 POP	Note 1: 1 1 W4 Before nstruction EDA8 1008 C45A 0000 [++W10] Before nstruction	This instruct This instruct instruction (f MOV. ; Pop TO W4 W15 Data 1006 SF ; Pre-in ; Pop TO	ion operates i tion is a spec MOV [W15 S to W4 After Instruction C45A 0000 C45A 0000 crement W10 S to [W10] After Instruction	n Word moo ific version i], wd). It	de only. of the "MOV	
Cycles: Example 1 POP I W4 W15 Data 1006 SR Example 2 POP	Note 1: 2: 1 1 1 W4 Before nstruction EDA8 1008 C45A 0000 [++W10] Before	This instruct This instruct instruction (1 MOV. ; Pop TO W4 W15 Data 1006 SF ; Pre-in	ion operates i tion is a spec MOV [W15 S to W4 After Instruction C45A 0000 Crement W10 S to [W10] After Instruction 0 0E04	n Word moo ific version i], wd). It	de only. of the "MOV	
Cycles: Example 1 POP I W4 W15 Data 1006 SR Example 2 POP	Note 1: 2: 1 1 W4 Before nstruction EDA8 1008 C45A 0000 [++W10] Before nstruction 0E02	This instruct This instruct instruction (i MOV. ; Pop TO W4 W15 Data 1006 SF ; Pre-in ; Pop TO W10	ion operates i tion is a spec MOV [W15 S to W4 After Instruction C45A 1006 C45A 0000 crement W10 S to [W10] After Instruction 0E04 1764	n Word moo ific version i], wd). It	de only. of the "MOV	
Cycles: Example 1 POP I W4 W15 Data 1006 SR Example 2 POP I W10 W15	Note 1: 1 1 W4 Before nstruction EDA8 1008 C45A 0000 [++W10] Before nstruction 0E02 1766	This instruct This instruct instruction (in MOV. ; Pop TO W4 W15 Data 1006 SF ; Pre-in ; Pop TO W10 W15	ion operates i ion is a spec MOV [W15 S to W4 After Instruction C45A 0000 Crement W10 S to [W10] After Instruction S to [W10] After Instruction 0E04 1764 C7B5 C7B5	n Word moo ific version i], wd). It	de only. of the "MOV	

© 2003 Microchip Technology Inc.

POP.D		Double Pop	TOS to Wr	d:Wnd+1		
Syntax:	{label:}	POP.D	Wnd			
Operands: Operation:	Wnd \in [W (W15)-2 – (TOS) \rightarrow \ (W15)-2 – (TOS) \rightarrow \	Vnd+1 → W15	. W14]			
Status Affected:	None					
Encoding:	1011	1110	0000	0ddd	0100	1111
Description:	Wnd:Wnd Least Sigr	word is poppe +1. The Most ificant Word i pointer (W15)	Significant V s stored to W	Vord is stored Ind. Since a d	d to Wnd+1, a	and the
	The 'd' bits	s select the a	ddress of the	destination	register pair.	
	2:	Wnd must b This instruct	on how doub e an even we ion is a spec	le-words are orking registe fic version of	aligned in m er.	emory. Ws, Wnd"
Words:	1					
Cycles:	2					
Example 1 POP.	D W6	; Doub	le pop TOS	to W6		
اr W6 W7 W15 Data 084C Data 084E SR	Before nstruction 07BB 89AE 0850 3210 7654 0000	W6 W7 W15 Data 084C Data 084E SF	7654 084C 3210 7654			
Example 2 POP.	D WO	; Doub	le pop TOS	to WO		
Ir W0 W1 W15 Data 0BB8 Data 0BBA SR	Before nstruction 673E DD23 0BBC 791C D400 0000	W0 W15 Data 0BB8 Data 0BBA SF	D400 0BB8 791C D400			

POP.S		Pop Shado	w Registers			
Syntax:	{label:}	POP.S				
Operands:	None					
Operation:	Pop shad	low registers				
Status Affected:	DC, N, O	-				
Encoding:	1111	1110	1000	0000	0000	0000
Description:	primary re	es in the shado egisters. The f N and DC Sta	ollowing regi	sters are affe	•	
		The shadow only be acce The shadow	essed with Pt	JSH.S and P	OP.S.	They may
Words:	1					
Cycles:	1					
Example 1 PO	-	Pop the sha (See PUSH.S	5		ents of sha	adows)
	Before		After			
	Instruction		Instruction	l		
W	0 07BB	١	VO 0000			
W			V1 1000			
W			V2 2000			
W			V3 3000			
SI	R 00E0 (II	PL=7) \$	SR 00E1	(IPL=7, C=1))	

Note: After instruction execution, contents of shadow registers are NOT modified.

	Push f to T	os			
{label:}	PUSH	f			
f ∈ [0 65	5534]				
None					
1111	1000	ffff	ffff	ffff	fff0
(TOS) loca	tion and then	the stack po	ointer (W15)	is incremente	ed by 2.
The 'f' bits	select the ad	dress of the	file register.		
1					
1					
0x2004	; Pus	sh (0x2004) to TOS		
Before		After			
0000					
0xC0E	; Pusl	n (OxCOE)	to TOS		
Before struction		After Instruction			
0920	W15	0922			
	D-1- 0000	6			
0000	Data 0920 Data 2004	-			
	f ∈ [0 65 (f) → (TOS (W15)+2 - None 1111 The conter (TOS) loca The file reg memory. The 'f' bits Note 1: 2: 1 1 0x2004 Before struction 0B00 791C D400 0000 0xC0E Before struction 0920	{label:}PUSH $f \in [0 \dots 65534]$ (f) \rightarrow (TOS) $(W15)+2 \rightarrow W15$ None11111000The contents of the spect(TOS) location and thenThe file register may restmemory.The 'f' bits select the addressNote 1: This instruction0x2004; PustBeforestruction0B00W15791CData 0B00D400Data 20040xC0E; PustBeforestruction0xC0E; PustBeforestruction0x2004W15791CData 0B00D400D400W150x201W15	$f \in [0 \dots 65534]$ (f) \rightarrow (TOS) (W15)+2 \rightarrow W15 None 1111 1000 ffff The contents of the specified file reg (TOS) location and then the stack por The file register may reside anywher memory. The 'f' bits select the address of the Note 1: This instruction operates 2: The file register address 1 1 0x2004 ; Push (0x2004 Before After struction Instruction 0B00 791C Data 0B00 D400 D400 0000 XR 0000 0xC0E ; Push (0xC0E) Before After struction Instruction 0xC0E ; Push (0xC0E)	{label:}PUSHf $f \in [0 \dots 65534]$ (f) \rightarrow (TOS) $(W15)+2 \rightarrow W15$ None1111100011111000ffffffffThe contents of the specified file register are writ(TOS) location and then the stack pointer (W15)The file register may reside anywhere in the lowermemory.The 'f' bits select the address of the file register.Note 1:Note 1:This instruction operates in Word mode2:The file register address must be word110x2004;Push (0x2004)to TOSBeforestruction0B00791CData 0B00D4000000SR00000xC0E;Push (0xC0E)to TOSBeforeAfterInstruction0x2004W150920W150920W150920	

		Push Ws	to TOS			
Syntax:	{label:}	PUSH	Ws [Ws] [Ws++] [Ws] [Ws] [++Ws] [Ws+Wb]			
Operands:	$Ws \in [W0]$ $Wb \in [W0]$					
Operation:	(Ws) → (T (W15)+2 –					
Status Affected:	None					
Encoding: Description:			w001 e written to th (W15) is incre			ssss cation and
	Note 1:	This instruc This instruc	ddress of the tion operates tion is a spe (MOV Ws,	in Word mo cific version	de only. of the "MON	
Words:	1	MOV.				
	•					
Cycles:	1					
Cycles: Example 1 PUSH		; Pusl	n W2 to TOS	1		
Example 1 PUSH		; Pusl W W1 Data 156 S	After Instruction 2 6889 5 1568 6 6889			
Example 1 PUSH In W2 W15 Data 1566	Before nstruction 6889 1566 0000 0000	W W1 Data 156 S	After Instruction 2 6889 5 1568 6 6889			

PUSH.D		Double Pus	sh Wns:Wns	+1 to TOS		
Syntax:	{label:}	PUSH.D	Wns			
Operands:	-	0, W2, W4	W14]			
Operation:	(Wns) → ((W15)+2 - (Wns+1) - (W15)+2 -	→ W15 → (TOS)				
Status Affected:	None					
Encoding:	1011	1110	1001	1111	1000	sss0
Description:	Least Sigr Significant	word (Wns:W hificant word (t word (Wns+ ord is pushed,	Wns) is push 1) is pushed	ied to the TC to the TOS I	OS first, and fast. Since a	the Most
	The 's' bits	s select the ad	ddress of the	source regis	ster pair.	
		for informati Wns must b This instruct instruction (1 as MOV.D.	e an even wo ion is a speci	orking registe fic version of	er. fthe "MOV.D	Wns, Wd"
Words:	1					
Cycles:	2					
Example 1 PUSH	I.D W6	; Pu	sh W6:W7 t	o TOS		
	Before		After			
г	nstruction	10/0	Instruction			
W6 W7	C451 3380	W6 W7				
W15	1240	W15				
Data 1240	B004	Data 1240				
Data 1242	0891	Data 1242	3380			
SR	0000	SF	0000			
Example 2 PUSE	H.D W10	; P	ush W10:W1	1 to TOS		
	Before		After			
-	nstruction	1.8.1.4.4	Instruction			
W10 W11	80D3	W10 W11				
W11	4550 0C08	W15				
Data 0C08	79B5	Data 0C08				
-						
Data 0C0A	008E	Data 0C0A	4550			

PUSH.S		Push Shade	ow Register	s		
Syntax:	{label:}	PUSH.S				
Operands:	None					
Operation:	Push sha	dow registers				
Status Affected:	None					
Encoding:	1111	1110	1010	0000	0000	0000
Description:	shadow re	nts of the prime gisters. The f DV, N and DC	ollowing regi	sters are sha		
	2:	The shadow only be acce The shadow	ssed with Pt	JSH.S and P	OP.S.	They may
Words:	1					
Cycles:	1					
Example 1	PUSH.S ; F	Push primary	y register	s into sha	adow regist	ers
	Before		After			
	Instruction		Instruction			
	WO 0000	V	/0 0000			
	W1 1000	V	/1 1000			
	W2 2000	V	/2 2000			
	W3 3000		/3 3000			
	SR 0001 (C	=1) S	R 0001	(C=1)		



PWRSA	V Enter Power Saving Mode
Syntax:	{label:} PWRSAV #lit1
Operands: Operation:	lit1 ∈ [0,1] 0 → WDT count register 0 → WDT prescaler A count 0 → WDT prescaler B count 0 → WDTO (RCON<4>) 0 → SLEEP (RCON<3>) 0 → IDLE (RCON<2>) <u>If (lit1 = 0):</u> Enter SLEEP mode <u>Else:</u> Enter IDLE mode
Status Affected: Encoding: Description:	None1111111001000000000kPlace the processor into the specified Power Saving mode. If lit1 = 0, SLEEP mode is entered. In SLEEP mode, the clock to the CPU and
	peripherals are shutdown. If an on-chip oscillator is being used, it is also shutdown. If lit1 = 1, IDLE mode is entered. In IDLE mode, the clock to the CPU shuts down, but the clock source remains active and the peripherals continue to operate.
	This instruction resets the Watchdog Timer Count register and the Prescaler Count registers. In addition, the WDTO, SLEEP and IDLE flags of the Reset System and Control (RCON) register are reset.
	 Note 1: The processor will exit from IDLE or SLEEP through an interrupt, processor RESET or Watchdog Time-out. See the dsPIC30F Data Sheet for details. 2: If awakened from IDLE mode, IDLE (RCON<2>) is set to '1' and the clock source is applied to the CPU. 3: If awakened from SLEEP mode, SLEEP (RCON<3>) is set to '1' and the clock source is started. 4: If awakened from a Watchdog Time-out, WDTO (RCON<4>) is set to '1'.
Words:	1
Cycles:	1
Example 1	PWRSAV #0 ; Enter SLEEP mode
Example 2	Before After Instruction Instruction SR 0040 (IPL=2) SR PWRSAV #1 ; Enter IDLE mode
	BeforeAfterInstructionInstructionSR0020(IPL=1)SR0020(IPL=1)

RCALL		Relative Ca	ll			
Syntax:	{label:}	RCALL	Expr			
Operands:				bel or express 16, where Sli		32767]
Operation:	. , .) → (TOS) → W15 >) → (TOS)				
Status Affected:	None					
Encoding:	0000	0111	nnnn	nnnn	nnnn	nnnn
Description:	from the cu pushed ont sign-extenc and the res	rrent PC. Be o the stack. <i>I</i> led 17-bit val ult is stored i	fore the call is After the retur ue (2 * Slit16 in the PC.	of 32K program s made, the re n address is s) is added to t	eturn address stacked, the the contents o	(PC+2) is of the PC
		are a signed ords) from (P		ecifies the siz	ze of the relat	ive call (in
		-		ction should b e word of pro		
Words:	1					
Cycles:	2					
01	2004 2006 2458 Task:	RCALL ADD 	_Task1 W0, W1, W2	2	l _Task1 .sk1 subrou	tine
	2450 _105K		NO, NZ, N.	, _10	BAL SUDIOU	cine
	Before			After		
	Instruction	-		nstruction		
P		_	PC	01 2458		
W1			W15	0814		
Data 081	-		Data 0810	2006		
Data 081			Data 0812	0001		
	₹ <u>000</u> 620E 6210	RCALL MOV	SR		l _Init	
	7000 Init		W2		it subrout	ine
	7002	•••		,		
	Before			After		
	Instruction			nstruction		
PC			PC	00 7000		
W1			W15	0C54		
Data 0C5			Data 0C50	6210		
Data 0C5			Data 0C52	0000		
SF	२ 000	U	SR	0000		

RCALL		Compute	ed Relative	Call	
Syntax:	{label:}	RCALL	Wn		
Operands:	Wn ∈ [W0	-			
Operation:	$\begin{array}{l} (PC) + 2 \to \\ (PC < 15:0> \\ (W15) + 2 \to \\ (PC < 22:16 \\ (W15) + 2 \to \\ (PC) + (2 * \\ NOP \to Ins \end{array}$	ightarrow (TOS ightarrow W15 ightarrow (TOS ightarrow W15 (Wn)) ightarrow	S) PC		
Status Affected:	None				
Encoding:	0000	0001	0010	00000 0000 s	SSS
Description:	The range of PC. Before stack. After (2 * (Wn)) is PC. Registe	of the call is the call is the return s added to er direct ac	s 32K progra made, the re address is s the contents dressing mu	specified by the working register V m words forward or back from the o eturn address (PC+2) is pushed on tacked, the sign-extended 17-bit va of the PC and the result is stored ist be used for Wn.	current to the alue
		select the	address of	the source register.	
Words:	1				
Cycles:	2				
Example 1 00FF8 00FF8		INC 	W2, W3	; Destination of RC	CALL
01000 01000 01000	A(RCALL MOVE	W6 W4, [W10	; RCALL with W6]	
	Before			After	
	Instruction			Instruction	
PC	01 000A		PC	00 FF8C	
W6	FFC0		W6	FFCO	
W15 Data 1004	1004		W15	1008	
Data 1004	98FF 2310		Data 1004 Data 1006	000C	
SR	0000		SR	0000	
Example 2 00030 00030)2	RCALL FF1L 	W2 W0, W1	; RCALL with W2	
00045 00045		CLR	W2	; Destination of RC	CALL
PC W2 W15 Data 1004 Data 1006 SR	Before Instruction 00 0302 00A6 1004 32BB 901A 0000		PC W2 W15 Data 1004 Data 1006 SR	After Instruction 00 0450 00A6 1008 0304 0000	

REPEAT		Repeat Ne	xt Instructio	n 'lit14+1' Ti	mes	
Syntax:	{label:}	REPEAT	#lit14			
Operands: Operation:	lit14 ∈ [0 . (lit14) → F (PC)+2 → Enable Co	RCOUNT				
Status Affected:	RA					
Encoding:	0000	1001	00kk	kkkk	kkkk	kkkk
Description:	(lit14 + 1)	times. The re	mmediately f peated instru for all iteratio	ction (or targ	et instructior	n) is held i
	repeat cou with each zero, the t	Int value spece execution of arget instruct execution co	xecutes, the l cified in the in the target ins ion is execute ontinues with	struction. RC truction. Whe ed one more	COUNT is de en RCOUNT time, and th	cremente equals en norma
	The 'k' bits	s are an unsi	gned literal th	at specifies	the loop cou	nt.
	 When the R. The ta an a D inst 	A bit is not se arget REPEAT	teral is '0', RE et. c instruction c at changes p JNK, MOV.D	an NOT be: rogram flow		
	Unex	pected result	s may occur i	f these targe	t instructions	s are used
	Note:	The REPEA	I and target	instruction a	re interruptib	le.
Words:	1					
Cycles:	1					
Example 1 0004			W1, [W2++]		ite ADD 10 or update	times
PC [RCOUNT] SR]	Before Instruction 00 0452 0000 0000 00000 00000	ן ק		After struction 00 0454 0009 0010 (R	A=1)	
Example 2 0008		AT #0x3FF [W6++]		te CLR 102 the scrat		
PC RCOUNT SR	Before Instruction 00 089E 0000 0000 0000 0000	а 		After struction	A=1)	

5

REPEAT	Repeat Next Instruction Wn+1 Times
Syntax:	{label:} REPEAT Wn
Operands:	Wn ∈ [W0 W15]
Operation:	$(Wn<13:0>) \rightarrow RCOUNT$ $(PC)+2 \rightarrow PC$ Enable Code Looping
Status Affected:	RA
Encoding:	0000 1001 1000 0000 0000 ssss
Description:	Repeat the instruction immediately following the REPEAT instruction (Wn<13:0>) times. The instruction to be repeated (or target instruction) is held in the instruction register for all iterations and is only fetched once.
	When this instruction executes, the RCOUNT register is loaded with the lower 14-bits of Wn. RCOUNT is decremented with each execution of the target instruction. When RCOUNT equals zero, the target instruction is executed one more time, and then normal instruction execution continues with the instruction following the target instruction.
	The 's' bits specify the Wn register that contains the repeat count.
	Special Features, Restrictions:
	1. When (Wn) = 0, REPEAT has the effect of a NOP and the RA bit is not set.
	2. The target REPEAT instruction can NOT be:
	 an instruction that changes program flow
	 a DO, DISI, LNK, MOV.D, PWRSAV, REPEAT or ULNK instruction
	a 2-word instruction
	Unexpected results may occur if these target instructions are used.
	Note: The REPEAT and target instruction are interruptible.
Words:	1
Cycles:	1
Example 1 000A	
	Before After nstruction Instruction
PC	00 0A26 PC 00 0A28
W4	0023 W4 0023
RCOUNT	0000 RCOUNT 0023
SR	0000 SR 0010 (RA=1)

Example 2	00089E 0008A0	REPEAT TBLRDL	W10 [W2++], [W3++]		TBLRD (W10+1) times t (0x840)
		Before struction		After Instruction	
	PC C	0 089E	PC	00 08A0	
N	V10	00FF	W10	00FF	
RCOL	JNT	0000	RCOUNT	00FF	
	SR	0000	SR	0010	(RA=1)

RESET		Reset		
Syntax:	{label:}	RESET		
Operands: Operation:	condition.	egisters that are affec (RCON<6>)	ted by a MCLI	\overline{R} Reset to their RESET
Status Affected:		AB, SA, SB, SAB, DA	A, DC, IPL<2:0)>, RA, N, OV, Z, C
Encoding:	1111	1110 0000		0000 0000
Description:	and periph set to 'o', t	eral registers will take he location of the RES	e their power-c	oftware RESET. All core on value. The PC will be truction. The SWR bit, e RESET instruction was
	Note:	Refer to the dsPIC30 power-on value of all		Reference Manual for the
Words:	1			
Cycles:	1			
Example 1 00202	A RESET	; Execute so	oftware RES	ET
	Before		After	
PC	Instruction	PC	Instruction	
wo	8901	WO	0000	
W1	08BB	W1	0000	
W2	B87A	W2	0000	
W3	872F	W3	0000	
W4	C98A	W4	0000	
W5	AAD4	W5	0000	
W6	981E	W6	0000	
W7	1809	W7	0000	
W8	C341	W8	0000	
W9	90F4	W9	0000	
W10	F409	W10	0000	
W11	1700	W11	0000	
W12 W13	1008	W12	0000	
W13 W14	6556	W13 W14	0000	
W14 W15	231D 1704	W14 W15	0000	
SPLIM	1800	SPLIM	0000	
TBLPAG	007F	TBLPAG	0000	
PSVPAG	0001	PSVPAG	0000	
CORCON	00F0	CORCON		(SATDW=1)
RCON	0000	RCON		(SWR=1)
SR		(IPL, C=1) SR	0000	

RETFIE		Return from Interrupt			
Syntax:	{label:}	RETFIE			
Operands: Operation:	(TOS<7>) → (TOS<6:0> (W15)-2 → (TOS<15:0	>) → (SR<7:0>) → (IPL3, CORCON<3>)) → (PC<22:16>)			
Status Affected:		RA, N, OV, Z, C			
Encoding:	0000	0110 0100	0000	0000	0000
Description:	the low byt Significant	n Interrupt Service Routine. e of the Status register, IPL Byte of the PC. The stack i ts of the PC.	<3> (CORC	CON<3>) and	d the Most
Words: Cycles:	1	restores the Interrupt Pric execution was processed. Before RETFIE is execut must be cleared in softwar ption pending)	ed, the ap	propriate int	errupt flag
Example 1 0002	A26 RETFI	E ; Return from ISR			
	Before Instruction	Inst	After ruction		
PC W15	00 0A26	PC 01 W15	0230		
Data 0830	0834	Data 0830	0830		
Data 0832	8101	Data 0832	8101		
CORCON	0001	CORCON	0001		
SR	0000	SR		L=4, C=1)	
Example 2 0080)50 RETFI	E ; Return from ISR			
	Before		fter		
_	Instruction		ruction		
PC	00 8050	PC 00			
W15	0926	W15	0922		
Data 0922	7008	Data 0922	7008		
Data 0924	0300	Data 0924 CORCON	0300		
CORCON SR	0000		0000		
		SR	0003 (Z,	C=1)	

RETLW		Return with	h Literal in	Wn
Syntax:	{label:}	RETLW{.B}	#lit10,	Wn
Operands:	-	. 255] for byta . 1023] for wa W15]	•	
Operation:	(W15)-2 →	PC<22:16>) → W15 PC<15:0>)		
Status Affected:	None			
Encoding:	0000	0101	0Bkk	kkkk kkkk dddd
Description:	in Wn. The signed liter pointer (W	e software sta ral is stored ir 15) is decrem	ck is popp n Wn. Sinc nented by 4	
	The 'k' bits	specify the v	alue of the	eration (0 for word, 1 for byte). e literal. he destination register.
	Note 1:			e instruction denotes a byte operation ration. You may use a .w extension to
	2:	denote a wo For byte oper value [0:255	rd operations, the]. See Sec	on, but it is not required. Iteral must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte
Words:	2 :	denote a wo For byte oper value [0:255 ands" for in	rd operations, the]. See Sec	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper
Words: Cycles:	1	denote a wo For byte oper value [0:255 ands" for in	rd operation rations, the]. See Sec formation of	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper
	1 3 (2 if exce	denote a wo For byte oper value [0:255 ands" for in mode.	rd operations, the rations, the]. See Sec formation of g)	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper
Cycles:	1 3 (2 if exce	denote a wo For byte oper value [0:255 ands " for in mode.	rd operations, the rations, the]. See Sec formation of g)	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte
Cycles: Example 1 0004	1 3 (2 if exce 40 RETLW Before Instruction	denote a wo For byte oper value [0:255 ands " for in mode.	rd operation rations, the]. See Sec formation of g) 0 ; Re	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction
Cycles: Example 1 0004 PC	1 3 (2 if exce 40 RETLW Before Instruction 00 0440	denote a wo For byte oper value [0:255 ands " for in mode.	rd operation rations, the]. See Sec formation of g) 0 ; Re PC [on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006
Cycles: Example 1 0004 PC W0	1 3 (2 if exce 40 RETLW Before Instruction 00 0440 9846	denote a wo For byte oper value [0:255 ands " for in mode.	rd operation rations, the J. See Sec formation of g) 0 ; Re PC W0	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A
Cycles: Example 1 0004 PC W0 W15	1 3 (2 if excell 40 RETLW Before Instruction 00 0440 9846 1988	denote a wo For byte oper value [0:255 ands" for in mode. eption pendin 7.B #0xA, W	rd operation rations, the]. See Sec formation (g) 0 ; Re PC W0 W15	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A 1984
Cycles: Example 1 0004 PC W0 W15 Data 1984	1 3 (2 if exce 40 RETLW Before Instruction 00 0440 9846 1988 7006	denote a wo For byte ope value [0:255 ands" for in mode. eption pendin T.B #0xA, W	rd operation rations, the]. See Sec formation of g) 0 ; Re W0 W15 ata 1984	turn with 0xA in W0 After Instruction 00 7006 980A 1984 7006
Cycles: Example 1 0004 PC W0 W15	1 3 (2 if exce 40 RETLW Before Instruction 00 0440 9846 1988 7006 0000	denote a wo For byte ope value [0:255 ands" for in mode. eption pendin T.B #0xA, W	rd operation rations, the]. See Sec formation of g) 0 ; Re W0 W15 ata 1984 ata 1986	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A 1984
Cycles: Example 1 0004 PC W0 W15 Data 1984 Data 1986	1 3 (2 if excellent 40 RETLW Before Instruction 00 0440 9846 1988 7006 00000	denote a wo For byte oper value [0:255 ands" for in mode. eption pendin 7.B #0xA, W Da Da	rd operation rations, the]. See Sec formation of g) 0 ; Re W0 W15 ata 1984 ata 1986 SR	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A 1984 7006 0000
Cycles: Example 1 0004 PC W0 W15 Data 1984 Data 1986 SR	1 3 (2 if excellent 40 RETLW Before Instruction 00 0440 9846 1988 7006 00000 00A	denote a wo For byte oper value [0:255 ands" for in mode. eption pendin 7.B #0xA, W Da Da	rd operation rations, the]. See Sec formation of g) 0 ; Re W0 W15 ata 1984 ata 1986 SR	turn with 0x230 in W2
Cycles: Example 1 0004 PC W0 W15 Data 1984 Data 1986 SR	1 3 (2 if excell 40 RETLW Before Instruction 00 0440 9846 1988 7006 0000 0000	denote a wo For byte oper value [0:255 ands" for in mode. eption pendin 7.B #0xA, W Da Da	rd operation rations, the]. See Sec formation of g) 0 ; Re W0 W15 ata 1984 ata 1986 SR W2 ; Re	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A 1984 7006 0000 0000
Cycles: Example 1 0004 PC W0 W15 Data 1984 Data 1986 SR	1 3 (2 if excell 40 RETLW Before Instruction 00 0440 9846 1988 7006 0000 000 000 000 88 7006 0000 0000 000 000 000 000 000 000 000 000 000	denote a wo For byte oper value [0:255 ands" for in mode. eption pendin 7.B #0xA, W Da Da	rd operation rations, the]. See Sec formation of g) 0 ; Re W0 W15 ata 1984 ata 1986 SR W2 ; Re	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A 1984 7006 0000 0000 turn with 0x230 in W2 After
Cycles: Example 1 0004 PC W0 W15 Data 1984 Data 1986 SR Example 2 0005	1 3 (2 if excell 40 RETLW Before Instruction 00 0440 9846 1988 7006 0000 00A RETLW Before Instruction	denote a wo For byte oper value [0:255 ands" for in mode. eption pendin 7.B #0xA, W Da Da	rd operation rations, the]. See Sec formation of g) 0 ; Re W0 W15 ata 1984 ata 1986 SR W2 ; Re	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A 1984 7006 0000 0000 turn with 0x230 in W2 After Instruction
Cycles: Example 1 0004 PC W0 W15 Data 1984 Data 1986 SR Example 2 0005	1 3 (2 if excell 40 RETLW Before Instruction 00 0440 9846 1988 7006 0000 00A RETLW Before Instruction 00 050A	denote a wo For byte oper value [0:255 ands" for in mode. eption pendin 7.B #0xA, W Da Da	rd operation rations, the]. See Sec formation of g) 0 ; Re PC W0 W15 ata 1984 ata 1984 SR W2 ; Re	on, but it is not required. literal must be specified as an unsigner ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A 1984 7006 0000 0000 turn with 0x230 in W2 After Instruction 01 7008
Cycles: Example 1 0004 PC W0 W15 Data 1984 Data 1986 SR Example 2 0005 PC W2	1 3 (2 if excell 40 RETLW Before Instruction 00 0440 9846 1988 7006 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 00000 00000	denote a wo For byte oper value [0:255 ands" for in mode. eption pendin 7.B #0xA, W Da Da Da	rd operations, the rations, the]. See Sec formation of g) 0 ; Re PC W0 W15 ata 1984 ata 1984 SR W2 ; Re PC W2	on, but it is not required. literal must be specified as an unsigned ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A 1984 7006 0000 0000 turn with 0x230 in W2 After Instruction 01 7008 0230
Cycles: Example 1 0004 PC W0 W15 Data 1984 Data 1986 SR Example 2 0005 PC W2 W15	1 3 (2 if excell 40 RETLW Before Instruction 00 0440 9846 1988 7006 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 000000 00000000 000000000000 000000000000000000000000000000000000	denote a wo For byte ope value [0:255 ands" for in mode. eption pendin T.B #0xA, W Da Da da t #0x230,	rd operation rations, the]. See Sec formation of g) 0 ; Re W0 W15 ata 1984 ata 1986 SR W2 ; Re W2 ; Re W2 W15 W2	on, but it is not required. Eliteral must be specified as an unsigner ction 4.6 "Using 10-bit Literal Oper on using 10-bit literal operands in Byte turn with 0xA in W0 After Instruction 00 7006 980A 1984 7006 0000 0000 turn with 0x230 in W2 After Instruction 01 7008 0230 11FC

RETURN		Return			
Syntax:	{label:}	RETURN			
Operands:	None				
Operation:	$(W15)-2 \rightarrow (TOS) \rightarrow (FOS)$	PC<22:16>) W15			
Status Affected:	None				
Encoding:	0000	0110 0000	0000	0000	0000
Description:		n subroutine. The sof nee two pops are mac ed by 4.		• •	
Words:	1				
Cycles:	3 (2 if exce	ption pending)			
Example 1 0012	A06 RETUR	N ; Return fr	om subroutin	e	
	Before		After		
PC	Instruction	PC	Instruction		
W15	1248	W15	1244		
Data 1244	0004	Data 1244	0004		
Data 1246	0001	Data 1246	0001		
SR	0000	SR	0000		
Example 2 0054	04 RETUR	N ; Return fr	om subroutin	e	
	D (A 44 a m		
	Before Instruction		After Instruction		
PC		PC			
W15	Instruction 00 5404 090A	W15	Instruction 00 0966 0906		
W15 Data 0906	Instruction 00 5404 090A 0966	W15 Data 0906	Instruction 00 0966 0906 0966		
W15	Instruction 00 5404 090A	W15	Instruction 00 0966 0906		

RLC	Rotate Left f through Carry							
Syntax:	{label:}	RLC{.B}	f	{,WREG}				
Operands:	f ∈ [0	8191]						
Operation:	$(C) \rightarrow (f < 6:0 > (f < 7 >))$ $(F < 7 >)$ $For word$ $(C) \rightarrow$	<u>operation:</u> Dest<0>)>) → Dest<1						
	-C	∢						
Status Affected:	N, Z, C							
Encoding:	1101	0110	1BDf	ffff ffff ff	ff			
Description:	Carry flag of the Sta	g and place th itus Register	ne result in is shifted in	pister f one bit to the left through the the destination register. The Carry nto the Least Significant bit of the tten with the Most Significant bit of	fla			
	WREG is	specified, th	e result is s	ermines the destination register. If stored in WREG. If WREG is not ne file register.				
	The 'D' b	it selects the	destination	peration (0 for word, 1 for byte). (0 for f, 1 for WREG). he file register.				
		rather than denote a w	a word ope ord operati	he instruction denotes a byte opera eration. You may use a .w extension ion, but it is not required. vorking register W0.				
Words:	1							
Cycles:	1							
Example 1 RL	C.B 0x123	3 ; Ro	tate Left	w/ C (0x1233) (Byte mode)				
Data 1232 SF		Data 123 S		7				
Example 2 RL	C 0x820,			z w/ C (0x820) (Word mode) Lt in WREG				
WREG (W0 Data 0820 SF) 216E	WREG (W Data 082 C=1) S	20 216E					

RLC	Rotate Left Ws through Carry							
Syntax:	{label:}	RLC{.B}	Ws,	Wd				
			[Ws],	[Wd]				
			[Ws++],	[Wd++]				
			[Ws],	[Wd]				
			[++Ws],	[++Wd]				
			[Ws],	[Wd]				
Operands:	Ws ∈ [W0 Wd ∈ [W0	-						
Operation:	(Ws<7∷ <u>For word</u> (C) → \ (Ws<14	Wd<0> :0>) → Wd<7 >) → C <u>operation:</u>						
Status Affected:	N, Z, C							
Status Affected: Encoding:	N, Z, C	0010	1Bqq	qddd	dppp	SSSS		
Encoding:	1101 Rotate the the Carry Carry flag Wd, and i register d The 'B' bi The 'q' bit The 'q' bit	e contents of flag and plac of the Status t is then over irect or indire t selects byte ts select the	the source r ce the result s register is s rwritten with ect addressin e or word ope destination A address of th	register Ws on in the destinat shifted into the the Most Signi og may be used eration (0 for w oddress mode. ne destination i	e bit to the le ion register V Least Signifi ficant bit of V d for Ws and vord, 1 for by	ft through Vd. The icant bit c Vs. Eithei Wd.		
	1101 Rotate the the Carry Carry flag Wd, and i register d The 'B' bi The 'q' bit The 'd' bit The 'p' bit	e contents of flag and place of the Status t is then over irect or indirect t selects byte ts select the ts select the ts select the	the source r ce the result s register is s rwritten with ect addressin destination A address of th source Addre	register Ws on in the destinat shifted into the the Most Signi og may be used eration (0 for w oddress mode. ne destination i	e bit to the le ion register V Least Signifi ficant bit of V d for Ws and vord, 1 for by register.	ft through Vd. The icant bit c Vs. Eithe Wd.		
Encoding:	1101 Rotate the the Carry Carry flag Wd, and i register d The 'B' bi The 'q' bit The 'd' bit The 'p' bit	e contents of flag and place of the Status t is then over irect or indirect t selects byte ts select the ts select the ts select the ts select the ts select the ts select the ts select the the select the select the the select the select the the select the select the the select the select the select the the select the select the select the the select the select the select the select the select the the select the s	the source r ce the result s register is s rwritten with ect addressin destination A address of th source Addre address of th source Addre address of th sion . B in the a word oper	register Ws on in the destinat shifted into the the Most Signi og may be used eration (0 for w address mode. he destination n ess mode. he source regis e instruction de ation. You may	e bit to the le ion register V Least Signifi ficant bit of V d for Ws and vord, 1 for by register. enotes a byte / use a . W ez	ft through Vd. The icant bit c Vs. Either Wd. te).		
Encoding: Description:	1101 Rotate the the Carry Carry flag Wd, and i register d The 'B' bi The 'G' bit The 'c' bit The 's' bit	e contents of flag and place of the Status t is then over irect or indirect t selects byte ts select the ts select the ts select the ts select the ts select the ts select the ts select the the select the select the the select the select the the select the select the the select the select the select the the select the select the select the the select the select the select the select the select the the select the s	the source r ce the result s register is s rwritten with ect addressin destination A address of th source Addre address of th source Addre address of th sion . B in the a word oper	register Ws on in the destinat shifted into the the Most Signi og may be used eration (0 for w address mode. he destination n ess mode. he source regis e instruction de	e bit to the le ion register V Least Signifi ficant bit of V d for Ws and vord, 1 for by register. enotes a byte / use a . W ez	ft through Vd. The icant bit c Vs. Either Wd. te).		
Encoding:	1101 Rotate the the Carry Carry flag Wd, and i register d The 'B' bi The 'G' bi The 'q' bi The 'p' bi The 's' bi Note:	e contents of flag and place of the Status t is then over irect or indirect t selects byte ts select the ts select the ts select the ts select the ts select the ts select the ts select the the select the select the the select the select the the select the select the the select the select the select the the select the select the select the the select the select the select the select the select the the select the s	the source r ce the result s register is s rwritten with ect addressin destination A address of th source Addre address of th source Addre address of th sion . B in the a word oper	register Ws on in the destinat shifted into the the Most Signi og may be used eration (0 for w address mode. he destination n ess mode. he source regis e instruction de ation. You may	e bit to the le ion register V Least Signifi ficant bit of V d for Ws and vord, 1 for by register. enotes a byte / use a . W ez	ft through Vd. The icant bit c Vs. Eithe Wd. te).		
Encoding: Description: Words: Cycles:	1101 Rotate the the Carry Carry flag Wd, and i register d The 'B' bi The 'G' bi The 'G' bi The 'g' bi The 's' bit Note:	e contents of flag and plac of the Status t is then over irect or indirect t selects byte ts select the ts select the ts select the ts select the ts select the ts select the the extens rather than denote a w	the source r ce the result s register is s rwritten with ect addressin e or word oper destination A address of th source Addre address of th source Addre address of th sion .B in the a word oper rord operatio	register Ws on in the destinat shifted into the the Most Signi og may be used eration (0 for w address mode. he destination n ess mode. he source regis e instruction de ation. You may	e bit to the le ion register V Least Signifi ficant bit of V d for Ws and vord, 1 for by register. enotes a byte / use a .we: required.	ft through Vd. The icant bit c Vs. Eithe Wd. te).		
Encoding: Description: Words: Cycles:	1101Rotate the the Carry Carry flag Wd, and i register dThe 'B' bi The 'q' bi The 'q' bi The 'g' bit The 's' bitNote:111LC.BW0, WBefore	e contents of flag and plac of the Status t is then over irect or indirect t selects byte ts select the ts select the ts select the ts select the ts select the ts select the the extens rather than denote a w	the source r ce the result s register is s rwritten with ect addressin e or word ope destination A address of th source Addre address of th sion . B in the a word oper rord operatio	register Ws on in the destinat shifted into the the Most Signi og may be used eration (0 for w address mode. the destination the ess mode. the source regist the source regist the instruction de ation. You may n, but it is not	e bit to the le ion register V Least Signifi ficant bit of V d for Ws and vord, 1 for by register. enotes a byte / use a .we: required.	ft through Vd. The icant bit c Vs. Eithe Wd. te).		
Encoding: Description: Words: Cycles: Example 1 R	1101Rotate the the Carry Carry flag Wd, and i register dThe 'B' bi The 'G' bi The 'G' bi The 'S' bitNote:11LC.BW0, W	e contents of flag and place of the Status t is then over irect or indirect t selects byte ts select the ts select the ts select the ts select the ts select the tas select tas select the tas select tas select the tas select tas select tas select tas tas select tas select tas tas select tas select tas select tas tas select tas select tas select tas select tas tas select tas se	the source r ce the result s register is s written with ect addressin e or word oper destination A address of th source Addre address of th sion . B in the a word oper rord operatio	register Ws on in the destinat shifted into the the Most Signi og may be used eration (0 for w address mode. the destination the ess mode. the source regist the source regist the instruction de ation. You may n, but it is not	e bit to the le ion register V Least Signifi ficant bit of V d for Ws and vord, 1 for by register. enotes a byte / use a .we: required.	ft through Vd. The icant bit c Vs. Eithe Wd. te).		

SR

0009 (N=1)

0001 (C=1)

SR

Example 2 RLC $\mbox{[W2++]},\mbox{[W8]}$; Rotate Left w/ C $\mbox{[W2]}$ (Word mode) ; Post-increment W2 ; Store result in [W8] Before After Instruction Instruction W2 2008 W2 200A W8 094E W8 094E Data 094E 3689 Data 094E 8082 Data 2008 C041 Data 2008 C041 SR SR 0001 (C=1) 0009 (N, C=1)

RLNC	Rotate Left f without Carry						
Syntax:	{label:}	RLNC{.B}	f	{,WREG}			
Operands:	f ∈ [0 8	191]					
Operation:	(f<7>) - <u>For word</u> (f<14:0	peration:) → Dest<7:1 → Dest<0> operation: >) → Dest<1) → Dest<0>					
Status Affected:	N, Z						
Encoding:	1101	0110	OBDf	ffff	ffff	ffff	
Description:	result in th	e contents of the destination Significant bi	register. The	e Most Signifi	cant bit of f is	s stored ir	
	WREG is	nal WREG op specified, the the result is s	e result is sto	red in WREG			
	The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 'D' bit selects the destination (0 for WREG, 1 for file register). The 'f' bits select the address of the file register.						
			a word opera ord operation	ition. You ma , but it is not	y use a . w ex required.		
Words:	1						
Cycles:	1						
Example 1 RL	NC.B 0x123	33 ; Ro	otate Left	(0x1233) (Byte mode)		
Data 1232 SF		Data 123 SF		√ =1)			
Example 2 RL	NC 0x820,	-	otate Left core result	(0x820) (W in WREG	ord mode)		
WREG (W0 Data 0820 SF	D 216E	WREG (W0 Data 082 c=1) SF	0 216E	C=0)			

RLNC		Rotate Left Ws without Carry						
Syntax:	{label:}	RLNC{.B}	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]				
Operands:	Ws ∈ [W0 Wd ∈ [W0							
Operation:	(Ws<7> <u>For word (</u> (Ws<14	$(0>) \rightarrow Wd<7$ $(\cdot) \rightarrow Wd<0>$	15:1>					
Status Affected:	Ч N, Z	₹						
Encoding:	1101	0010	0Bqq	qddd	dppp	SSSS		
Description:	place the Ws is stor	esult in the d ed in the Lea Either registe	estination regist Significan	egister Ws on gister Wd. The t bit of Wd, ar lirect address	e Most Signif	icant bit o [.] flag is not		
	The 'q' bit The 'd' bit The 'p' bit	s select the c s select the a s select the s	lestination Ad address of the source Addre	ration (0 for b ddress mode. e destination i ss mode. e source regis	register.	rd).		
	Note:	rather than	a word opera	instruction de ation. You mag	y use a .we			
Words:	1							
Cycles:	1							
Example 1	RLNC.B W0, W		ate Left bre the rea	(WO) (Byte sult in W3	mode)			
	Before Instruction W0 9976 W3 5879 SR 0001 (C	W W =1) SF	3 58EC	N, C=1)				

Example 2	RLNC	[W2++],	[W8]	; Po	tate Le st-incr pre res	eme	ent W2	2	mode)
	Before				After				
	Inst	ruction	Instruction						
,	W2	2008		W2	200	A			
,	W8	094E		W8	094	Е			
Data 09	94E	3689	Data	a 094E	808	3			
Data 20	800	C041	Data	a 2008	C04	1			
	SR	0001 (C	=1)	SR	000	9 (N	I, C=1)	

RRC	Rotate Right f through Carry						
Syntax:	{label:}	RRC{.B}	f	{,WREG}			
Operands:	f ∈ [0 8	8191]					
Operation:	(C) → (f<7:1> (f<0>) <u>For word</u> (C) →	<u>operation:</u> Dest<15> >) → Dest<1					
Status Affected:	N, Z, C						
Encoding:	1101	0111	1B	BDf ffff ffff	ffff		
Description:	Carry flag of the Sta	and place th tus Register	ne result is shifteo	register f one bit to the right throug t in the destination register. The Ca ed into the Most Significant bit of the written with the Least Significant b	arry flag ne		
	WREG is	specified, th	e result i	determines the destination registe is stored in WREG. If WREG is no in the file register.			
	The 'D' bi	t selects the	destinat	d operation (0 for byte, 1 for word) tion (0 for WREG, 1 for file registe of the file register.			
		rather than denote a w	a word ord ope	in the instruction denotes a byte o operation. You may use a .w exte eration, but it is not required. to working register W0.			
Words:	1						
Cycles:	1						
Example 1 RR	C.B 0x123	3 ; Ro	tate R:	ight w/ C (0x1233) (Byte mo	ode)		
Data 123 SI	۲ 0000		R 00	ction 407 000			
Example 2 RR	C 0x820,			ight w/ C (0x820) (Word mod sult in WREG	le)		
WREG (W0 Data 082 SI	0 216E	WREG (Data (C=1)	Insti (W0))820	After truction 90B7 216E 0008 (N=1)			

RRC	Rotate Right Ws through Carry							
Syntax:	{label:}	RRC{.B}	Ws,	Wd				
			[Ws],	[Wd]				
			[Ws++],	[Wd++]				
			[Ws],	[Wd]				
			[++Ws],	[++Wd]				
			[Ws],	[Wd]				
Operands:	Ws ∈ [W0 Wd ∈ [W0							
Operation:	(C) → (Ws<7 (Ws<0 <u>For word</u>	pperation: Wd<7> :1>) → Wd<6 >) → C operation: Wd<15>	3:0>					
	(Ws<1	5:1>) → Wd< >) → C	<14:0>					
	>	- > C-						
Status Affected:	N, Z, C							
Encoding:	1101	0011	1Bqq	qddd	dppp	SSSS		
Description:	the Carry Carry flag Wd, and i	flag and plac of the Status t is then over	ce the result s Register is written with	egister Ws one in the destinat shifted into the the Least Sign ig may be use	ion register Most Signi ificant bit of	Wd. The ficant bit of Ws. Either		
	The 'q' bi The 'd' bi The 'p' bi	ts select the ts select the ts select the	destination A address of th source Addre	eration (0 for w Address mode. ne destination ess mode. ne source regis	register.	∕te).		
	Note:	rather than	a word ope	e instruction de ration. You ma n, but it is not	yusea.we			
Words:	1							
Cycles:	1							
Example 1	RRC.B W0, W		-	w/ C (WO) esult in W3	(Byte mod	e)		
	Before		After					
	Instruction		Instruction					
	Instruction W0 9976 W3 5879							

Example 2 RRC [W2++], [W8] ; Rotate Right w/ C [W2] (Word mode) ; Post-increment W2 ; Store result in [W8] Before After Instruction Instruction W2 2008 W2 200A W8 W8 094E 094E Data 094E Data 094E E020 3689 Data 2008 C041 Data 2008 C041 SR 0001 (C=1) SR 0009 (N, C=1)

RRNC	Rotate Right f without Carry						
Syntax:	{label:}	RRNC{.B}	f	{,WREG}			
Operands:	f ∈ [0 8	191]					
Operation:	(f<0>)	$\rightarrow \text{Dest}^{<}6:0$ $\rightarrow \text{Dest}^{<}7>$	>				
	(f<15:1	<u>operation:</u> >) → Dest<14 → Dest<15>	l:0>				
	►						
Status Affected:	N, Z						
Encoding:	1101	0111	OBDf	ffff	ffff	ffff	
Description:	result in th	e contents of t ne destination Significant bit	register. The	e Least Signif	icant bit of f is	s stored in	
	WREG is	nal WREG op specified, the the result is s	result is stor	ed in WREG	-		
	The 'D' bi	t selects byte t selects the d s select the ac	lestination (0	for WREG, 1			
			a word opera ord operation	tion. You mag , but it is not	y use a . w ex required.		
Words:	1						
Cycles:	1						
Example 1 RRN	C.B 0x12	33 ; Ro	tate Right	(0x1233)	(Byte mode	e)	
	Before Instruction		After Instruction				
Data 1232 SR		Data 1232 SF	2 7407				
Example 2 RRN	C 0x820,	WREG ; Ro ; St	tate Right ore result		Word mode)		
WREG (W0)	Before Instruction	WREG (V	After Instruction	n			

RRNC		Rotate Right Ws without Carry						
Syntax:	{label:}	RRNC{.B}	Ws,	Wd				
			[Ws],	[Wd]				
			[Ws++],	[Wd++]				
			[Ws],	[Wd]				
			[++Ws],	[++Wd]				
			[Ws],	[Wd]				
Operands:	Ws ∈ [W0 Wd ∈ [W0							
Operation:	(Ws<0> For word (Ws<15)	$(1>) \rightarrow Wd<6$ $(2>) \rightarrow Wd<7>$	14:0>					
Status Affected:	N, Z							
Encoding:	1101	0011	0Bqq	qddd	dppp	SSSS		
Description:	place the of Ws is s	result in the c tored in the M	lestination re lost Significa	egister Ws on egister Wd. Th nt bit of Wd, a irect addressi	ne Least Sigr and the Carry	ificant bit flag is not		
	The 'q' bit The 'd' bit The 'p' bit	s select the d s select the a s select the s	lestination A ddress of the ource Addre	ration (0 for w ddress mode. e destination ss mode. e source regis	register.	te).		
	Note:	rather than	a word opera	instruction de ation. You ma a, but it is not	yusea.we			
Words:	1		·	,	I			
Cycles:	1							
Example 1	RRNC.B W0, W			t (WO) (Byt esult in W3				
	Before		After					
		147	Instruction					
	W0 9976 W3 5879	W						
	VVJ 20/9	vv	- 203B					

Example 2	RRNC	[W2++],		; Po	tate Ri st-incr ore res	emen	t W2	mode)
	В	efore			After			
	Ins	truction		I	nstructior	ו		
	W2	2008		W2	200A			
	W8	094E		W8	094E			
Data ()94E	3689	Data 0)94E	E020			

Data 2008

SR

C041

0008 (N=1)

Data 2008

SR

C041

0000

-5	-5	
-		

SAC		Store Acc	cumulator			
Syntax:	{label:}	SAC	Acc,	{#Slit4,}	Wd	
					[Wd]	
					[Wd++]	
					[Wd]	
					[Wd]	
					[++Wd]	
					[Wd+Wb]	
Operands:	Acc ∈ [A,I Slit4 ∈ [-8 Wb, Wd ∈	-	5]			
Operation:	Shift _{Slit4} (A (Acc[31:1)	Acc) (optionation) \rightarrow Wd	al)			
Status Affected:	None					
Encoding:	1100	1100	Awww	wrrr	rhhh	dddd
	The 'A' bit The 'w' bit The 'r' bits The 'h' bit	specifies the s specify the s encode the s select the	may be used the source accu e offset regist e optional acc destination A e destination r	umulator. er Wb. umulator pre ddress mod		
	2:	This instru instruction accumulat If Data Wri 1), the val	iction does no iction stores th SAC.R may b or contents. ite saturation is ue stored to V hift is performe	ne truncated be used to s s enabled (S Vd is subjec	contents of A tore the round	Acc. The Jed CON<5>, =
Words:	2:	This instru instruction accumulat If Data Wri 1), the val	Iction stores the SAC.R may be concerned as the saturation is used to V and the saturation is the saturation is the saturation to V and the stored	ne truncated be used to s s enabled (S Vd is subjec	contents of A tore the round	Acc. The Jed CON<5>, =
Words: Cycles:	2: 3:	This instru instruction accumulat If Data Wri 1), the val	Iction stores the SAC.R may be concerned as the saturation is used to V and the saturation is the saturation is the saturation to V and the stored	ne truncated be used to s s enabled (S Vd is subjec	contents of A tore the round	Acc. The Jed CON<5>, =
Cycles: Example 1 SAC ; R ; S	2: 3:	This instruction accumulat If Data Wri 1), the val optional st ACCA by Lt to W5	Action stores the SAC . R may be sore contents. The saturation is use stored to V hift is performed to V	ne truncated be used to s s enabled (S Vd is subjec	contents of A tore the round	Acc. The Jed CON<5>, =
Cycles: Example 1 SAC ; R ; S	2: 3: 1 1 A, #4, W ight shift tore resul ORCON = 02 Before	This instruction accumulat If Data Wri 1), the val optional sl x5 ACCA by Lt to W5 c0010 (SAT	Action stores the SAC . R may be sore contents. The saturation is use stored to V hift is performed to V	he truncated be used to s s enabled (S Vd is subjec ed. After	contents of A tore the round ATDW, COR t to saturation	Acc. The Jed CON<5>, =
Cycles: Example 1 SAC ; R ; S ; C	2: 3: 1 A, #4, W ight shift tore resul ORCON = 02	This instruction accumulat If Data Wri 1), the val optional sl ACCA by Lt to W5 0010 (SAT	A SAC . R may b SAC . R may b for contents. Ite saturation is us stored to V nift is performed A TDW = 1)	ne truncated be used to s s enabled (S Vd is subjec ed.	ontents of A tore the round ATDW, COR t to saturation	Acc. The Jed CON<5>, =
Cycles: Example 1 SAC ; R ; S ; C W5	2: 3: 1 1 A, #4, W ight shift tore resul ORCON = 03 Before Instruct	This instruction accumulat If Data Wri 1), the val optional sl x5 ACCA by Lt to W5 c0010 (SAT e ion B900	4 4 4 W5	he truncated be used to s s enabled (S Vd is subjec ed. After Instruct	A contents of A tore the round of ATDW, CORO to saturation	Acc. The Jed CON<5>, =
Cycles: Example 1 SAC ; R ; S ; C	2: 3: 1 1 A, #4, W ight shift tore resul ORCON = 03 Before Instruct	This instruction accumulat If Data Wri 1), the val optional sl ACCA by Lt to W5 0010 (SAT	A SAC . R may b SAC . R may b for contents. Ite saturation is us stored to V nift is performed A TDW = 1)	he truncated be used to s s enabled (S Vd is subjec ed. After	ontents of A tore the round ATDW, COR t to saturation	Acc. The ded CON<5>, =

Example 2 SAC B, #-4, [W5++]
; Left shift ACCB by 4
; Store result to [W5], Post-increment W5
; CORCON = 0x0010 (SATDW = 1)

	Befor Instruct	-		I	Aftei nstruct	
W5		2000	W5			2002
ACCB	FF C891	8F4C	ACCB	FF	C891	1F4C
Data 2000		5BBE	Data 2000			8000
CORCON		0010	CORCON			0010
SR		0000	SR			0000

SAC.R		Store Rou	unded Accum	nulator		
Syntax:	{label:}	SAC.R	Acc,	{#Slit4,}	Wd	
					[Wd]	
					[Wd++]	
					[Wd]	
					[Wd]	
					[++Wd]	
					[Wd+Wb]	
Operands:	Acc ∈ [A,I Slit4 ∈ [-8 Wb ∈ [W0 Wd ∈ [W0	+7]) W15]				
Operation:	Shift _{Slit4} (A Round(Ac (Acc[31:10	,	al)			
Status Affected:	None	-				
Encoding:	1100	1101	Awww	wrrr	rhhh	dddd
	positive o (Conventi Either reg The 'A' bit The 'w' bit The 'r' bits The 'h' bit	perand indic onal or Com- ister direct of specifies th ts specify the s encode the s select the s specify the	tive operand ir cates an arithm vergent) is set or indirect add ne source accu e offset registe e optional accu destination Ac e destination r	netic right shi t by the RND ressing may umulator. er Wb. umulator pre- ddress mode egister Wd.	ft. The Round bit, CORCOI be used for V shift.	ding mode N<1>.
	Note 1:		iction does no	t modify the d	contents of th	•
		instruction accumulat If Data Wri = 1), the v	Inction stores the SAC may be a tor contents. ite saturation i alue stored to hift is performe	used to store is enabled (S Wd is subjec	the truncated	c. The d CON<5>,
Words:		instruction accumulat If Data Wri = 1), the v	SAC may be i for contents. ite saturation i ralue stored to	used to store is enabled (S Wd is subjec	the truncated	c. The d CON<5>,
Words: Cycles:	3:	instruction accumulat If Data Wri = 1), the v	SAC may be i for contents. ite saturation i ralue stored to	used to store is enabled (S Wd is subjec	the truncated	c. The d CON<5>,
Cycles: Example 1 SAC ; R ; S	3 :	instruction accumulat If Data Wri = 1), the v optional sh w5 ACCA by ded result	A SAC may be to cor contents. ite saturation i alue stored to hift is performe	used to store is enabled (S Wd is subjec	the truncated	c. The d CON<5>,
Cycles: Example 1 SAC ; R ; S	3: 1 1 C.R A, #4, Right shift Store round	instruction accumulat If Data Wri = 1), the v optional sh ACCA by ded result	A SAC may be to cor contents. ite saturation i alue stored to hift is performe	used to store is enabled (S Wd is subjec	the truncated	c. The d CON<5>,
Cycles: Example 1 SAC ; R ; S ; C	3: 1 1 S.R A, #4, Sight shift Store round CORCON = 03 Before Instruct	instruction accumulat If Data Wri = 1), the v optional sh & W5 = ACCA by ded result coolo (SAT	A SAC may be it for contents. ite saturation it salue stored to nift is performed to W5 TDW = 1)	used to store is enabled (S Wd is subjected.	the truncated ATDW, COR to saturatio	c. The d CON<5>,
Cycles: Example 1 SAC ; R ; S ; C W5	3: 1 1 C.R A, #4, Right shift Store round CORCON = 02 Before Instructi	instruction accumulat If Data Wri = 1), the v optional sh ACCA by ded result coold (SAT e ion B900	A SAC may be to cor contents. ite saturation i value stored to nift is performe to W5 TDW = 1) W5	used to store is enabled (S Wd is subject ed. After Instructio	on	c. The d CON<5>,
Cycles: Example 1 SAC ; R ; S ; C	3: 1 1 S.R A, #4, Sight shift Store round CORCON = 03 Before Instruct	instruction accumulat If Data Wri = 1), the v optional sh ACCA by ded result coold (SAT e ion B900	A SAC may be it for contents. ite saturation it salue stored to nift is performed to W5 TDW = 1)	After Instruction	on	c. The d CON<5>,

Example 2	SAC.R	в, #-4,	[W5++]
	• Left	ghift Z	CCB by 4

; Left shift ACCB by 4 ; Store rounded result to [W5], Post-increment W5

; CORCON = 0×0010 (SATDW = 1)

		Befor struct	-		I	After nstruct	
W5			2000	W5			2002
ACCB	FF 1	F891	8F4C	ACCB	FF	F891	8F4C
Data 2000			5BBE	Data 2000			8919
CORCON			0010	CORCON			0010
SR			0000	SR			0000

SE		Sign-Exter	nd Ws			
Syntax:	{label:}	SE	Ws,	Wnd		
			[Ws],			
			[Ws++],			
			[Ws],			
			[++Ws],			
			[Ws],			
Operands:	Ws ∈ [W0 Wnd ∈ [W	-				
Operation:	-	→ Wnd<7:0>				
•	<u>lf (Ws<7></u>					
	<u>Else:</u> 0 → Wn	d<15:8>				
Status Affected:	N, Z, C					
Encoding:	1111	1011	0000	0ddd	dppp	SSSS
	direct add	ressing must nt of the N fl	-	Nnd. The C f	lag is set to t	-
	The 'p' bits	s select the s	ddress of the ource Addres ddress of the	s mode.		
		.wextensic The source	on converts a n. Ws is addres dification is b	sed as a byte		
Words:	1			y <u> </u> .		
Cycles:	1					
Example 1 SE	W3, W4 ;	Sign-exte	end W3 and	store to W	4	
	Before		After			
W	Instruction 7839	W	Instruction 3 7839			
W2		W				
SF		SI		2=1)		
Example 2 SE	[W2++], W12		extend [W2] ncrement W		to W12	
	Before Instruction		After Instruction			
	manucuon					
10/2	0900	\٨/	2 0901			
W2 W12		W W1				
W2 W12 Data 0900	2 1002	W W1 Data 090	2 FF8F			

SETM		Set f or WF	REG			
Syntax:	{label:}	SETM{.B}	f			
			WREG			
Operands:	f ∈ [0 81	191]				
Operation:	For word c	→ destination operation:	designated b	-		
Status Affected:	None		J			
Encoding:	1110	1111	1BDf	ffff	ffff	ffff
Description:			ied register a et. Otherwise,			
	The 'D' bit	selects the c	or word opera lestination (0 ldress of the	for WREG, 1		
		rather than a denote a wo	on . B in the in a word operat ord operation, is set to work	ion. You may but it is not	vuse a .wex required.	
Words:	1					
Cycles:	1					
Example 1 SETM.	B 0x891	; Set Ox	891 (Byte	mode)		
ا Data 0890 SR	Before nstruction 2739 0000	Data 089 SF				
Example 2 SETM	WREG	; Set WR	EG (Word m	ode)		
l WREG (W0) SR	Before nstruction	WREG (W0 SF	·			

SETM		Set Ws				
Syntax:	{label:}	SETM{.B}	Wd			
			[Wd]			
			[Wd++]			
			[Wd]			
			[++Wd]			
			[Wd]			
Operands:	Wd ∈ [W	0 W15]				
Operation:	For byte					
		→ Wd for byte	operation			
		operation: $F \rightarrow Wd$ for w	ord operation	l		
Status Affected:	None					
Encoding:	1110	1011	1Bqq	qddd	d000	0000
Description:		s of the specif			Either registe	er direct or
	indirect a	ddressing ma	y be used for	Wd.		
	The 'q' bi	ts selects byte ts select the d ts select the a	estination Ad	dress mode.		yte).
	Note:	rather than	on . B in the a word opera ord operation	tion. You ma	yusea.we	
Words:	1		•			
Cycles:	1					
Example 1 SETM	.B W13	; Set W1	.3 (Byte mo	de)		
	Before		After			
	Instruction		Instruction			
W1:		W1				
SI	₹ 0000	SI	₹ 0000			
Example 2 SETM	[W6] ; Pre-de ; Set [W		(Word mod	le)	
	Before Instruction		After Instruction			
W		W				
Data 124	3CD9	Data 124	FFFF			
SI	۲ 0000	SI	۲ 0000			

SFTAC		Arithmetic	Shift Accun	nulator by SI	110	
Syntax:	{label:}	SFTAC	Acc,	#Slit6		
Operands:	Acc ∈ [A,B] Slit6 ∈ [-16					
Operation:	Shift _k (Acc)	\rightarrow Acc				
Status Affected:	OA, OB, O	AB, SA, SB,	SAB			
Encoding:	1100	1000	A000	0000	01kk	kkkk
Description:	signed, 6-b shift range a positive o	it literal and is -16:16, w	store the res here a negati cates a right s	f the specified oult back into t ive operand ir shift. Any bits	he accumulandicates a lef	ator. The ft shift and
			ccumulator f	or the result. f bits to be shi	ifted.	
	2:	CORCON< the accumu If the shift modification	6> or SATB, lator is subje amount is g	or the target a CORCON<7> ct to saturatio reater than 1 ade to the), the value n. 6 or less th	stored to an -16, n
Words:	1					
Cycles:	1					
; St	AC A, #12 rithmetic r tore result ORCON = 0x0	to ACCA		12		
	Before			After		
	Instructio	n	F	Instruction	<u>ו</u>	
ACCA	00 120F F		ACCA	00 0001 2	OFF	
CORCON	C	080				
			CORCON		080	
SR Example 2 SFTF ; Ar ; St		eft shift to ACCB	SR	0	080	
SR Example 2 SFTF ; Ar ; St	AC B, #-10 rithmetic l core result DRCON = 0x0	eft shift to ACCB	SR	0		
SR Example 2 SFTF ; Ar ; St	AC B, #-10 cithmetic 1 core result	eft shift to ACCB 0040 (SATB	SR	0		
SR Example 2 SFTF ; Ar ; St	AC B, #-10 rithmetic l core result DRCON = 0x0 Before	n	SR ACCB by 1 = 1)	0 0 After	000	
SR Example 2 SFTZ ; AI ; St ; CC	AC B, #-10 rithmetic l core result DRCON = 0x0 Before Instruction FF FFF1 8	n	SR ACCB by 1 = 1)	0 After Instruction	000	

SFTAC		Anumeu	o onne Accou	nulator by W	D I	
Syntax:	{label:}	SFTAC	Acc,	Wb		
Operands:	$Acc \in [A,E]$ $Wb \in [W0]$					
Operation:	Shift _(Wb) (A	Acc) \rightarrow Acc				
Status Affected:	OA, OB, C	AB, SA, SE	8, SAB			
Encoding:	1100	1000	A000	0000	0000	SSSS
Description:	store the re Wb are us where a ne	esult back ir ed to specif egative valu	nto the accum y the shift am e indicates a	of the specifie ulator. The Le ount. The shif left shift and a h are shifted o	east Significa ft range is -1 a positive val	nt 6 bits o 6:16, ue
				or the source shift count re		
		CORCON- the accum If the shift modification	<6> or SATB, ulator is subje amount is g n will be m	or the target a CORCON<7 oct to saturation reater than 1 nade to the	>), the value on. 6 or less th	stored to an -16, r
		arithmetic	trap will occu	:		
Words:	1	arithmetic	trap will occur			
Cycles:	1	arithmetic	trap will occur			
; St	1 AC A, W0 rithmetic tore resul DRCON = 0x Before Instructio 00 320F	shift ACC t to ACCA 0000 (sat on FFFC	А Ъу (W0)	Sabled) After Instructio F 03 20FA B 0	FFC 090 000	0AB=1)
Cycles: Example 1 SFTZ ; A ; St ; CO WO ACCA CORCON SR Example 2 SFTZ ; A ; St	1 AC A, W0 rithmetic tore resul ORCON = 0x Before Instructio 00 320F 00 320F Core resul ORCON = 0x	shift ACC t to ACCA 0000 (sat con FFFC AB09 0000 0000 0000 shift ACC t to ACCB 0040 (SAT	A by (W0) uration dia W0 ACCA CORCON SR B by (W12)	sabled) Instructio 03 20FA B 0 8	FFC 090 000	0AB=1)
Cycles: Example 1 SFT2 ; A; ; Sf ; Co W0 ACCA CORCON SR Example 2 SFT2 ; Ar ; St	1 AC A, W0 rithmetic tore resul ORCON = 0x Before Instructio 00 320F 00 320F AC B, W12 rithmetic core resul DRCON = 0x Before	shift ACC t to ACCA 0000 (sat 000 FFFC AB09 0000 0000 0000 shift ACC t to ACCB 0040 (SAT	A by (W0) uration dia W0 ACCA CORCON SR B by (W12)	Sabled) Instructio 03 20FA B 0 8	FFC 090 000 800 (OA, C)AB=1)
Cycles: Example 1 SFT2 ; A; ; Sf ; Co W0 ACCA CORCON SR Example 2 SFT2 ; Ar ; St	1 AC A, W0 rithmetic tore resul DRCON = 0x Before Instructio 00 320F 00 320F 00 320F Core resul DRCON = 0x Before Instructio	shift ACC t to ACCA 0000 (sat on FFFC AB09 0000 0000 0000 shift ACC t to ACCB 0040 (SAT	A by (W0) uration dia W0 ACCA CORCON SR B by (W12) B = 1)	After Instructio 03 20FA B 0 8 After Instruction	FFC 090 800 (OA, C)AB=1)
Cycles: Example 1 SFTZ ; A: ; St ; CO W0 ACCA CORCON SR Example 2 SFTZ ; An ; St ; CO	1 AC A, W0 rithmetic tore resul DRCON = 0x Before Instructio 00 320F 00 320F AC B, W12 rithmetic core resul DRCON = 0x Before Instructio	shift ACC t to ACCA 0000 (sat 000 FFFC AB09 0000 0000 0000 shift ACC t to ACCB 0040 (SAT	A by (W0) uration dia W0 ACCA CORCON SR B by (W12)	After Instructio 03 20FA B 0 8 0 8 0 8 0 0 0 0 0 0 0 0 0 0 0 0 0	FFC 090 000 800 (OA, C	0AB=1)
Cycles: Example 1 SFTJ ; Af ; Sf ; CO W0 ACCA CORCON SR Example 2 SFTZ ; An ; St ; CO W12	1 AC A, W0 rithmetic tore resul ORCON = 0x Before Instructio 00 320F 00 320F 00 320F Core resul ORCON = 0x Before Instructio	shift ACC t to ACCA 0000 (sat 500 FFFC AB09 0000 0000 0000 shift ACC t to ACCB 0040 (SAT 000F	A by (W0) uration dia W0 ACCA CORCON SR B by (W12) B = 1) W12	After Instructio 03 20FA B 00 8 00 8 00 8 00 8 00 8 00 8 00 8 00	FFC 090 800 (OA, C)AB=1)

SL		Shift Left	f			
Syntax:	{label:}	SL{.B}	f	{,WREG}		
Operands:	f ∈ [0 8	191]				
Operation:	$\begin{array}{c} (f < 7 >) \\ (f < 6:0 > \\ 0 \rightarrow D \\ \hline For word \\ (f < 15 > \\ (f < 14:0 \\ \end{array} \end{array}$	pperation: → (C) → Dest<7: est<0> operation:) → (C) >) → Dest<1 est<0>				
	[]	◀-0				
Status Affected:	N, Z, C		- 1	-		
Encoding:	1101	0100	OBDf	ffff	ffff	ffff
Description:	result in t register is shifted in	he destinatio s shifted into to the Least s	n register. Th the Carry bit Significant bit	one bit to the e Most Signif of the Status of the destina	icant bit of the register, and a tion register.	e file zero is
	WREG is	specified, th		mines the des red in WREG file register.	-	
	The 'D' b	t selects the		ration (0 for v 0 for WREG, ∷ e file register.		
		rather thar denote a w	a word operation	instruction de ation. You ma n, but it is not rking register	y use a . w ex required.	
Words:	1					
Cycles:	1					
Example 1 SL.B	0x909	; Shift l	eft (0x909) (Byte mod	le)	
Data 0908 SR	Before Instruction 9439 0000	Data 090 S		C=1)		
Example 2 SL	0x1650,			t (0x1650) ult in WREG		e)
li WREG (W0) Data 1650 SR	Before nstruction 0900 4065 0000	WREG (W Data 169 S	-	N=1)		

SL		Shift Left	Ws			
Syntax:	{label:}	SL{.B}	Ws,	Wd		
			[Ws],	[Wd]		
			[Ws++],	[Wd++]		
			[Ws],	[Wd]		
			[++Ws],	[++Wd]		
			[Ws],	[Wd]		
Operands:	Ws ∈ [W0 Wd ∈ [W0	-				
Operation:	$(Ws < 7)$ $(Ws < 6)$ $0 \rightarrow W$ <u>For word</u> $(Ws < 1)$	$(0^{>}) \rightarrow Wd < d<0^{>}$ operation: $(5^{>}) \rightarrow C$ $(4:0^{>}) \rightarrow Wd$				
		-0				
Status Affected:	N, Z, C				-	
Encoding:	1101	0000	0Bqq	qddd	dppp	SSSS
Description:	the result shifted int Least Sig	in the destir to the Carry	nation register bit of the Sta of Wd. Either	gister Ws one I r Wd. The Mos tus register, an register direct o	t Significant b ld 0 is shifted	oit of Ws i I into the
	The 'q' bi The 'd' bi The 'p' bi	ts select the ts select the ts select the	destination A address of th source Addr	eration (0 for w Address mode. ne destination ess mode. ne source regis	register.	te).
	Note:	rather tha	n a word ope	e instruction de ration. You ma n, but it is not	yusea.we	
	1					
Words:						
Words: Cycles:	1					
			left W3 (By result to W			
Cycles:	W3, W4 Before		result to V	14		
Cycles: Example 1 SL.B	W3, W4 Before Instruction	; Store :	After	14		
Cycles:	W3, W4 Before Instruction 3 78A9	; Store :	result to V	14		

dsPIC30F Programmer's Reference Manual

Example 2 SL	[W2++],	[W12] ; ; ;	Store	left resul increm	t to	[W12]	mode)
	Before		After				
	Instructio	n	Instruction				
W2	0900		W2	0902	2		
W12	2 1002		W12	1002	2		
Data 0900) 800F	Dat	a 0900	8001	7		
Data 1002	6722	Dat	a 1002	0011	Ξ		
SF	0000	'	SR	0001	L (C=1	I)	

SL Shift Left by Short Literal								
Syntax:	{label:}	SL	Wb,	#lit4,	Wnd			
Operands:	Wb ∈ [W0 lit4 ∈ [0 ⁻ Wnd ∈ [W							
Operation:	Wnd<15:8	4<3:0> → Shift_Val /nd<15:Shift_Val> = Wb<15-Shift_Val:0> /d <shift_val-1:0> = 0</shift_val-1:0>						
Status Affected:	N, Z							
Encoding:	1101	1101	0www	wddd	d100	kkkk		
Description: Shift left the contents of the source register Wb by the 4-bit unsigned literal and store the result in the destination register Wnd. Any bits shifted out of the source register are lost. Direct addressing must be used for Wb and Wnd.								
	The 'd' bit	ts select the a s select the ac s provide the l	ddress of the	edestination	register.	er.		
	Note:	This instruct	ion operates	in Word mo	de only.			
Words:	1							
Cycles:	1							
Example 1 SL	W2, #4, W2	-	left W2 b result to	-				
	Before Instruction /2 78A9 R 0000	W2 SR		√ =1)				
Example 2 SL	W3, #12, W		left W3 k result to	-				
V	Before Instruction /3 0912 /8 1002 R 0000	W3 W8 SR	3 2000					

Syntax: [label:] SL Wb, Wns, Wnd Operands: Wb ∈ [W0 W15] Wns ∈ [W0 W15] Wns ∈ [W0 W15] Wns <4:0> → Shift_Val Wns <4:0> → Shift_Val Wns <4:0> → Shift_Val Wnd <15:Shift_Val> = Wb <15-Shift_Val:0> Wd <shift_val-1:0> = 0 Status Affected: N, Z Encoding: 1101 1101 0www wddd d000 Description: Shift left the contents of the source register Wb by the 5 Leas Significant bits of Wns (only up to 15 positions) and store the the destination register Wnd. Any bits shifted out of the source are lost. Register direct addressing must be used for Wb, Wr Wnd. The 'w' bits select the address of the base register. The 'd' bits select the address of the base register. The 'd' bits select the address of the source register. The 'd' bits select the address of the source register. The 'd' bits select the address of the oscillation register. The 's' bits select the address of the oscillation of only. 2: If Wns is greater than 15, Wnd will be loaded with' Words: 1 Cycles: 1 Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> ; Store result to W2 Before After Instruction Instruction V0 0004 W1 0000 SR W2 78A9 SR 0000 SR SR Store result to W6 Store result to</shift_val-1:0>				y Wns	Shift Left b		SL				
Wns ∈ [W0W15] Wnd ∈ [W0 W15] Operation: Wns < [W0 W15] Operation: Wns < [W0 W15] Operation: Wns < [W15] Wd <shift_val> = Wb<15-Shift_Val:0> Wd<shift_val-1:0> = 0 Status Affected: N, Z Encoding: 1101 1101 0www wddd d000 Description: Shift left the contents of the source register Wb by the 5 Leas Significant bits of Wns (only up to 15 positions) and store the the destination register Wnd. Any bits shifted out of the sourc are lost. Register direct addressing must be used for Wb, Wr Wnd. The 'w' bits select the address of the base register. The 's' bits select the address of the source register. The 's' bits select the address of the source register. Note 1: This instruction operates in Word mode only. 2: Words: 1 Cycles: 1 Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> ; Store result to W2 Before After Instruction Instruction W0 09A4 W1 8903 W2 W2 4D20 SR 0000 SR 0000 SR Before After Instruction Instruction <</shift_val-1:0></shift_val>		Wnd	Wns,	Wb,	SL	{label:}	Syntax:				
Wnd<15:Shift_Val> = Wb<15-Shift_Val:0> Wd <shift_val-1:0> = 0 Status Affected: N, Z Encoding: 1101 1101 0www wddd d000 Description: Shift left the contents of the source register Wb by the 5 Leas Significant bits of Wns (only up to 15 positions) and store the the destination register Wnd. Any bits shifted out of the source are lost. Register direct addressing must be used for Wb, Wr Wnd. The w' bits select the address of the base register. The 'd' bits select the address of the source register. The 'd' bits select the address of the source register. Note 1: This instruction operates in Word mode only. 2: If Wns is greater than 15, Wnd will be loaded with Words: 1 Cycles: 1 Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> ; Store result to W2 Before After Instruction Instruction W1 8903 W1 8903 W2 78A9 SR 0000 SR 00000 SR Store result to W6 Before After Instruction Instruction Before After Instruction Instruction Before After Instruction Store result to W6</shift_val-1:0>)W15]	Wns ∈ [W0	Operands:				
Encoding: 1101 1101 0www wdd d000 Description: Shift left the contents of the source register Wb by the 5 Leas Significant bits of Wns (only up to 15 positions) and store the the destination register Wnd. Any bits shifted out of the source are lost. Register direct addressing must be used for Wb, Wr Wnd. The 'w' bits select the address of the base register. The 'd' bits select the address of the destination register. The 'd' bits select the address of the source register. Note 1: This instruction operates in Word mode only. 2: If Wns is greater than 15, Wnd will be loaded with Words: 1 Cycles: 1 Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> ; Store result to W2 Before After Instruction Instruction W0 09A4 W1 8903 W2 78A9 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6 Before After Instruction Instruction W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6			/al:0>	/b<15-Shift_`	hift_Val> = W	Wnd<15:S	Operation:				
Description: Shift left the contents of the source register Wb by the 5 Lead Significant bits of Wns (only up to 15 positions) and store the the destination register Wnd. Any bits shifted out of the source are lost. Register direct addressing must be used for Wb, Wr Wnd. The 'w' bits select the address of the base register. The 'd' bits select the address of the destination register. The 'd' bits select the address of the source register. The 'd' bits select the address of the source register. The 's' bits select the address of the source register. Note 1: This instruction operates in Word mode only. 2: If Wns is greater than 15, Wnd will be loaded with Words: Words: 1 Cycles: 1 Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> ; Store result to W2 Before After Instruction W1 W1 8903 W2 W2 78A9 0000 SR 00000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6 Before After Instruction Instruction Instruction						N, Z	Status Affected:				
Significant bits of Wns (only up to 15 positions) and store the the destination register Wnd. Any bits shifted out of the source are lost. Register direct addressing must be used for Wb, Wr Wnd. The 'w' bits select the address of the base register. The 'd' bits select the address of the base register. The 'd' bits select the address of the base register. The 'd' bits select the address of the base register. The 'd' bits select the address of the base register. The 's' bits select the address of the base register. The 's' bits select the address of the base register. The 's' bits select the address of the base register. The 's' bits select the address of the base register. The 's' bits select the address of the base register. The 's' bits select the address of the base register. The 's' bits select the address of the base register. Note 1: This instruction operates in Word mode only. 2: If Wns is greater than 15, Wnd will be loaded with Words: 1 Cycles: 1 Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> W0 09A4 W0 W1 8903 W2 78A9 SR 0000 SR 0000 SR 0000 <td>SSSS</td> <td>d000</td> <td>wddd</td> <td>0www</td> <td>1101</td> <td>1101</td> <td>Encoding:</td>	SSSS	d000	wddd	0www	1101	1101	Encoding:				
The 'd' bits select the address of the destination register. The 's' bits select the address of the source register. Note 1: This instruction operates in Word mode only. 2: If Wns is greater than 15, Wnd will be loaded with Words: 1 Cycles: 1 Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> j: Store result to W2 Before After Instruction Instruction Instruction W0 09A4 W0 09A4 W1 8903 W1 8903 W2 78A9 W2 4D20 SR 0000 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> j: Store result to W6 Before After Instruction Instruction instruction	result in registe	nd store the r of the source	positions) a s shifted out	only up to 15 Wnd. Any bit	bits of Wns (ition register	Significant the destina are lost. Re					
2: If Wns is greater than 15, Wnd will be loaded with Words: 1 Cycles: 1 Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> ; Store result to W2 Before After Instruction Unstruction W0 09A4 W1 8903 W2 78A9 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6 Before After Instruction Unstruction		register.	destination	ddress of the	select the ad	The 'd' bits					
Cycles: 1 Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> ; Store result to W2 Before After Instruction Instruction W0 09A4 W0 09A4 W1 8903 W1 8903 W2 78A9 W2 4D20 SR 0000 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6	10x0.					2:					
Example 1 SL W0, W1, W2 ; Shift left W0 by W1<0:4> ; Store result to W2 Before After Instruction Instruction W0 09A4 W1 8903 W2 78A9 W2 4D20 SR 0000 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6											
; Store result to W2 Before After Instruction Instruction W0 09A4 W0 09A4 W1 8903 W1 8903 W2 78A9 W2 4D20 SR 0000 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6						1	Cycles:				
Instruction Instruction W0 09A4 W0 09A4 W1 8903 W1 8903 W2 78A9 W2 4D20 SR 0000 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> Before After Instruction Instruction			-			0, W1, W2	Example 1 SL W				
W0 09A4 W0 09A4 W1 8903 W1 8903 W2 78A9 W2 4D20 SR 0000 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> Before After Instruction Instruction				After		Before					
W1 8903 W1 8903 W2 78A9 W2 4D20 SR 0000 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> Before After Instruction Instruction						struction	-				
W2 78A9 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6 Before After Instruction Instruction							_				
SR 0000 SR 0000 Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6 Before After Instruction Instruction											
Example 2 SL W4, W5, W6 ; Shift left W4 by W5<0:4> ; Store result to W6 Before After Instruction Instruction											
; Store result to W6 Before After Instruction Instruction				0000	38	0000	SK				
Instruction Instruction			-			4, W5, W6	Example 2 s⊾ พ				
				After		Before					
							-				
					W4	A409	W4				
W5 FF01 W5 FF01											
W6 0883 W6 4812 SR 0000 SR 0000											

SUB		Subtract W	REG from	F			
Syntax:	{label:}	SUB{.B}	f	{,WREG}			
Operands:	f ∈ [0 81	-					
Operation:	(f) – (WRE	G) \rightarrow destina	ation designa	ated by D			
Status Affected:	DC, N, OV,	Z, C					
Encoding:	1011	0101	OBDf	ffff	ffff	ffff	
Description:	Scription: Subtract the contents of the default working register WREG from the contents of the specified file register, and place the result in the destination register. The optional WREG operand determines the destination register. If WREG is specified, the result is stored in WREG if WREG is not specified, the result is stored in the file register.						
The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 'D' bit selects the destination (0 for WREG, 1 for file register). The 'f' bits select the address of the file register.							
		The extension rather than a denote a wo The WREG	a word opera ord operation	tion. You ma , but it is not	y use a . w e required.		
Words:	1						
Cycles:	1						
Example 1 SUB.B		; Sub. WRE ; Store re		-	te mode)		
	Before		After				
-	nstruction		Instruction	I			
WREG (W0)	7804	WREG (W					
Data 1FFE SR	9439 0000	Data 1FF S		(N, C=1)			
Example 2 SUB	0xA04, WR	-	. WREG fro re result		(Word mod	le)	
Ir WREG (W0) Data 0A04 SR	Before nstruction 6234 4523 0000	WREG (M Data 0A	4523	n (N=1)			

SUB		Subtract L	iteral from \	Vn		
Syntax:	{label:}	SUB{.B}	#lit10,	Wn		
Operands:		255] for by 1023] for w) W15]		n		
Operation:	(Wn) – lit1	$0 \rightarrow Wn$				
Status Affected:	DC, N, O\	/, Z, C				
Encoding:	1011	0001	0Bkk	kkkk	kkkk	dddd
Description:	working re	egister Wn, a	nd store the	operand from result back in st be used for	the working	
	The 'k' bit	t selects byte s specify the s select the a	literal operar		ister.	
		rather than denote a we For byte op unsigned va eral Opera	a word opera ord operation erations, the alue [0:255].	instruction de ation. You may n, but it is not literal must b See Section rmation on us	vuse a .wex required. e specified a 4.6 "Using ?	xtension to is an 10-bit Lit-
Words:	1					
Cycles:	1					
Example 1 SUB.B	#0x23, V		o. 0x23 fro ore result	om W0 (Byte to W0	e mode)	
lr W0 SR	Before Instruction 7804 0000	W SI		N=1)		
Example 2 SUB	#0x108,		o. 0x108 f: pre result	rom W4 (Wor to W4	d mode)	
lr W4 SR	Before Instruction 6234 0000	W SI		C=1)		

SUB	Subtract Short Literal from Wb							
Syntax:	{label:}	SUB{.B}	Wb,	#lit5,	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]			
Operands:	Wb ∈ [W0 lit5 ∈ [0 Wd ∈ [W0	31]						
Operation:	(Wb) – lit5	\rightarrow Wd						
Status Affected:	DC, N, OV	, Z, C						
Encoding:	0101	0www	wBqq	qddd	d11k	kkkk		
Description:	register Wi direct addr	o, and place	ned literal op the result in t be used for ed for Wd.	he destinatio	n register Wo	I. Register		
	The 'B' bit The 'q' bits The 'd' bits	selects byte select the c select the a provide the The extens rather than	address of the or word oper lestination Ac address of the literal operar ion . B in the a word operation	ation (0 for v ldress mode e destination id, a five-bit instruction d tion. You ma	vord, 1 for by register. integer numb enotes a byte y use a .we	er. e operation		
Words:	1				required.			
Cycles:	1							
Example 1 SUB.B	W4, #0x1		Sub. 0x10 Store resu		Byte mode)			
W4 W5 SR Example 2 SUB	Before nstruction 1782 7804 0000 W0, #0x8	W W SI , [W2++]	5 7872 R 0005 (C ; Sub. 0x ; Store r	DV,C=1) 8 from W0 result to crement W3		٤)		
W0 W2 Data 2004 SR	Before nstruction F230 2004 A557 0000	W W Data 200 Sl	After Instruction 0 F230 2 2006 4 F228					

SUB		Subtract W	s from Wb			
Syntax:	{label:}	SUB{.B}	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wd ∈ [W0	W15]				
Operation:	(Wb) – (W	(s) → Wd				
Status Affected:	DC, N, O\	/, Z, C				
Encoding:	0101	0www	wBqq	qddd	dppp	SSSS
Description:	Subtract the contents of the source register Ws from the contents of base register Wb and place the result in the destination register Wd. Register direct addressing must be used for Wb. Either register direct indirect addressing may be used for Ws and Wd. The 'w' bits select the address of the base register. The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 'q' bits select the destination Address mode. The 'd' bits select the address of the destination register. The 'p' bits select the source Address mode. The 's' bits select the address of the source register.					
	Note:	rather than a		ition. You ma	enotes a byte ay use a .we required.	
Words:	1					
Cycles:	1					
Example 1 SUB.B	W0, W1,	-	. W1 from re result		mode)	
W0 W1 SR	Before nstruction 1732 7844 0000	W(W ^r SF	7844	DC, N=1)		

Example 2 SUB	W7, [W8++],	[W9++]	; Sub. [W8] from W7 (Word mode) ; Store result to [W9] ; Post-increment W8 ; Post-increment W9
	Before		After
	nstruction	I	nstruction
W7	2450	W7	2450
W8	1808	W8	180A
W9	2020	W9	2022
Data 1808	92E4	Data 1808	92E4
Data 2022	A557	Data 2022	916C
SR	0000	SR	010C (DC, N, OV=1)

SUB	Subt	tract Accumulato	rs				
Syntax:	{label:} SUB	Acc					
Operands:	$Acc \in [A,B]$						
Operation:	If (Acc = A): ACCA – ACCI Else: ACCB – ACC						
Status Affected:	OA, OB, OAB, S	A, SB, SAB					
Encoding:	1100 1	011 A011	0000	0000	0000		
Description:	Description: Subtract the contents of the unspecified accumulator from the contents of Acc, and store the result back into Acc. This instruction performs a 40-bit subtraction.						
	The 'A' bit specif	ies the destination	accumulator.				
Words:	1						
Cycles:	1						
Example 1 SUB	; Store	ract ACCB from the result to DN = 0x0000 (no	ACCA	.)			
			Allei				
	Instruction		Instructio	n			
ACCA		ACCA	Instructio				
ACCA ACCB	Instruction	ACCA ACCB	Instruction				
	Instruction 76 120F 098A		Instruction 52 1EFC 4 23 F312 B	D73 C17 000			
ACCB	Instruction 76 120F 098A 23 F312 BC17	ACCB	Instruction 52 1EFC 4 23 F312 B 0 0 0	D73 C17	ıB=1)		
ACCB CORCON	Instruction 76 120F 098A 23 F312 BC17 0000 0000 B ; Subtr ; Store	ACCB CORCON	Instruction 52 1EFC 4 23 F312 B 0 0 1 ACCB ACCB	D73 C17 000	ıB=1)		
ACCB CORCON SR	Instruction 76 120F 098A 23 F312 BC17 0000 0000 B ; Subtr ; Store ; CORCC Before	ACCB CORCON SR act ACCA from	Instruction 52 1EFC 4 23 F312 B 0 1 ACCB ACCB TB = 1) After	D73 C17 000 100 (OA, C	ıB=1)		
ACCB CORCON SR Example 2 SUB	Instruction 76 120F 098A 23 F312 BC17 0000 0000 B ; Subtr ; Store ; CORCC Before Instruction	ACCB CORCON SR Fact ACCA from the result to N = 0x0040 (SA	Instruction 52 1EFC 4 23 F312 B 0 1 ACCB ACCB TB = 1) After Instruction	D73 C17 000 100 (OA, C	₽B=1)		
ACCB CORCON SR	Instruction 76 120F 098A 23 F312 BC17 0000 0000 B ; Subtr ; Store ; CORCC Before	ACCB CORCON SR act ACCA from	Instruction 52 1EFC 4 23 F312 B 0 1 ACCB ACCB TB = 1) After Instruction	D73 C17 000 100 (OA, C	'B=1)		
ACCB CORCON SR Example 2 SUB	Instruction 76 120F 098A 23 F312 BC17 0000 0000 B ; Subtr ; Store ; CORCC Before Instruction FF 9022 2EE1	ACCB CORCON SR act ACCA from the result to N = 0x0040 (SA ACCA	Instruction 52 1EFC 4 23 F312 B 0 0 1 ACCB 1 1 A	D73 C17 000 100 (OA, C	B=1)		

SUBB		Subtract W	REG and C	arry bit fror	n f	
Syntax:	{label:}	SUBB{.B}	f	{,WREG}		
Operands: Operation: Status Affected:	f ∈ [0 8′ (f) – (WRE DC, N, OV	$(\overline{G}) - (\overline{C}) \rightarrow d$	estination de	esignated by	D	
Encoding:	1011	0101	1BDf	ffff	ffff	ffff
Description:	Borrow fla register ar WREG op specified,	te contents of g (Carry flag i d place the re erand determ the result is s ored in the file	nverse, \overline{C}) fires the constant of the constant of the destimation of the destimation of the destimation of the destimation of the destination	rom the cont lestination re tination regis	ents of the s gister. The o ster. If WRE	pecified file optional G is
	The 'D' bit	selects byte of selects the d select the ad	estination (0	for WREG,	1 for file reg	
	2:	The extension rather than a denote a wo The WREG The Z flag is These instru	word opera rd operation is set to wor "sticky" for	tion. You ma , but it is not king register ADDC,CPB	y use a . w e required. W0.	extension to
Words:	1					
Cycles:	1					
Example 1 SUBB.1	B 0x1FFF	; Sub. WRE ; Store re			F) (Byte r	node)
l WREG (W0) Data 1FFE SR	Before nstruction 7804 9439 0000	WREG (W0) Data 1FFE SF	8F39	√ =1)		
Example 2 SUBB	0xA04, WRE		WREG and to result to	E from (0x) WREG	A04) (Word	d mode)
ا WREG (W0) Data 0A04 SR		WREG (W0) Data 0A04 SR	6235	C=1)		

SUBB	Subtract Wn from Literal with Borrow						
Syntax:	{label:} SUBB{.B} #lit10, Wn						
Operands:	lit10 \in [0 255] for byte operation lit10 \in [0 1023] for word operation Wn \in [W0 W15]						
Operation:	$(Wn) - Iit10 - (\overline{C}) \rightarrow Wn$						
Status Affected:	DC, N, OV, Z, C						
Encoding:	1011 0001 1Bkk kkkk kkkk dddd						
Description:	Subtract the <u>unsigned 10-bit literal operand and the Borrow flag (Carry</u> flag inverse, C) from the contents of the working register Wn, and store the result back in the working register Wn. Register direct addressing must be used for Wn. The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 'k' bits specify the literal operand. The 'd' bits select the address of the working register.						
	 denote a word operation, but it is not required. 2: For byte operations, the literal must be specified as an unsigned value [0:255]. See Section 4.6 "Using 10-bit Literal Operands" for information on using 10-bit literal operands in Byte mode. 3: The Z flag is "sticky" for ADDC, CPB, SUBB and SUBBR. These instructions can only clear Z. 						
Words:	1						
Cycles:	1						
Example 1 SUBB	5.B #0x23, W0 ; Sub. 0x23 and \overline{C} from W0 (Byte mode) ; Store result to W0						
W SI Example 2 SUBB	R 0000 SR 0108 (DC, N=1)						
W							

SUBB		Subtract S	hort Liter	al from Wb w	vith Borrow	
Syntax:	{label:}	SUBB{.B}	Wb,	#lit5,	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 lit5 ∈ [0 Wd ∈ [W0	. 31]				
Operation:	-	$\overline{5} - (\overline{C}) \rightarrow Wd$				
Status Affected:	DC, N, O	V, Z, C				
Encoding:	0101	lwww	wBqq	qddd	d11k	kkkk
Description:						d place th ng must b
	The 'B' bi The 'q' bit The 'd' bit The 'k' bit Note 1	t selects byte ts select the c ts select the a s provide the The extensi rather than denote a wo	or word o lestination address of literal ope ion . B in th a word operat	Address mod the destinatio rand, a five-b he instruction eration. You m ion, but it is no	r word, 1 for by le. n register. it integer numb denotes a byte nay use a . w e:	oer. e operatio xtension
	E.			n only clear Z		SODDIC.
Words:	1					
Cycles:	1					
Example 1 SUBB	.B W4, #0:			10 and \overline{C} fresult to W5	om W4 (Byte	mode)
W4 W5 SF Example 2 SUBB	5 7804		5 7871 R 0005	(OV, C=1) (OV, C=1)	om W0 (Word	mode)
	Before	-		result to [ncrement W2		
W(W2	Instruction 0 0009 2 2004	W W	Instructio 0 0009 2 2006)		
Data 2004 SF		Data 200 =1) SI		(DC, Z, C=1)	

SUBB		Subtract V	/s from Wb	with Borrow	N	
Syntax:	{label:}	SUBB{.B}	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wd ∈ [W0	W15]				
Operation:	(Wb) – (N	$(s) - (\overline{C}) \rightarrow W$	/d			
Status Affected:	DC, N, O	/, Z, C				
Encoding:	0101	lwww	wBqq	qddd	dppp	SSSS
	place the addressin may be us The 'w' bi The 'B' bit The 'q' bit The 'q' bit The 'p' bit The 's' bit	result in the c g must be us sed for Ws ar ts select the a s selects byte s select the d s select the a s select the a	destination re ed for Wb. R address of th or word ope lestination A address of the ource Addre ddress of the	egister Wd. F legister dired e base regis ration (0 for ddress mode e destination ss mode. e source reg	word, 1 for by e. n register. ister.	ddressing te).
		rather than to denote a The Z flag i	a word operat	ation. You m ion, but it is ADDC,CPE	lenotes a byte ay use a . W not required. B, SUBB and	extensior
Words:	1					
Cycles:	1					
Example 1 SUBB.	B W0, W1,		ub. W1 and core result) (Byte mode	e)
W0 W1 SR	7844	W W SI	1 7844	DC, N=1)		

Example 2 SUBB	W7, [W8++], [ī	; S ; P	tore rea ost-inci	and C from W7 sult to [W9] rement W8 rement W9	(Word mode)
	Before		After		
I	nstruction	I	nstruction	I	
W7	2450	W7	2450		
W8	1808	W8	180A		
W9	2022	W9	2024		
Data 1808	92E4	Data 1808	92E4		
Data 2022	A557	Data 2022	916C		
SR	0000	SR	010C	(DC, N, OV=1)	

SUBBR	Subtract f from WREG with Borrow					
Syntax:	{label:}	SUBBR{.B}	f	{,WREG}		
Operands: Operation:		$(f) - (\overline{C}) \rightarrow de$	estination de	esignated by	D	
Status Affected:	DC, N, O\		1000			
Encoding: Description:	(Carry flag in the dest destination	1101 the contents of p inverse, \overline{C}) fit ination register n register. If W s not specified	rom the cont er. The optio /REG is spe	ents of WRE nal WREG o cified, the re	G, and place perand dete sult is stored	e the result rmines the in WREG.
	The 'D' bit	selects byte of selects the de select the ad	estination (0	for WREG,		
	2:	The extension rather than a denote a wo The WREG is The Z flag is These instru	word opera rd operation is set to worl "sticky" for 2	tion. You ma , but it is not king register ADDC, CPB,	y use a .w e required. W0.	xtension to
Words:	1			•		
Cycles:	1					
Example 1 SUBBR	.B 0x803	; Sub. (0x8 ; Store res			G (Byte mo	de)
Before After Instruction Instruction WREG (W0) 7804 Data 0802 9439 SR 0002 (Z=1) SR						
Example 2 SUBBR	0xA04, WR		(0xA04) an result to		WREG (Word	mode)
WREG (W0) Data 0A04 SR	Before nstruction 6234 6235 0000	WREG (W Data 0A S	6235]		

SUBBR		Suptract W	D from Sho	rt Literal wit	In Borrow	
Syntax:	{label:}	SUBBR{.B}	Wb,	#lit5,	Wd [Wd]	
					[Wd++]	
					[Wd]	
					[++Wd] [Wd]	
					[
Operands:	Wb ∈ [W0 lit5 ∈ [0 Wd ∈ [W0	31]				
Operation:	lit5 – (Wb)	$-(\overline{C}) \rightarrow Wd$				
Status Affected:	DC, N, OV	/, Z, C				
Encoding:	0001	lwww	wBqq	qddd	d11k	kkkk
Description:	flag invers destinatior	The contents of e, \overline{C}) from the register Wd.	5-bit unsig Register di	ned literal an rect addressi	d place the re ng must be u	esult in the
	The 'B' bit The 'q' bits The 'd' bits	s select the a selects byte o s select the de s select the ac s provide the l	or word ope estination Ad Idress of the	ration (0 for v ddress mode e destination	word, 1 for by register.	
		The extension rather than a denote a wo The Z flag is These instru	word operation rd operation "sticky" for	ation. You ma n, but it is not ADDC,CPB	y use a . w ex required.	tension t
Words:	1					
Cycles:	1					
Example 1 SUBB	R.B W0, #0	x10, W1 ; ; ; ;	Sub. W0 an Store rest		0x10 (Byte	mode)
	Before		After			
	Instruction		Instruction			
W		WC W1				
SI				DC, Z, C=1)		
	R W0, #0x8	, [W2++] ;		and \overline{C} from sult to [W		mode)
Example 2 SUBB	,		Post-inc:	rement W2		
Example 2 SUBB	Before	;	After	rement W2		
	Before	;	After Instruction	rement W2		
w	Before Instruction	; WC	After Instruction	rement W2		
	Before Instruction 0 0009 2 2004	;	After Instruction	rement W2		

SUBBR	SUBBR Subtract Wb from Ws with Borrow					
Syntax:	{label:}	SUBBR{.B}	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wd ∈ [W0) W15]				
Operation:	(Ws) – (W	$(b) - (\overline{C}) \rightarrow W_{C}$	d			
Status Affected:	DC, N, O	V, Z, C				
Encoding:	0001	lwww	wBqq	qddd	dppp	SSSS
	Ws and W The 'w' bi The 'B' bi The 'q' bit The 'd' bit The 'p' bit	or Wb. Registe /d. ts select the act s selects byte of s select the de s select the act s select the so	ddress of the or word oper estination Ac ddress of the ource Addres	e base regis ration (0 for ddress mode e destinatior ss mode.	ster. word, 1 for by e. n register.	
		The extension rather than a denote a wo The Z flag is These instru	a word opera rd operation "sticky" for	ition. You ma , but it is no ADDC,CPE	ay use a . w ex t required.	ctension t
Words:	1					
Cycles:	1					
Example 1 SUBB	R.B W0, W3		ub. W0 and tore resul		11 (Byte moo	le)
W(W SF	1 7844	W0 W1 SR	7844	C=1)		

Example 2 SUBBR	W7, [W8++],[[W9++] ; ; ; ;	Sub. W7 and \overline{C} from [W8] (Word mode) Store result to [W9] Post-increment W8 Post-increment W9
I	Before nstruction		After Instruction
W7	2450	N	7 2450
W8	1808	N	/8 180A
W9	2022	N	/9 2024
Data 1808	92E4	Data 180	08 92E4
Data 2022	A557	Data 202	22 6E93
SR	0000	S	R 0005 (OV, C=1)

SUBR		Subtract f	from WREG	ì		
Syntax:	{label:}	SUBR{.B}	f	{,WREG}		
Operands:	f ∈ [0 81	-				
Operation:		$(f) \rightarrow destination destinatio$	ation designa	ated by D		
Status Affected:	DC, N, OV	, Z, C	i	i		
Encoding:	1011	1101	OBDf	ffff	ffff	ffff
Description:	the default destination destination	Subtract the contents of the specified file register from the contents of the default working register WREG, and place the result in the destination register. The optional WREG operand determines the destination register. If WREG is specified, the result is stored in WREG If WREG is not specified, the result is stored in the file register.				
	The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 'D' bit selects the destination (0 for WREG, 1 for file register). The 'f' bits select the address of the file register.					
			a word opera ord operation	tion. You ma , but it is not	y use a . w e required.	
Words:	1					
Cycles:	1					
Example 1 SUBR. F	3 Ox1FFF		0x1FFF) fi result to	com WREG () 0x1FFF	Byte mode)	
	Before		After			
F	nstruction		Instruction			
WREG (W0)	7804	WREG (W0)				
Data 1FFE SR	9439 0000	Data TFFE SF				
Example 2 SUBR	0xA04, WR	-	(0xA04) i e result t	From WREG	(Word mode	2)
	Before		After			
-	nstruction		Instruction			
WREG (W0) Data 0A04	6234 6235	WREG (W0 Data 0A04				
SR	0000	SR		N=1)		

SUBR		Subtract W	/b from Sh	ort Literal		
Syntax:	{label:}	SUBR{.B}	Wb,	#lit5	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [Wi lit5 ∈ [0 Wd ∈ [Wi	. 31]				
Operation:	lit5 – (Wb	$\rightarrow Wd$				
Status Affected:	DC, N, O	V, Z, C				
Encoding:	0001	0www	wBqq	qddd	d11k	kkkk
Description:	literal ope Register	the contents or erand, and pla direct address ddressing ma	ce the resu	It in the destine used for Wb	nation registe	r Wd.
	The 'B' bi The 'q' bi The 'd' bi		or word ope lestination A ddress of th literal opera- on . B in the	eration (0 for address mode ne destination and, a five-bit e instruction o	word, 1 for by e. n register. integer numb denotes a byte	ber. e operati
				n, but it is no	ay use a .⊮ e t required.	xtension
Words:	1					
Cycles:	1					
Example 1 SUBR.	.B W0, #0			from 0x10 (Sult to W1	Byte mode)	
WC W1 SR	786A	W W SI	1 7800	(DC, Z, C=1)		
	W0, #0x	8, [W2++]	-		(Word mode)
Example 2 SUBR			-	result to (ncrement W2		

SUBR		Subtract W	b from Ws			
Syntax:	{label:}	SUBR{.B}	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wd ∈ [W0	W15]				
Operation:	(Ws) – (Wl	$() \rightarrow Wd$				
Status Affected:	DC, N, OV	, Z, C				
Encoding:	0001	0www	wBqq	qddd	dppp	SSSS
	indirect ad The 'w' bits The 'B' bit The 'q' bits The 'd' bits The 'p' bits The 's' bits	dressing may s select the a selects byte s select the d s select the a s select the s	/ be used for ddress of the or word oper estination Ac ddress of the ource Addres ddress of the	Ws and Wc e base regis ration (0 for v ddress mode e destination ss mode. e source regi	ter. word, 1 for by register. ster.	te).
	Note:	rather than		ation. You ma	lenotes a byte ay use a .we : required.	-
Words:	1					
Cycles:	1					
Example 1 SUBR.B	8 WO, W1,	-	b. W0 from ore result	-	mode)	
lr W0 W1 SR	Before nstruction 1732 7844 0000	Wi W ¹ SF	7844	C=1)		

Example 2 SUBR	W7, [W8++],	[W9++]	; Sub. W7 from [W8] (Word mode) ; Store result to [W9] ; Post-increment W8 ; Post-increment W9
	Before		After
I	nstruction	I	nstruction
W7	2450	W7	2450
W8	1808	W8	180A
W9	2022	W9	2024
Data 1808	92E4	Data 1808	92E4
Data 2022	A557	Data 2022	6E94
SR	0000	SR	0005 (OV, C=1)

SWAP		Byte or Nit	oble Swap V	Vn				
Syntax:	{label:}	SWAP{.B}	Wn					
Operands:	Wn ∈ [W0) W15]						
Operation:	For word	$:4> \leftrightarrow (Wn) < 3$						
Status Affected:	None							
Encoding:	1111	1101	1B00	0000	0000	SSSS		
Description:	Swap the contents of the working register Wn. In Word mode, the two bytes of Wn are swapped. In Byte mode, the two nibbles of the Least Significant Byte of Wn are swapped, and the Most Significant Byte of Wn is unchanged. Register direct addressing must be used for Wn.							
The 'B' bit selects byte or word operation (0 for word, 1 for byte). The 's' bits select the address of the working register.								
	Note:	The extension rather than a denote a wo	a word opera	tion. You ma	ay use a .we			
Words:	1							
Cycles:	1							
Example 1 SWAP.E	3 0x1021	; Nibble	e swap (0x	1021)				
ا Data 1020	Before Instruction	Data 1020	After Instruction					
SR	0000	SR	_					
Example 2 SWAP	0x1760	; Byte	swap (0x17	60)				
lı Data 1760 SR	Before nstruction 8095 0000	Data 1760 SR						

© 2003 Microchip Technology Inc.

TBLRDH		Table Read F	ligh					
Syntax:	{label:}	TBLRDH{.B}	[Ws],	Wd				
			[Ws++],	[Wd]				
			[Ws],	[Wd++]				
			[++Ws],	[Wd]				
			[Ws],	[++Wd]				
				[Wd]				
Operands:	Ws ∈ [W0 . Wd ∈ [W0 .	-						
Operation:	For byte op If (LSB(V $0 \rightarrow V$ Else	Vs)=1)						
	For word o	Mem [(TBLPA						
Status Affected:	None							
Encoding:	1011	1010	1Bqq	qddd	dppp	5555		
Description:	Read the contents of the Most Significant Word of program memory and store it to the destination register Wd. The target word address of program memory is formed by concatenating the 8-bit Table Pointer register, TBLPAG<7:0>, with the effective address specified by Ws. Indirect addressing must be used for Ws, and either register direct or indirect addressing may be used for Wd.							
	In Word mode, zero is stored to the Most Significant Byte of the destination register (due to non-existent program memory) and the third program memory byte (PM<23:16>) at the specified program memory address is stored to the Least Significant Byte of the destination register.							
	In Byte mode, the source address depends on the contents of Ws. If Ws is not word aligned, zero is stored to the destination register (due to non-existent program memory). If Ws is word aligned, the third program memory byte (PM<23:16>) at the specified program memory address is stored to the destination register.							
	memory by	nt program me te (PM<23:16	mory). If Ws >) at the spec	destination re is word aligne	egister (due ed, the third	to program		
	memory by stored to th The 'B' bits The 'q' bits The 'd' bits The 'p' bits	nt program me te (PM<23:16	mory). If Ws >) at the spec register. word operat tination Addr lress of the d rce Address	destination re is word aligne cified program ion (0 for work ress mode. lestination (da mode.	egister (due ed, the third n memory ac d, 1 for byte nta) register.	to program ldress is		
	memory by stored to th The 'B' bits The 'q' bits The 'd' bits The 'p' bits The 's' bits Note:	nt program me te (PM<23:16) ne destination i selects byte or select the des select the add select the sou	mory). If Ws register. word operat tination Addr lress of the d lress of the s l B in the ins nove. You m	destination re is word aligned cified program ion (0 for work ress mode. estination (da mode. ource (address struction dence ay use a .w	egister (due ed, the third n memory ac d, 1 for byte nta) register. ss) register. otes a byte n	to program Idress is). nove rather		
Words:	memory by stored to th The 'B' bits The 'q' bits The 'd' bits The 'p' bits The 's' bits Note:	nt program me te (PM<23:16: e destination i selects byte or select the des select the ado select the ado select the ado The extension than a word r	mory). If Ws register. word operat tination Addr lress of the d lress of the s l B in the ins nove. You m	destination re is word aligned cified program ion (0 for work ress mode. estination (da mode. ource (address struction dence ay use a .w	egister (due ed, the third n memory ac d, 1 for byte nta) register. ss) register. otes a byte n	to program Idress is). nove rathe		

Example 1 TBL	RDH.B [WO]	•	PM (TBLPAG:[to [W1] increment W1	W0]) (Byte mode)
	Before		After	
	Instruction		Instruction	
W0	0812	W0	0812	
W1	0F71	W1	0F72	
Data 0F70	0944	Data 0F70	EF44	
Program 01 0812	EF 2042	Program 01 0812	EF 2042	
TBLPAG	0001	TBLPAG	0001	
SR	0000	SR	0000	
Example 2 TBL	RDH [W6++]	; Store t]) (Word mode)
	Before		After	
	Instruction		Instruction	
W6	3406	W6	3408	
W8	65B1	W8	0029	
Program 00 3406	29 2E40	Program 00 3406	29 2E40	
TBLPAG	0000	TBLPAG	0000	
SR	0000	SR	0000	

TBLRDL		Table Read L	.ow			
Syntax:	{label:}	TBLRDL{.B}	[Ws],	Wd		
			[Ws++],	[Wd]		
			[Ws],	[Wd++]		
			[++Ws],	[Wd]		
			[Ws],	[++Wd]		
				[Wd]		
Operands:	Ws ∈ [W0 Wd ∈ [W0	-				
Operation:	For byte op If (LSB(\ Progra		PAG).(Ws)]	<15:8> → W	d	
	Else			10.0	u	
	-	am Mem [(TBl	LPAG),(Ws)]	$<7:0> \rightarrow Wd$		
	<u>For word o</u> Program	<u>peration:</u> Mem [(TBLP	AG),(Ws)] <1	$5:0> \rightarrow Wd$		
Status Affected:	None	•				
Encoding:	1011	1010	0Bqq	qddd	dppp	SSSS
Description:	store it to tl memory is TBLPAG<7 addressing	ontents of the ne destination formed by cor ':0>, with the must be used may be used	register Wd. ncatenating tl effective addr d for Ws, and	The target w ne 8-bit Table ress specified	ord address of Pointer regis I by Ws. Indire	of program ster, ect
	destination contents of memory wo word aligne	ode, the lower register. In By Ws. If Ws is in ord (PM<15:7 ad, the first by ne destination	yte mode, the not word align >) is stored to te of the prog	e source addr ned, the seco o the destinat	ess depends nd byte of the ion register. If	on the program Ws is
	The 'q' bits The 'd' bits The 'p' bits	selects byte o select the de select the ad select the sol select the add	stination Add dress of the c urce Address	ress mode. lestination (d mode.	ata) register.	r byte).
	Note:	The extension than a word word move, b	move. You n	nay use a .w	•	
				quilou.		
Words:	1	nord more, s		quireu.		

Section 5. Instruction Descriptions

Example 1 TBL	RDL.B [W0+	+], W1 ; Read PM ; Store t ; Post-in		10]) (Byte mode)
	Before		After	
	Instruction		Instruction	
W0	0813	W0	0814	
W1	0F71	W1	0F20	
Data 0F70	0944	Data 0F70	EF44	
Program 01 0812	EF 2042	Program 01 0812	EF 2042	
TBLPAG	0001	TBLPAG	0001	
SR	0000	SR	0000	
Example 2 TBL	RDL [W6],	; Store	PM (TBLPAG: to W8 increment W	:[W6]) (Word mode) N8
	Before		After	
	Instruction		Instruction	
W6	3406	W6	3408	
W8	1202	W8	1204	
Data 1202	658B	Data 1202	2E40	
Program 00 3406	29 2E40	Program 00 3406	29 2E40	
TBLPAG	0000	TBLPAG	0000	
SR	0000	SR	0000	

TBLWTH		Table Write F	ligh			
Syntax:	{label:}	TBLWTH{.B}	Ws,	[Wd]		
			[Ws],	[Wd++]		
			[Ws++],	[Wd]		
			[Ws],	[++Wd]		
			[++Ws],	[Wd]		
			[Ws],			
Operands:	Ws ∈ [W0 Wd ∈ [W0					
Operation:	<u>For byte o</u> If (LSB(NOP	<u>peration:</u> Wd) = 1)				
	Else				_	
	(Ws) <u>For word c</u>	→ Program Me operation:	em [(TBLPA	G),(Wd)]<23:1	6>	
		$0> \rightarrow \text{Program}$	Mem [(TBL	PAG),(Wd)] <2	23:16>	
Status Affected:	None					
Encoding:	1011	1011	1Bqq	qddd	dppp	SSSS
Description:	Significant program m register, Tl direct or in	contents of the Word of progra nemory is forme BLPAG<7:0>, v direct addressi	am memory. ed by concat vith the effec	The destinati enating the 8- ctive address s	on word addr bit Table Poir specified by V	ess of nter Vd. Eithe
	must be us	sed for Wd.				
	Since prog the upper using a We	sed for Wd. gram memory is byte of progran d that is word a ch a Wd that is	s 24-bits wid n memory (F ligned in Byt	e, this instruct PM<23:16>). T te mode or Wo	his may be p ord mode. If B	erformed yte mode
	Since prog the upper using a Wo is used wit The 'B' bit The 'q' bits The 'd' bits The 'p' bits	gram memory is byte of progran d that is word a	s 24-bits wid n memory (F ligned in Byt not word alig word opera stination Add lress of the o urce Address	e, this instruct PM<23:16>). T e mode or Wo gned, no opera tion (0 for wor lress mode. destination (ac s mode.	This may be p ord mode. If B ation is perfor d, 1 for byte) ddress) regist	erformeo yte mode med.
	Since prog the upper using a Wo is used wit The 'B' bit The 'q' bits The 'd' bits The 'p' bits	gram memory is byte of program d that is word a th a Wd that is selects byte or s select the des s select the ado	s 24-bits wid n memory (F ligned in Byt not word alig word opera tination Add tress of the s liress of the s n . B in the in nove. You n	e, this instruct 2M<23:16>). T a mode or Wo gned, no opera- tion (0 for wor lress mode. destination (ac source (data) i struction deno- nay use a .w	This may be p pord mode. If B ation is perfor rd, 1 for byte) ddress) regist register. otes a byte mo	erformed yte mode med. er. er.
Words:	Since prog the upper using a Wo is used wit The 'B' bit The 'A' bits The 'd' bits The 'p' bits The 's' bits	gram memory is byte of program d that is word a selects byte or select the des select the add select the add select the add The extension than a word r	s 24-bits wid n memory (F ligned in Byt not word alig word opera tination Add tress of the s liress of the s n . B in the in nove. You n	e, this instruct 2M<23:16>). T a mode or Wo gned, no opera- tion (0 for wor lress mode. destination (ac source (data) i struction deno- nay use a .w	This may be p pord mode. If B ation is perfor rd, 1 for byte) ddress) regist register. otes a byte mo	erformed yte mode med. er. ove rathe

Example 1 TBL	WTH.B [W0+	; to PM	e [W0] (H I Latch High increment W	n (TBLPAG:[W1])
	Before		After	
	Instruction		Instruction	
W0	0812	W0	0812	
W1	0F70	W1	0F70	
Data 0812	0944	Data 0812	EF44	
Program 01 0F70	EF 2042	Program 01 0F70	44 2042	
TBLPAG	0001	TBLPAG	0001	
SR	0000	SR	0000	
Note	are not u	•	SH memory	ntents of program memory is programmed using the Reference Manual.
Example 2 TBL	WTH W6, [; to PM I	M6 (Word Latch High Acrement W8	mode) (TBLPAG:[W8])
	Before		After	
	Instruction		Instruction	
W6	0026	W6	0026	
W8	0870	W8	0872	
Program 00 0870	22 3551	Program 00 0870	26 3551	
TBLPAG	0000	TBLPAG	0000	

SR

0000

Note: Only the Program Latch is written to. The contents of program memory are not updated until the FLASH memory is programmed using the procedure described in the dsPIC30F Family Reference Manual.

0000

SR

TBLWTL	ı	Table Write L	ow			
Syntax:	{label:}	TBLWTL{.B}	Ws,	[Wd]		
			[Ws],	[Wd++]		
			[Ws++],	[Wd]		
			[Ws],	[++Wd]		
			[++Ws],	[Wd]		
			[Ws],			
Operands:	Ws ∈ [W0 . Wd ∈ [W0 .	-				
Operation:	Else (Ws) – <u>For word op</u>	/d)=1) → Program Me → Program Me	em [(TBLPAC	6),(Wd)] <15:8 6),(Wd)] <7:0> ′Wd)] <15:0>		
Status Affected:	None	0				
Encoding:	1011	1011	0Bqq	qddd	dppp	SSSS
Description:	Store the co	ntanta af tha				<u>.</u>
	Word of pro memory is f TBLPAG<7	gram memory ormed by con :0>, with the e ressing may b	catenating the frective addr	ce register Ws ation word add ne 8-bit Table ess specified Vs, and indired	dress of prog Pointer regist by Wd. Eithe	ram ter, r direct o
	Word of pro memory is f TBLPAG<7 indirect add used for Wo In Word mo Byte mode, If Wd is not memory (PI	gram memory ormed by con :0>, with the e ressing may b I. de, Ws is stor the Least Sig word aligned	y. The destination of the destination of the destination of the second s	ation word ade ne 8-bit Table ress specified	dress of prog Pointer regist by Wd. Eithe ct addressing program men es the destina d byte of prog	ram ter, r direct o must be nory. In ation byte gram
	Word of pro memory is f TBLPAG<7 indirect add used for Wo In Word mo Byte mode, If Wd is not memory (PI program me The 'B' bits The 'q' bits The 'd' bits The 'p' bits	gram memory formed by con 0>, with the e ressing may b d. de, Ws is stor the Least Sig word aligned M<15:8>). If V emory (PM<7: elects byte or select the des select the ado select the sou	y. The destination of the destination of the destination of the low of the lo	ation word add ne 8-bit Table ess specified Vs, and indired ver 2 bytes of Wd determin d to the secon igned, Ws is s ion (0 for work ress mode. lestination (ad	dress of prog Pointer regist by Wd. Eithe ct addressing program men es the destina d byte of prog stored to the fi d, 1 for byte). Idress) registe	ram ter, r direct of must be nory. In ation byte gram irst byte o
	Word of pro memory is f TBLPAG<7 indirect add used for Wo In Word mo Byte mode, If Wd is not memory (PI program me The 'B' bits s The 'a' bits The 'd' bits The 's' bits The 's' bits	gram memory ormed by con 0>, with the e ressing may b d. de, Ws is stor the Least Sig word aligned M<15:8>). If V emory (PM<7: elects byte or select the des select the ado select the ado select the ado	y. The destination of the destination of the destination of the low of the low of the low on the low of the lo	ation word add ne 8-bit Table ess specified Vs, and indired ver 2 bytes of Wd determin d to the secon igned, Ws is s ion (0 for work ress mode. lestination (ad mode. ource (data) r nstruction den y use a . w ext	dress of prog Pointer regist by Wd. Eithe ct addressing program men es the destina d byte of prog tored to the fi d, 1 for byte). Idress) register egister. otes a byte m	ram ter, r direct of must be nory. In ation byte gram irst byte of er.
Words:	Word of pro memory is f TBLPAG<7 indirect add used for Wo In Word mo Byte mode, If Wd is not memory (PI program me The 'B' bits s The 'a' bits The 'd' bits The 's' bits The 's' bits	gram memory ormed by con 0>, with the e ressing may b d. de, Ws is stor the Least Sig word aligned, M<15:8>). If V emory (PM<7: elects byte or select the des select the ado select the ado select the ado the extension than a word m	y. The destination of the destination of the destination of the low of the low of the low on the low of the lo	ation word add ne 8-bit Table ess specified Vs, and indired ver 2 bytes of Wd determin d to the secon igned, Ws is s ion (0 for work ress mode. lestination (ad mode. ource (data) r nstruction den y use a . w ext	dress of prog Pointer regist by Wd. Eithe ct addressing program men es the destina d byte of prog tored to the fi d, 1 for byte). Idress) register egister. otes a byte m	ram ter, r direct of must be nory. In ation byte gram irst byte of er.

Example 1	TBLWTL.B	WO, [1		; Write W ; to PM L ; Post-in	atch	Low (1	mode) TBLPAG:[W1])
	Befo				Af		
	Instru	ction		_	Instru	uction	
١	N0	6628		W0		6628	
١	<i>N</i> 1	1225		W1		1226	
Program 00 12	24 78	0080	Progra	m 01 1224	78	2880	
TBLP	AG	0000		TBLPAG		0000	
:	SR	0000		SR		0000	
Ν	are	not up	dated ur	itil the FLA	SH me	emory	ntents of program me is programmed usin Reference Manual.
Example 2	FBLWTL	[W6],		; Write [; to PM L ; Post-in	atch	Low (1	rd mode) TBLPAG:[W8])
	Befo	ore			Af	ter	
	Instru	ction			Instru	uction	
١	N6	1600		W6		1600	

nemory ing the

	Before		After Instruction
W6	1600	W6	1600
W8	7208	W8	7208
Data 1600	0130	Data 1600	0130
Program 01 7208	09 0002	Program 01 7208	09 0130
TBLPAG	0001	TBLPAG	0001
SR	0000	SR	0000

Note: Only the Program Latch is written to. The contents of program memory are not updated until the FLASH memory is programmed using the procedure described in the dsPIC30F Family Reference Manual.

ULNK		De-allocate	Stack Fram	ne		
Syntax:	{label:}	ULNK				
Operands:	None					
Operation:	W14 \rightarrow W (W15)-2 \rightarrow (TOS) \rightarrow V	→ W15				
Status Affected:	None					
Encoding:	1111	1010	1000	0000	0000	0000
Description:	sequence. (W15) equ	ction de-alloo The stack fra al to the fram rame pointer	ame is de-all ne pointer (W	ocated by se	etting the sta	ick pointer
Words:	1					
Cycles:	1					
Example 1 ULNK		k the stac				
In	Before struction		After Instruction			
W14 [2002	W14				
W15	20A2	W1	5 2000			
Data 2000	2000	Data 2000	2000			
SR	0000	SF	R 0000			
Example 2 ULNK	; Unlir	k the stac	k frame			
	Before		After			
-	struction		Instruction			
W14	0802	W14				
W15	0812	W1				
Data 0800 SR	0800	Data 0800 SF				
SK	0000	J	0000			

Section 5. Instruction Descriptions

XOR		Exclusive	OR f and W	REG		
Syntax:	{label:}	XOR{.B}	f	{,WREG}		
Operands:	f ∈ [0 8	191]				
Operation:	(f).XOR.(\	$WREG) \rightarrow de$	stination des	signated by D		
Status Affected:	N, Z					
Encoding:	1011	0110	1BDf	ffff	ffff	ffff
Description:	default wo register a WREG op specified,	orking registe nd place the perand deterr	r WREG and result in the nines the de stored in WF	operation of the destination re- stination regis REG. If WREG	of the specif gister. The or ter. If WREG	ied file otional i is
	The 'D' bi	t selects the	destination (ration (0 for w 0 for WREG, 2 e file register.		
		rather than denote a w	a word operation	instruction de ation. You may n, but it is not rking register	/ use a . w ex required.	•
Words:	1					
Cycles:	1					
Example 1 XOR.B	0x1FFF		0x1FFF) an result to	d WREG (Byt 0x1FFF	e mode)	
	Before		After			
F	struction					
WREG (W0) Data 1FFE	7804 9439	WREG (WO Data 1FF	-			
SR	0000	S		N=1)		
Example 2 XOR	0xA04, N		R (0xA04) ore result	and WREG (W to WREG	lord mode)	
lr WREG (W0) Data 0A04 SR	Before Instruction 6234 A053 0000	WREG (W0 Data 0A0 Sf	4 A053	N=1)		

XOR		Exclusive	OR Literal a	and Wn		
Syntax:	{label:}	XOR{.B}	#lit10,	Wn		
Operands:		1023] for v	/te operation word operatic			
Operation:	lit10.XOR	$(Wn) \rightarrow Wn$				
Status Affected:	N, Z					
Encoding:	1011	0010	1Bkk	kkkk	kkkk	dddd
Description:	operand a	nd the conte	nts of the wo	operation of the rking register egister direct a	Wn and store	e the resul
	The 'k' bit	s specify the	literal opera	eration (0 for w nd. e working reg		te).
Words:	-	unsigned v eral Oper	alue [0:255]	he literal mu See Section information c	4.6 "Using	10-bit Li
Cycles:	1					
	DR.B #0x23, W	-	OR 0x23 an tore resul	d W0 (Byte t to W0	mode)	
	Before Instruction W0 7804 SR 0000		After Instruction /0 7827 R 0000			
Example 2 X	OR #0x108, V		OR 0x108 a tore resul	nd W4 (Word t to W4	mode)	
	Before Instruction W4 6134 SR 0000		After Instruction /4 603C R 0000			

XOR		Exclusive	OR Wb and	Short Litera	al	
Syntax:	{label:}	XOR{.B}	Wb,	#lit5,	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	lit5 ∈ [0	0 W15] . 31] 0 W15]				
Operation:	_	R.lit5 → Wd				
Status Affected:	N, Z					
Encoding:	0110	1www	wBqq	qddd	d11k	kkkk
Description:	register V the destir	Vb and the ur nation registe	isigned 5-bit r Wd. Regist	literal operan	he contents o d and place th essing must b may be used	ne result i be used fo
	The 'B' bi The 'q' bi The 'd' bi	ts select the o ts select the a ts provide the	e or word ope destination A address of th e literal opera	eration (0 for address mode he destination and, a 5-bit in	word, 1 for by e.	
	Note:	rather than	a word oper		ayusea .wie	
Words:	1					
Cycles:	1					
Example 1	XOR.B W4, #	0x16, W5		4 and 0x14 result to	(Byte mode W5	e)
	Before Instruction W4 C822 W5 1200 SR 0000	١	After Instruction N4 C822 N5 1234 SR 0000	n 		
Example 2	XOR W2, #	0x1F, [W8+	; Stor	W2 by 0x1F re result t -increment		e)
Data 1	Before Instruction W2 8505 W8 1004 004 6628 SR 0000	\ Data 10	After Instruction W2 8505 W8 1006 04 851A SR 0008	(N=1)		

XOR		Exclusive	OR Wb and V	Ws		
Syntax:	{label:}	XOR{.B}	Wb,	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wd [Wd] [Wd++] [Wd] [++Wd] [Wd]	
Operands:	Wb ∈ [W0 Ws ∈ [W0 Wd ∈ [W0) W15]				
Operation:	-	$R.(Ws) \rightarrow Wd$				
Status Affected:	N, Z	、 ,				
Encoding:	0110	1www	wBqq	qddd	dppp	SSSS
		ts select the a		-		
	The 'q' bit The 'd' bit The 'p' bit	s selects byte s select the d s select the a s select the s s select the a	estination Ad ddress of the ource Addres	dress mode destination s mode.	register.	rte).
	The 'q' bit The 'd' bit The 'p' bit	s select the d s select the a s select the s s select the a The extensi rather than	estination Ad ddress of the ource Addres ddress of the on .B in the	dress mode destination ss mode. source reg instruction tion. You m	e. register. ister. denotes a byte ay use a .we	e operatio
Words:	The 'q' bit The 'd' bit The 'p' bit The 's' bit	s select the d s select the a s select the s s select the a The extensi rather than	estination Ad ddress of the ource Addres ddress of the on .B in the a word opera	dress mode destination ss mode. source reg instruction tion. You m	e. register. ister. denotes a byte ay use a .we	e operatio
	The 'q' bit The 'd' bit The 'p' bit The 's' bit Note:	s select the d s select the a s select the s s select the a The extensi rather than	estination Ad ddress of the ource Addres ddress of the on .B in the a word opera	dress mode destination ss mode. source reg instruction tion. You m	e. register. ister. denotes a byte ay use a .we	e operatio
Words: Cycles: Example 1 XOR	The 'q' bit The 'd' bit The 'p' bit The 's' bit Note: 1	s select the d s select the a s select the s s select the a The extensi rather than	estination Ad ddress of the ource Address ddress of the on .B in the a word operation ord operation +] ; XOR ; Stor	dress mode destination s mode. source reg instruction tion. You m but it is not w1 and [W e result	e. register. ister. denotes a bytr ay use a . w e t required. 5] (Byte mo	e operatio xtension t
Cycles: Example 1 XOR	The 'q' bit The 'd' bit The 'p' bit The 's' bit Note: 1 1 2.B W1, [T Before	s select the d s select the a s select the s s select the a The extensi rather than denote a wo	estination Ad ddress of the ource Address ddress of the on .B in the a word operation ord operation +] ; XOR ; Stor ; Post After	Manual and [W1 and [W2 e result -increment	e. register. ister. denotes a bytr ay use a .w e t required. 5] (Byte mo to [W9]	e operatio xtension f
Cycles: Example 1 XOR	The 'q' bit The 'd' bit The 'p' bit The 's' bit Note: 1 1 2.B W1, [T Before Instruction	s select the d s select the a s select the s s select the a The extensi rather than denote a wo	estination Ad ddress of the ource Address ddress of the on .B in the a word operation ord operation +] ; XOR ; Stor ; Post After Instruction	Manual and [W1 and [W2 e result -increment	e. register. ister. denotes a bytr ay use a .w e t required. 5] (Byte mo to [W9]	e operatic xtension f
Cycles: Example 1 XOR W1	The 'q' bit The 'd' bit The 'p' bit The 's' bit Note: 1 1 2.B W1, [T Before Instruction AAAA	s select the d s select the a s select the s s select the a The extensi rather than denote a wo	estination Ad ddress of the ource Address ddress of the on .B in the a word opera- brd operation +] ; XOR ; Stor ; Post After Instruction /1 AAAA	Manual and [W1 and [W2 e result -increment	e. register. ister. denotes a bytr ay use a .w e t required. 5] (Byte mo to [W9]	e operatic xtension f
Cycles: Example 1 XOR W1 W5	The 'q' bit The 'd' bit The 'p' bit The 's' bit Note: 1 1 2.B W1, [T Before Instruction AAAA 2000	s select the d s select the a s select the s s select the a The extensi rather than denote a wo v5++], [w9+	estination Ad ddress of the ource Address ddress of the on . B in the a word opera- ord operation +1] ; XOR ; Stor ; Post After Instruction /1 AAAA /5 2001	Manual and [W1 and [W2 e result -increment	e. register. ister. denotes a bytr ay use a .w e t required. 5] (Byte mo to [W9]	e operatic xtension
Cycles: Example 1 XOR W1 W5 W9	The 'q' bit The 'd' bit The 'p' bit The 's' bit Note: 1 1 2.B W1, [T Before Instruction AAAA 2000	s select the d s select the a s select the s s select the a The extensi rather than denote a wo N5++], [W9+ W W W	estination Ad ddress of the ource Address ddress of the on . B in the a word operation a word operation (1 AAAA (5 2001 (9 2601	Manual and [W1 and [W2 e result -increment	e. register. ister. denotes a bytr ay use a .w e t required. 5] (Byte mo to [W9]	e operatic xtension f
Cycles: Example 1 XOR W1 W5	The 'q' bit The 'd' bit The 'p' bit The 's' bit Note: 1 1 2.B W1, [T Before Instruction AAAA 2000 2600 115A	s select the d s select the a s select the s s select the a The extensi rather than denote a wo v5++], [w9+	estination Ad ddress of the ource Address ddress of the on . B in the a word operation ord operation (1 AAAA /5 2001 /9 2601 00 115A	Manual and [W1 and [W2 e result -increment	e. register. ister. denotes a bytr ay use a .w e t required. 5] (Byte mo to [W9]	e operatic xtension f

Example 2	XOR	W1, W5,	, W9				(Word mode) lt to W9
	In	Before struction		Ir	After struction		
	W1	FEDC	۱. ۱	W1	FEDC		
	W5	1234	N N	W5	1234		
	W9	A34D	N N	W9	ECE8		
	SR	0000		SR	0008	(N=1)	

ZE		Zero-Exten	d Wn			
Syntax:	{label:}	ZE	Ws, [Ws], [Ws++], [Ws], [++Ws], [Ws],	Wnd		
Operands:	Ws ∈ [W0 Wnd ∈ [W0	-				
Operation:	Ws<7:0> $_{-}$ 0 \rightarrow Wnd<	→ Wnd<7:0> <15:8>				
Status Affected:	N, Z, C					
Encoding:	1111	1011	10qq	qddd	dppp	SSSS
	Wnd. Eithe and registe cleared an positive. The 'q' bits The 'd' bits	lue and store er register dire er direct addre d the C flag is select the de select the ac select the ac	ect or indirec essing must s set, becaus estination Ad ddress of the	t addressing be used for V e the zero-e> dress mode. destination r	may be used Vnd. The N fl stended word	l for Ws, lag is
	The 's' bits	select the ac This operation	ldress of the	source regis		sno.bc
	2:	. w extension The source address mod	Ws is addr		byte operan	id, so an
Words:	1					
Cycles:	1					
Example 1 ZE	W3, W4 ; ;	zero-exter Store rest				
	Before		After			
	Instruction		Instruction			
W3 W4		W3 W4				
SR		SR		;=1)		
Example 2 ZE	[W2++], W12	; Store t	ctend [W2] to W12 ncrement W	2		
W2 W12 Data 0900 SR	1002 268F	W2 W12 Data 0900 SR	008F 268F	:=1)		

6



Section 6. Reference

HIGHLIGHTS

This section of the manual contains reference information for the dsPIC30F. It consists of the following sections:

6.1	Data Memory Map	. 6-2
	Core Special Function Register Map	
6.3	Program Memory Map	. 6-6
	Instruction Bit Map	
6.5	Instruction Set Summary Table	. 6-9

6.1 Data Memory Map

A sample dsPIC30F data memory map is shown in Figure 6-1.

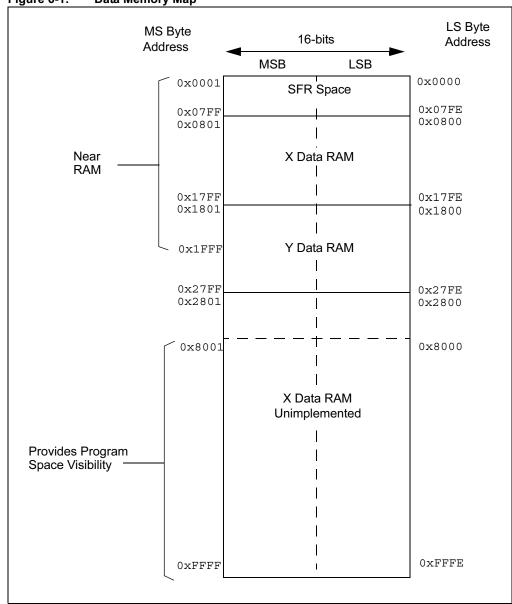


Figure 6-1: Data Memory Map

Note 1: The partition between the X and Y data spaces is device specific. Refer to the appropriate device data sheet for further details. The data space boundaries indicated here are for example purposes only.
2: Refer to Section 4. "Instruction Set Details" for information on Data Addressing

- modes, performing byte accesses and word alignment requirements.
- **3:** Refer to the dsPIC30F MCU Family Reference Manual for information on accessing program memory through data address space.

6.2 Core Special Function Register Map

The Core Special Function Register Map is shown in Table 6-1. Please refer to the dsPIC30F Data Sheet for complete register descriptions and the memory map of the remaining special function registers.

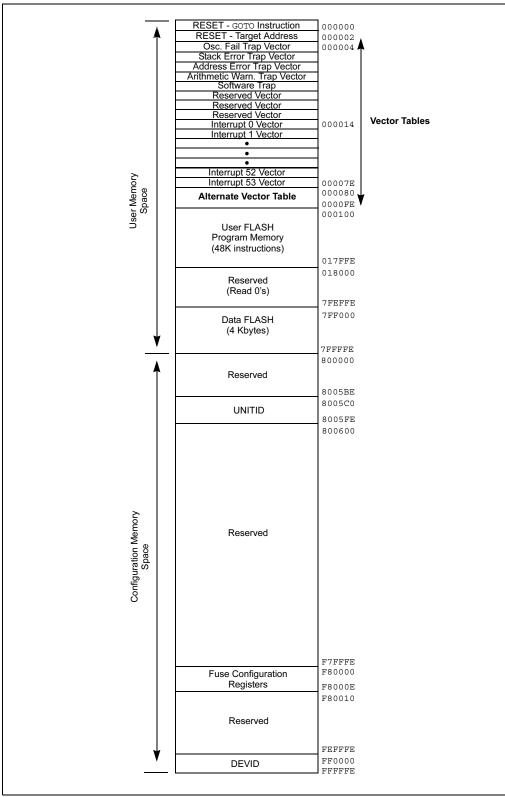
Addr Bits Bits <th< th=""><th>Table 6-1:</th><th></th><th>sPIC30F</th><th>dsPIC30F Core Register Map</th><th>gister N</th><th>lap</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></th<>	Table 6-1:		sPIC30F	dsPIC30F Core Register Map	gister N	lap												
000 000 000 0001 VIII VIII 0002 VIII VIII 0003 VIII VIII 0004 VIII VIII 0005 VIII VIII 0006 VIII VIII 0007 VIII VIII 0008 VIII VIII 0009 VIII VIII 0010 VIII VIII 0011 VIII VIII 0012 VIIII VIII 0013 VIIII VIIII 0014 VIIII VIIII 0015 VIIII VIIII 0016 VIIII VIIII 0017 VIIII VIIII 0018 VIIII VIIII 0016 VIIII VIIII 0016 VIIII VIIII 0020 VIIII VIIII 0020 VIIII VIIII 0020 VIIII VIIII	Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12			Bit	Bit 8		Bit 6	Bit 5			2	Bit 0	RESET State
000 001 002 000 001 001 000 001 001 000 001 001 000 001 001 001 001 001 001 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 001 001 0010 Internet 001 0010 Internet 001 002 Internet 002 002 Internet 002 002 <td< td=""><td>MO</td><td>0000</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>) 0M</td><td>WREG)</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0000 0000 0000 0000</td></td<>	MO	0000) 0M	WREG)							0000 0000 0000 0000
000	W1	0002								-	W1							0000 0000 0000 0000
006 007 0008 0004 0009 0004 0001 0005 0010 001 0010 001 0010 001 0010 0014 0010	W2	0004								-	W2							0000 0000 0000 0000
008 ····································	W3	0006								-	W3							0000 0000 0000 0000
000 000 <td>W4</td> <td>0008</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>-</td> <td>W4</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0000 0000 0000 0000</td>	W4	0008								-	W4							0000 0000 0000 0000
0000 Not Not <td>W5</td> <td>000A</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>W5</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0000 0000 0000 0000</td>	W5	000A									W5							0000 0000 0000 0000
0000 0000 <th< td=""><td>W6</td><td>000C</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>-</td><td>WG</td><td></td><td></td><td></td><td></td><td></td><td></td><td>0000 0000 0000 0000</td></th<>	W6	000C								-	WG							0000 0000 0000 0000
010 ····································	W7	000E									W7							0000 0000 0000 0000
0012 901 901 0018	W8	0010									W8							0000 0000 0000 0000
011 111 <td>6M</td> <td>0012</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>6M</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0000 0000 0000 0000</td>	6M	0012									6M							0000 0000 0000 0000
010 011 <td>W10</td> <td>0014</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>~</td> <td>V10</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>0000 0000 0000 0000</td>	W10	0014								~	V10							0000 0000 0000 0000
	W11	0016								~	V11							0000 0000 0000 0000
	W12	0018								~	V12							0000 0000 0000 0000
	W13	001A								~	V13							0000 0000 0000 0000
	W14	001C								~	V14							0000 0000 0000 0000
0020 SPIIM 0023 ACCAL 0024 ACCAL 0026 ACCAL 0028 ACCAL 0029 ACCAL 0020 ACCAL 0030 AC 0031 AC 0032 CH 0033 CH 0034 AC 0035 CH 0036 AC 0037 CH 0038 AC 0039 AC 0030 AC 0030 AC 0030 AC 0030 AC AC	W15	001E								1	V15							0000 1000 0000 0000
002 ACCAL 002	SPLIM	0020								S	MIJc							0000 0000 0000 0000
0024 ACCAH 0026 Sign-extension of ACCA<39> ACCAH ACCAH 0028	ACCAL	0022								AC	CAL							0000 0000 0000 0000
0026Sign-extension of ACCA<39-Sign-extension of ACCA<39-ACCBL0028 0.22 $$	ACCAH	0024								AC	CAH							0000 0000 0000 0000
0028 ····································	ACCAU	0026			Sign-e	*xtension	of ACCA	<39>						ACCAU				0000 0000 0000 0000
002A $002A$ $ACCBH002C002CSign-xtension of ACCB-39 ACCBH002002Sign-xtension of ACCB-39 ACCBH00300P-00310P-0032013210034100341003610038100381003810038100381003810038100381003811111<$	ACCBL	0028								AC	CBL							0000 0000 0000 0000
	ACCBH	002A								AC	CBH							0000 0000 0000 0000
002E PCI 0030 PCI AG 0032 PC PCI AG 0032 PCI TBLPAG AG 0034 TBLPAG AG 0034 TBLPAG AG 0034 TBLPAG AIT 0036 TBLPAG AIT 0038 TBLPAG AIT 0038 TBLPAG AIT 0038 DOSTAT AIT 036 DOSTAT AI	ACCBU	002C			Sign-e	xtension	of ACCB	<39>						ACCBU				0000 0000 0000 0000
0030	PCL	002E								ł	CL							0000 0000 0000 0000
0032 TBLPAG 0034 TBLPAG 0034 TBLPAG 0034 PSVPG 0036 PSVPG 0038 PSVPG 1 0038 PSVPG 1 0036 PSVPG 1 0036 PSVPG 1 0036 PSVPG	PCH	0030												P(Н			0000 0000 0000 0000
0034 PSVPAG 0036 - - - - PSVPAG 0036 - - - - - PSVPAG 1 0038 - - - - - - PSVPAG 1 0038 - - - - - - PSVPAG 1 0036 - - - - - - PSVPAG 1 0036 - - - - - - - PSVPAG 0040 - - - - - - - - - - - - - - - PSVPAG	TBLPAG	0032												TBLPAG				0000 0000 0000 0000
0036 RCOUNT 1 0038 1 003 1 003 1 003 1 003 1 003 1 003 1 003 1 003 1 I I 1 I I 1 I I 1 I I 1 I I 1 I I 1 I I	PSVPAG	0034												PSVPAG				0000 0000 0000 0000
0038 DCOUNT L 003A DCOUNT H 003C DOSTARTL O 003C DOSTARTL N 003C N 003C N 003C N 0040	RCOUNT	0036								RC	OUNT							XXXX XXXX XXXX XXXX
I 003A DOSTARTL H 003C 003E 0040	DCOUNT	0038								DC	OUNT							XXXX XXXX XXXX XXXX
H 003C - - - - - 003E - - - - - - 0040 - - - - - -	DOSTARTL	003A								DOS	TARTL							XXXX XXXX XXXX XXXX
003E DOENDL	DOSTARTH	003C												1	DOSTARTH	-		0000 0000 00xx xxxx
	DOENDL	003E								DO	ENDL							XXXX XXXX XXXX XXXX
-	DOENDH	0040			Ι	Ι		Ι	Ι	Ι					DOENDH			0000 0000 00XX XXXX

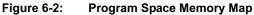
Table 6-1:		dsPIC30F Core Register Map (Continue	Core Rec	jister M	ap (Cor	ntinued)												
Name	Addr	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	RESET State
SR	0042	AO	OB	SA	SB	OAB	SAB	DA	БС	IPL2	IPL1	IPL0	RA	z	20	Z	υ	0000 0000 0000 0000
CORCON	0044				SN	EDT	DL2	DL1	DLO	SATA	SATB	SATDW	ACCSAT	IPL3	PSV	RND	Ε	0000 0000 0010 0000
MODCON	0046	XMODEN	YMODEN				BWM<3:0>	<3:0>			ΥWM	YWM<3:0>			XWM<3:0>	<3:0>		0000 0000 0000 0000
XMODSRT	0048								XMODSRT<15:0>	T<15:0>								XXXX XXXX XXXX XXXX
XMODEND	004A								XMODEND<15:0>	D<15:0>								XXXX XXXX XXXX XXXX
YMODSRT	004C								YMODSRT<15:0>	T<15:0>								XXXX XXXX XXXX XXXX
YMODEND	004E							-	YMODEND<15:0>	D<15:0>								XXXX XXXX XXXX XXXX
XBREV	0020	BREN							XE	XBREV<14:0>	<(XXXX XXXX XXXX XXXX
DISICNT	0052									DISICN	DISICNT<13:0>							0000 0000 0000 0000
Reserved	0054 - 007E						I		ļ					I			I	0000 0000 0000

ntinue	
<u>c</u>	
r Map	
giste	
e Re	
Cor	
PIC30F	
ds	
6-1:	

6.3 Program Memory Map

A sample dsPIC30F program memory map is shown in Figure 6-2.





6.4 Instruction Bit Map

Instruction encoding for the dsPIC30F is summarized in Table 6-2. This table contains the encoding for the Most Significant Byte of each instruction. The first column in the table represents bits 23:20 of the opcode, and the first row of the table represents bits 19:16 of the opcode. The first byte of the opcode is formed by taking the first column bit value and appending the first row bit value. For instance, the Most Significant Byte of the PUSH instruction (last row, ninth column) is encoded with 11111000b (0xF8).

Note: The complete opcode for each instruction may be determined by the instruction descriptions in **Section 5. "Instruction Descriptions"**, using Table 5.2 through Table 5-12.

	TIII	BRA (SB)										BTSC	NOM	FF1L FF1R	FBCL	CLR SETM	NOPR
	0111	BRA (SA)			BRA (GTU)							BTSS	MOV.D	I	ASR LSR	COM NEG	CLRWDT PWRSAV POP.S PUSH.S RESET
	TOTT	BRA (OB)			BRA (GE)							BSW	SUB SUBB	SAC.R	SL	DEC DEC2	DAW EXCH SWAP
	1100	BRA (OA)	3R		BRA (GT)	Q	B	æ	>			BTSTS	MUL	SAC	-	INC INC2	DISI
	1011	1	SUBBR		BRA (NN)	ADDC	SUBB	XOR	MOV			BTST	ТВLWTL ТВLWTH	ADD NEG SUB		CLR SETM	SE ZE
	1010	1			(ZN)							BTG	TBLRDL TBLRDL	LAC	-	COM NEG	LNK
	1001	REPEAT			BRA (NC)							BCLR	MUL.SS MUL.SU	ADD	DIVF	DEC DEC2	POP
	1000	Od			BRA (NOV)							BSET	MUL.US MUL.US	SFTAC	DIV.S DIV.U	INC INC2	HSUA
	1110	RCALL		NOM	BRA					MOV	MOV	BTSC	IOR MOV	MOVSAC	RRC RRNC	CPSEQ CPSNE	
	0110	RETFIE RETURN			BRA (LEU)							BTSS	AND XOR		RLC RLNC	CPSGT CPSLT	
	1010	RETLW			BRA (LT)							BTST	SUB SUBB	MAC MPY MSC	ASR LSR	I	
	0010	GOTO	ЗR		BRA (LE)	Q	В	Q	Ъ			BTSTS	ADD ADDC		SL	I	
incoding	1100	I	SUBR		BRA (N)	ADD	SUB	AND	IOR			BTST	IOR MOV	CLRAC	RRC RRNC	CP CPB	
truction E	0100	CALL			BRA (Z)							BTG	AND XOR		RLC RLNC	CP0	_ ۷۷≻
dsPIC30F Instruction Encoding	1000	BRA CALL GOTO RCALL			BRA (C)							BCLR	SUB SUBB	MAC MPY MSC	ASR LSR	CPB CPB	ED EDAC MPY
	0000	NOP			BRA (OV)							BSET	ADD ADDC		SL	СРО	
Table 6-2:	bit 23 bit 19	0000	1000	0010	TTOO	0010	1010	0110	TTT0	1000	1001	1010	TIOI	0011	1011	0111	IIII

6.5 Instruction Set Summary Table

The complete dsPIC30F instruction set is summarized in Table 6-3. This table contains an alphabetized listing of the instruction set. It includes instruction assembly syntax, description, size (in 24-bit words), execution time (in instruction cycles), affected status bits and the page number in which the detailed description can be found. Table 1-2 identifies the symbols which are used in the Instruction Set Summary Table.

	Assembly Syntax Mnemonic,Operands	Description	Words	Cycles	AO	OB	SA	SB	OAB	SAB	g	z	2	N	ပ	Page #
ADD	f {,WREG}	Destination = f + WREG	-	-				I			⇔	⇔	⇔	⇔	⇔	2-2
ADD	#lit10,Wn	Wn = lit10 + Wn	-	4						I	⇔	⇔	€	€	⇔	5-8
ADD	Wb,#lit5,Wd	Wd = Wb + lit5	٦	4	I	I				I	€	¢	Ŷ	¢	Ŷ	6-3
ADD	Wb,Ws,Wd	SW + dW = bW	-	-	Ι		Ι		Ι	I	⇔	⇔	¢	€	⇔	5-10
ADD	Acc	Add accumulators	1	1	¢	Ŷ	Ų	Ų	Ŷ	Û					I	5-11
ADD	Ws,#Slit4,Acc	16-bit signed add to accumulator	÷	،	€	ţ	Ŷ	Ŷ	€	Ŷ	I			1	I	5-12
ADDC	f {,WREG}	Destination = f + WREG + (C)	-	-	1	1		I	I	I	⇔	⇔	≎	⇔	⇔	5-14
ADDC	#lit10,Wn	Wn = lit10 + Wn + (C)	÷	-						I	⇔	⇔	⇔	⇔	⇔	5-15
ADDC	Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	-	-			Ι			I	⇔	⇔	€	₽	⇔	5-16
ADDC	Wb,Ws,Wd	Wd = Wb + Ws + (C)	-	-	I		Ι			I	⇔	⇔	¢	⇔	⇔	5-17
AND	f {,WREG}	Destination = f .AND. WREG	1	1	Ι							Û		Û		5-19
AND	#lit10,Wn	Wn = lit10 .AND. Wn	٢	1			Ι			I	I	Û		€	I	5-20
AND	Wb,#lit5,Wd	Wd = Wb .AND. lit5	-	-	I		Ι			I	I	⇔		€	I	5-21
AND	Wb,Ws,Wd	Wd = Wb .AND. Ws	-	-			Ι	I	I	I	I	⇔		€	I	5-22
ASR	f {,WREG}	Destination = arithmetic right shift f	1	1						I		Ŷ		¢	Ŷ	5-24
ASR	Ws,Wd	Wd = arithmetic right shift Ws	٢	-	I		Ι	I	I	I	I	Û	I	€	Ŷ	5-25
ASR	Wb,#lit4,Wnd	Wnd = arithmetic right shift Wb by lit4	١	1			Ι			I	I	Ŷ		¢	I	5-27
ASR	Wb,Wns,Wnd	Wnd = arithmetic right shift Wb by Wns	٢	1			Ι			I	I	Û		€	I	5-28
BCLR	f,#bit4	Bit clear f	1	1	I		Ι	Ι	Ι	I	I		I	1		5-29
BCLR	Ws,#bit4	Bit clear Ws	٢	1				Ι		I		Ι	Ι			5-30
BRA	Expr	Branch unconditionally	٢	2	Ι	I		Ι		I		Ι	I			5-31
BRA	Wn	Computed branch	-	2		Ι	Ι	Ι	Ι	Ι	Ι	Ι				5-32
BRA	C,Expr	Branch if Carry	٢	1 (2)	Ι			Ι		Ι						5-33
BRA	GE, Expr	Branch if greater than or equal	۲	1 (2)		I					I			Ι		5-35
BRA	GEU,Expr	Branch if Carry	-	1 (2)	Ι											5-36
BRA	GT,Expr	Branch if greater than	-	1 (2)	Ι			Ι		Ι		Ι	Ι			5-37
BRA	GTU,Expr	Branch if unsigned greater than	-	1 (2)				Ι								5-38
BRA	LE,Expr	Branch if less than or equal	-	1 (2)	Ι	I		I	I	Ι		I	I			5-39
BRA	LEU,Expr	Branch if unsigned less than or equal	-	1 (2)	Ι			Ι		Ι		Ι	Ι			5-40
BRA	LT,Expr	Branch if less than	۲	1 (2)	Ι	I		Ι	I	Ι			Ι			5-41
BRA	LTU,Expr	Branch if not Carry	-	1 (2)	Ι	I		Ι	I						I	5-42
BRA	N, Expr	Branch if Negative	-	1 (2)	I	I		Ι		I			I			5-43
BRA	NC,Expr	Branch if not Carry	-	1 (2)	Ι						Ι					5-44
BRA	NN,Expr	Branch if not Negative	1	1 (2)	I			Ι	Ι	Ι		I	Ι			5-45

		et summary lable (continued)												_		
	Assembly Syntax Mnemonic,Operands	Description	Words	Cycles	AO	OB	SA	SB	OAB	SAB	Ы	z	δ	z	ပ	Page #
BRA	NOV,Expr	Branch if not Overflow	۲	1 (2)	1			1			1	1		1	-	5-46
BRA	NZ,Expr	Branch if not Zero	1	1 (2)	I		I									5-47
BRA	OA,Expr	Branch if Accumulator A overflow	1	1 (2)	I	I	I			I			I	I		5-48
BRA	OB,Expr	Branch if Accumulator B overflow	1	1 (2)	Ι		Ι									5-49
BRA	OV,Expr	Branch if Overflow	1	1 (2)	I		I									5-50
BRA	SA,Expr	Branch if Accumulator A saturated	1	1 (2)			I								-	5-51
BRA	SB,Expr	Branch if Accumulator B saturated	£	1 (2)	I	I	I			I			I	Ι		5-52
BRA	Z, Expr	Branch if Zero	-	1 (2)		I	I			Ι			1	Ι		5-53
BSET	f,#bit4	Bit set f	-	-	I	I	I			I			1	I	-	5-54
BSET	Ws,#bit4	Bit set Ws	Ł	-	Ι	Ι	Ι			Ι			Ι	Ι		5-55
BSW.C	Ws,Wb	Write C bit to Ws <wb></wb>	1	1	I		I									5-56
BSW.Z	Ws,Wb	Write Z bit to Ws <wb></wb>	1	1	I		I									5-56
BTG	f,#bit4	Bit toggle f	1	1			I									5-58
BTG	Ws,#bit4	Bit toggle Ws	1	۲			I								-	5-59
BTSC	f,#bit4	Bit test f, skip if clear	١	1 (2 or 3)			I							I		5-60
BTSC	Ws,#bit4	Bit test Ws, skip if clear	1	1 (2 or 3)	I		I									5-62
BTSS	f,#bit4	Bit test f, skip if set	1	1 (2 or 3)	Ι		I									5-64
BTSS	Ws,#bit4	Bit test Ws, skip if set	1	1 (2 or 3)	I		I									5-65
BTST	f,#bit4	Bit test f	1	1			I	I	I		I	I		€		5-67
BTST.C	Ws,#bit4	Bit test Ws to C	1	1			I		1					I	; \$	5-68
BTST.Z	Ws,#bit4	Bit test Ws to Z	1	1			I							⇔	-	5-68
BTST.C	Ws,Wb	Bit test Ws <wb> to C</wb>	1	1			I							Ι	i t	5-69
BTST.Z	Ws,Wb	Bit test Ws <wb> to Z</wb>	-	-			I	I			1			⇔		5-69
BTSTS	f,#bit4	Bit test then set f	1	1			I							⇔	-	5-71
BTSTS.C	Ws,#bit4	Bit test Ws to C then set	1	1			I		Ι		Ι	Ι		I	; \$	5-72
BTSTS.Z	Ws,#bit4	Bit test Ws to Z then set	1	1			I		1					≎	-	5-72
CALL	Expr	Call subroutine	2	2	I	I	1	Ι	1	1	1	1	1	1	-	5-73
CALL	Wn	Call indirect subroutine	1	2	Ι		Ι							Ι		5-74
CLR	f	$f = 0 \times 0000$	-	-	Ι	Ι	Ι			Ι			Ι	Ι		5-75
CLR	WREG	WREG = 0×0000	-	-	Ι		Ι									5-75
CLR	Wd	Wd = 0	1	1	Ι		Ι							Ι		5-76
CLR	Acc,Wx,Wxd,Wy,Wyd,AWB	Clear Accumulator	-	-	0	0	0	0	0	0			Ι	Ι		5-77
CLRWDT		Clear Watchdog Timer	-	-	Ι		Ι	Ι	Ι	Ι		1	Ι	Ι		5-79
COM	f {,WREG}	Destination = \overline{f}	٦	٦			Ι	Ι	Ι		Ι	€		€	-	5-80
COM	Ws,Wd	Wd = WS	٢	٦				Ι	Ι		Ι	€		€	-	5-81
Legend: Note:	 set or cleared; may be cleared, bu SA, SB and SAB are only modified if 	③ set or cleared; ¹ <i>may</i> be cleared, but never set; ¹ <i>may</i> be set, but never cleared; ¹ ¹ always set; SA, SB and SAB are only modified if the corresponding saturation is enabled, otherwise unchanged.	'1' always set; ise unchanged.		'0' always cleared;		— unchanged	anged								

Table 6-3: dsPIC30F Instruction Set Summary Table (Continued)

	Assembly Syntax Mnemonic,Operands	Description	Words	Cycles	AO	OB	SA	SB	OAB	SAB	БС	z	٥	N	ပ	Page #
СР		Compare (f – WREG)	-	-							⇔	⇔	⇔	⇔	⇔	5-82
СР	Wb,#lit5	Compare (Wb – lit5)	-	-							€	¢	¢	€	⇔	5-83
СР	Wb,Ws	Compare (Wb – Ws)	-	.						1	⇔	¢	¢	⇔	⇔	5-84
CP0	Ļ	Compare (f – 0x0000)	÷	.		1					⇔	≎	¢	≎	⇔	5-85
CP0	Ws	Compare (Ws – 0x0000)	-	.							€	¢	¢	€	⇔	5-86
CPB	Ļ	Compare with borrow $(f - WREG - \overline{C})$	-	.		1	1	I	I	I	⇔	≎	¢	⇔	⇔	5-87
СРВ	Wb,#lit5	Compare with borrow (Wb – lit5 – \overline{C})	÷	-	1	1		1	1	1	≎	¢	¢	₽	⇔	5-88
СРВ	sw,dw	Compare with borrow (Wb – Ws – \overline{C})	÷	-	1	1		I	I	1	≎	¢	¢	₽	⇔	5-89
CPSEQ	Wb, Wn	Compare (Wb with Wn), skip if =	۲	1 (2 or 3)	1		1	1	1	I		1	1	1	1	5-91
CPSGT	Wb, Wn	Signed Compare (Wb with Wn), skip if >	-	1 (2 or 3)	Ι		1	I	I	I		1	1	I	1	5-92
CPSLT	Wb, Wn	Signed Compare (Wb with Wn), skip if <	1	1 (2 or 3)												5-93
CPSNE	Wb, Wn	Signed Compare (Wb with Wn), skip if ≠	1	1 (2 or 3)	I		I	I	I	Ι					I	5-94
DAW.B	Wn	Wn = decimal adjust Wn	-	-		I		I	I	I					⇔	5-95
DEC	f {,WREG}	Destination = $f - 1$	٢	٢							¢	Ŷ	¢	Ŷ	⇔	5-96
DEC	Ws,Wd	Wd = Ws - 1	-	-						1	⇔	¢	€	⇔	⇔	5-97
DEC2	f {,WREG}	Destination = $f - 2$	٢	٢		1					€	Ŷ	€	Ŷ	⇔	5-98
DEC2	Ws,Wd	Wd = Ws - 2	٢	٢							¢	¢	¢	Ŷ	⇔	5-99
DISI	#lit14	Disable interrupts for lit14 instruction cycles	1	-			1	1	1	1		1	1	1		5-100
DIV.S	Wm, Wn	Signed 16/16-bit integer divide	٦	18		I	I	I	I	I	I	¢	€	Û	⇔	5-101
DIV.SD	Wm, Wn	Signed 32/16-bit integer divide	١	18		I		I	I	I		¢	¢	Ŷ	⇔	5-101
DIV.U	Wm, Wn	Unsigned 16/16-bit integer divide	٢	18		1						0	¢	Ŷ	⇔	5-103
DIV.UD	Wm, Wn	Unsigned 32/16-bit integer divide	٢	18								0	€	Ŷ	⇔	5-103
DIVF	Wm, Wn	Signed 16/16-bit fractional divide	٢	18								Ŷ	¢	Ŷ	⇔	5-105
DO	#lit14, Expr	Do code to PC+Expr, (lit14+1) times	2	2					I							5-107
DO	Wn, Expr	Do code to PC+Expr, (Wn+1) times	2	2	Ι		Ι	Ι	Ι	Ι		Ι	Ι	Ι	Ι	5-109
ED	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance (no accumulate)	-	-	0	0	1	Ι	0			1		I		5-111
EDAC	Wm*Wm,Acc,Wx,Wy,Wxd	Euclidean distance	1	-	⇔	¢	Û	Ŷ	⇔	Ŷ	Ι			Ι		5-113
EXCH	Wns,Wnd	Swap Wns and Wnd	1													5-115
FBCL	Ws,Wnd	Find bit change from left (MSb) side	1	1											¢	5-116
FF1L	Ws,Wnd	Find first one from left (MSb) side	٦	-	I	I	I	I	I	I	I			I	⇔	5-118
FF1R	Ws,Wnd	Find first one from right (LSb) side	٢	٢											⇔	5-120
GOTO	Expr	Go to address	2	2		Ι										5-122
GOTO	Wn	Go to address indirectly	-	2												5-123

DS70030E-page 6-12

Advance Information

© 2003 Microchip Technology Inc.

lable 6-3	3: dsPIC30F Instruction Set Summary 1ab	et summary lable (continued)										-		-		
	Assembly Syntax Mnemonic,Operands	Description	Words	Cycles	AO	ОВ	SA	SB	OAB	SAB	Ы	z	ò	N	υ	Page #
INC	f{,WREG}	Destination = f + 1	1	1			Ι	Ι	1		Û	Û	¢	Û	; \$}	5-124
INC	Ws,Wd	Wd = Ws + 1	1	1			I				¢	Ŷ	€	Ŷ	; \$	5-125
INC2	f {,WREG}	Destination = $f + 2$	1	1			I				¢	Ŷ	€	Ŷ	; \$	5-126
INC2	Ws,Wd	Wd = Ws + 2	1	1							Û	Û	¢	Û	; \$	5-127
IOR	f {,WREG}	Destination = f .IOR. WREG	1	1	I		I	I	I		I	¢		ţ		5-128
IOR	#lit10,Wn	Wn = lit10 .IOR. Wn	1	1		I	Ι	Ι				Û		Û		5-129
IOR	Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	٦			I	I				Ŷ		Ŷ		5-130
IOR	Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	٦	I	I	I	I				Ŷ		Ŷ		5-131
LAC	Ws,#Slit4, Acc	Load Accumulator	1	1	¢	€	Ŷ	Ŷ	Ŷ	Ŷ						5-133
LNK	#lit14	Link frame pointer	1	1	I				I	I		1		I		5-135
LSR	f {,WREG}	Destination = logical right shift f	1	1	I			I	I			Ŷ		ţ	€	5-136
LSR	Ws,Wd	Wd = logical right shift Ws	-	-	I	I	I	I	1	1		⇔		⇔	⇔	5-137
LSR	Wb,#lit4,Wnd	Wnd = logical right shift Wb by lit4	-	-	I		I					€		⇔		5-139
LSR	Wb,Wns,Wnd	Wnd = logical right shift Wb by Wns	1	٢			Ι	Ι				Ŷ		Ŷ		5-140
MAC	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd,AWB	Multiply and accumulate	1	٢	Ŷ	¢	Û	Ŷ	Ŷ	Ŷ						5-141
MAC	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd,	Square and accumulate	1	٦	¢	€	Ŷ	Ŷ	Ŷ	Ŷ						5-143
MOV	f{,WREG}	Move f to destination	1	1			I					Ŷ		Ŷ		5-145
MOV	WREG,f	Move WREG to f	1	1												5-146
MOV	f,Wnd	Move f to Wnd	1	1				I		Ι	I					5-147
MOV	Wns,f	Move Wns to f	1	۲				I								5-148
MOV.B	#lit8,Wnd	Move 8-bit unsigned literal to Wnd	1	1				Ι								5-149
MOV	#lit16,Wnd	Move 16-bit literal to Wnd	1	1	I	Ι	I	Ι	I	Ι	I					5-150
MOV	[Wns+Slit10],Wnd	Move [Wns + Slit10] to Wnd	1	1		Ι	I	Ι	I		I					5-151
MOV	Wns,[Wnd+Slit10]	Move Wns to [Wnd + Slit10]	4	+	Ι	Ι		Ι	Ι	Ι	1					5-152
MOV	Ws,Wd	Move Ws to Wd	1	1	I	Ι	I	Ι	I	Ι	I					5-153
MOV.D	Ws,Wnd	Move double Ws to Wnd:Wnd+1	1	2		I		I								5-155
MOV.D	Wns,Wd	Move double Wns:Wns+1 to Wd	1	2	-	Ι	I									5-157
MOVSAC	Acc,Wx,Wxd,Wy,Wyd,AWB	Move [Wx] to Wxd, and [Wy] to Wyd	1	1				Ι								5-159
МРҮ	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wn by Wm to accumulator	1	1	0	0		Ι	0	Ι		I				5-161
МРΥ	Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square to Accumulator	1	1	0	0	I	Ι	0	I						5-163
MPY.N	Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wn by Wm) to Accumulator	1	1	0	0		Ι	0	Ι						5-165
MSC	Wm*Wn,Acc,Wx,Wxd,Wyd,AWB	Multiply and subtract from Accumulator	1	1	¢	⇔	¢	Û	€	Û						5-167
MUL	f	W3:W2 = f * WREG	-	-	Ι	I		Ι	Ι	Ι		I				5-169
MUL.SS	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * sign(Ws)	-	-	Ι	Ι	Ι	Ι	Ι	Ι	1	Ι		Ι		5-170
Legend: Note:	 the set or cleared; ↓ may be cleared, bu SA, SB and SAB are only modified if if 	set or cleared; $ end{tabular}$ be cleared, but never set; $ end{tabular}$ may be set, but never cleared; '1' always set; SA, SB and SAB are only modified if the corresponding saturation is enabled, otherwise unchanged	'⊥' always set; ise unchanged.		'0' always cleared;		— unchanged	anged								

Table 6-3: dsPIC30F Instruction Set Summary Table (Continued)

	Assembly Syntax Mnemonic,Operands	Description	Words	Cycles	OA	OB	SA	SB	OAB	SAB	DC	z	٥v	z	C	Page #
MUL.SU	Wb,#lit5,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(lit5)	1	-			I	I							- 5	5-172
MUL.SU	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(Ws)	1	1		Ι			1						- 5	5-174
MUL.US	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * sign(Ws)	1	-											- 2	5-176
Μυμ.υυ	Wb,#lit5,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(lit5)	1	-	I	I	I	I							- 5	5-178
MUL.UU	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(Ws)	1	-		I	I	I							- 5	5-179
NEG	f {,WREG}	Destination = $\overline{f} + 1$	٦	٢							¢	¢	Ŷ	¢	÷ 5	5-181
NEG	Ws,Wd	Wd = Ws + 1	-	-	1		I	I	1	1	€	⇔	€	⇔	÷ 5	5-182
NEG	Acc	Negate Accumulator	۲	-	⇔	⇔	Ŷ	Ŷ	⇔	Ŷ		1			1	5-183
NOP		No operation	-	-					1	1					1	5-184
NOPR		No operation	-	-	1	1	I	I	I						1	5-185
РОР	f	Pop TOS to f	٢	-	1		I	I	1						- 5	5-186
РОР	Md	Pop TOS to Wd	۲	-	I	I	I	I	I						2	5-187
POP.D	Wnd	Pop double from TOS to Wnd:Wnd+1	1	2			I	I							- 5	5-188
POP.S		Pop shadow registers	1	٢		Ι			Ι		€	¢	Ŷ	€	Û 5	5-189
PUSH	f	Push f to TOS	٢	-	1	1	I	I	1						- 2	5-190
PUSH	Ws	Push Ws to TOS	1	1		Ι	I	I							- 5	5-191
PUSH.D	Wns	Push double Wns:Wns+1 to TOS	1	2	I	Ι	Ι	Ι							- 5	5-192
PUSH.S		Push shadow registers	1	٢	Ι	Ι	Ι	Ι							- 2	5-193
PWRSAV	#lit1	Enter Power Saving mode	1	1	Ι	Ι	Ι	Ι							- 2	5-194
RCALL	Expr	Relative call	1	2	Ι	Ι	I	I	Ι						- 5	5-195
RCALL	Wn	Computed call	1	2	Ι	Ι	Ι	Ι							- 2	5-196
REPEAT	#lit14	Repeat next instruction (lit14+1) times	1	1	Ι	Ι	Ι	Ι							- 2	5-197
REPEAT	Wn	Repeat next instruction (Wn+1) times	1	٢		Ι	I	I	Ι						- 5	5-198
RESET		Software device RESET	1	٢	Ι	Ι	Ι	Ι							- 2	5-200
RETFIE		Return from interrupt enable	-	3 (2)	I	I	I	I	I	I		⇔	⇔	⇔	5 \$	5-201
RETLW	#lit10, Wn	Return with lit10 in Wn	1	3 (2)			I	I							- 5	5-202
RETURN		Return from subroutine	1	3 (2)	Ι	Ι	Ι	Ι	Ι						- 5	5-203
RLC	f {,WREG}	Destination = rotate left through Carry f	-	-					I		I	⇔		⇔	5 ()	5-204
RLC	Ws,Wd	Wd = rotate left through Carry Ws	1	٢	I	I	I	I	I		I	¢		€	€ Ş	5-205
RLNC	f {,WREG}	Destination = rotate left (no Carry) f	-	-								⇔		€	- 2	5-207
RLNC	Ws,Wd	Wd = rotate left (no Carry) Ws	-	-	1							⇔		€	- 2	5-208
RRC	f {,WREG}	Destination = rotate right through Carry f	-	-	1		I	I	1			⇔		⇔	÷ 5	5-210
RRC	Ws,Wd	Wd = rotate right through Carry Ws	-	-		I	I	I				⇔		⇔	÷ 5	5-211
RRNC	f {,WREG}	Destination = rotate right (no Carry) f	1	-								¢		¢	- 5	5-213
RRNC	Ws,Wd	Wd = rotate right (no Carry) Ws	1	-								€		Ŷ	- 5	5-214

												-	╞		-	
	Assembly Syntax Mnemonic,Operands	Description	Words	Cycles	AO	OB	SA	SB	OAB	SAB	Ы	z	2	о N	ь С	Page #
SAC	Acc,#Slit4,Wd	Store Accumulator	-	1		I									- 5-2	5-216
SAC.R	Acc,#Slit4,Wd	Store rounded Accumulator	1	1		Ι									- 5-2	5-218
SE	Ws,Wd	Wd = sign-extended Ws	1	1					Ι			¢		ţ \$	Û 5-2	5-220
SETM	f	f = 0×FFFF	1	-		I									- 5-2	5-221
SETM	WREG	WREG = 0xFFFF	1	1	I	I		I	I	I				-	- 5-2	5-221
SETM	Ws	WS = 0xFFFF	٢	1					I						- 2-3	5-222
SFTAC	Acc,#Slit6	Arithmetic shift accumulator by Slit6	1	1	Ŷ	Ŷ	Û	Û	€	Û				-	- 2-5	5-223
SFTAC	Acc,Wn	Arithmetic shift accumulator by (Wn)	-	-	¢	⇔	¢	Ŷ	⇔	Ŷ				' 	- 2-5	5-224
SL	f {,WREG}	Destination = arithmetic left shift f	٦	1			Ι	Ι	I			€		ţ \$	¢ 2-3	5-225
SL	Ws,Wd	Wd = arithmetic left shift Ws	-	-			Ι	Ι	Ι		1	⇔		ţ ţ	⊕ 2-5	5-226
SL	Wb,#lit4,Wnd	Wnd = left shift Wb by lit4	-	-	I	I	I		I		1	⇔		- \$	- 5-2	5-228
SL	Wb,Wns,Wnd	Wnd = left shift Wb by Wns	٦	1			Ι	I	I			€		- \$	- 2-2	5-229
SUB	f {,WREG}	Destination = f – WREG	1	1					Ι		Ŷ	Ŷ	Û	ţ ţ	Û 5-2	5-230
SUB	#lit10,Wn	Wn = Wn – lit10	٦	1					I		Ŷ	Ŷ	Û	ţ ţ	¢ 2-3	5-231
SUB	Wb,#lit5,Wd	Wd = Wb – lit5	٦	1			Ι	I	I		Ŷ	€	¢	ţ \$	Û 5-2	5-232
SUB	Wb,Ws,Wd	$s_{M} - q_{M} = p_{M}$	1	1				I	I	I	¢	¢	Ŷ	ţ \$	€-5 ŷ	5-233
SUB	Acc	Subtract Accumulators	1	1	Ŷ	Ŷ	Û	Û	€	Û				-	- 2-5	5-235
SUBB	f {,WREG}	destination = $f - WREG - (\overline{C})$	1	1							Ŷ	€	Û	۲ ۲	¢ 2-3	5-236
SUBB	#lit10,Wn	Wn = Wn – lit10 – (\overline{C})	٦	1							Ŷ	€	¢	۲	¢ 2-3	5-237
SUBB	Wb,#lit5,Wd	Wd = Wb – lit5 – (\overline{C})	٦	1					I		Ŷ	Ŷ	Û	ا (Û 5-2	5-238
SUBB	Wb,Ws,Wd	$Wd = Wb - Ws - (\overline{C})$	٦	1			Ι	I	I		Ŷ	€	Ŷ	۲ ۲	Û 5-2	5-239
SUBBR	f {,WREG}	Destination = WREG $-f - (\overline{C})$	1	1			Ι	Ι	Ι		Û	Û	¢	ر (¢ 2-3	5-241
SUBBR	Wb,#lit5,Wd	Wd = lit5 – Wb – (\overline{C})	٦	1					I		Ŷ	Ŷ	Û	ا (¢ 2-3	5-242
SUBBR	Wb,Ws,Wd	$Wd = Ws - Wb - (\overline{C})$	٦	1			Ι	I	I		Ŷ	€	Ŷ	۲ ۲	¢ 2-3	5-243
SUBR	f {,WREG}	Destination = WREG – f	1	1			Ι	Ι	I		Û	Û	¢	ţ (¢ 2-3	5-245
SUBR	Wb,#lit5,Wd	Wd = lit5 – Wb	1	1	I			I	I		Û	Û	Û	ţ ((5-2	5-246
SUBR	Wb,Ws,Wd	Md = Ws - Wb	1	1			I				Û	Û	¢	ţ (¢ 2-3	5-247
SWAP	Wn	Wn = byte or nibble swap Wn	1	1	I				Ι		I			' 	- 5-2	5-249
TBLRDH	Ws,Wd	Read high program word to Wd	1	2		Ι			I						- 5-2	5-250
TBLRDL	Ws,Wd	Read low program word to Wd	-	2		Ι	Ι	Ι	Ι	Ι					- 2-5	5-252
TBLWTH	Ws,Wd	Write Ws to high program word	-	2		I			I						- 2-5	5-254
TBLWTL	Ws,Wd	Write Ws to low program word	-	2	I	I	Ι	I	Ι	I	I	I			- 2-5	5-256
ULNK		Unlink frame pointer	٢	-	I	Ι	I	Ι	I	I					- 5-2	5-258
Legend: Note:	${\mathfrak S}$ set or cleared; ${\mathfrak Q}$ may be cleared, but never set; ${\mathfrak Q}$ may be SA, SB and SAB are only modified if the corresponding sate	e set, but never cleared; uration is enabled, otherw	'1' always set; ise unchanged.	set; '0' always cleared; ged.	ays cle		— unchanged	anged								

Table 6-3: dsPIC30F Instruction Set Summary Table (Continued)

Table 6-3:		dsPIC30F Instruction Set Summary Table (Continued)														
	Assembly Syntax Mnemonic,Operands	Description	Words	Words Cycles OA OB	OA	ОВ	SA	SB	OAB	SB OAB SAB DC	DC	z	ov	z	ပ	C Page #
XOR	f {,WREG}	Destination = f .XOR. WREG	-	-	Ι	I	I	I	I	I	Ι	⇔	I	⇔		5-259
XOR	#lit10,Wn	Wn = lit10 .XOR. Wn	٢	1	I	Ι	I			I		⇔	I	€		5-260
XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1		-						¢		Û	-	5-261
XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1			I			I		¢	Ι	Û	-	5-262
ZE	Ws,Wd	Wd = zero-extended Ws	1	1	I		I	l		I		0	I	€	1	5-264
Legend: Note:	(a) set or cleared; \oplus may be cleared, bucket SA. SB and SAB are only modified if	Legend: ③ set or cleared; ④ may be cleared, but never set; ① may be set, but never cleared; '1' always set; '0' always cleared; — unchanged Note: SA SB and SAB are only modified if the corresponding saturation is enabled, otherwise unchanged.	1' always : e unchane	set; 'o' alw aed.	/ays cle	eared;	— uncł	nanged								

INDEX

Α

Accumulator A, Accumulator B	2-5
Accumulator Access	
Accumulator Usage	
Addressing Modes for Wd Destination Register	5-3
Addressing Modes for Ws Source Register	5-3
Assigned Working Register Usage	

в

te Operations 4-13

С

Code Examples	
'Z' Status bit Operation for 32-bit Addition	. 4-26
Base MAC Syntax	
File Register Addressing	
File Register Addressing and WREG	4-3
Frame Pointer Usage	
Illegal Word Move Operations	. 4-18
Immediate Addressing	. 4-10
Indirect Addressing with Effective Address Update	4-6
Indirect Addressing with Register Offset	4-7
Legal Word Move Operations	. 4-17
MAC Accumulator WB Syntax	. 4-36
MAC Pre-Fetch Syntax	. 4-35
Move with Literal Offset Instructions	4-7
MSC Instruction with Two Pre-Fetches and	
Accumulator Write Back	. 4-36
Normalizing with FBCL	. 4-39
Register Direct Addressing	4-4
Sample Byte Math Operations	. 4-15
Sample Byte Move Operations	. 4-14
Scaling with FBCL	. 4-38
Stack Pointer Usage	
Unsigned f and WREG Multiply	
(Legacy MULWF Instruction)	. 4-29
Using 10-bit Literals for Byte Operands	.4-19
Using the Default Working Register WREG	. 4-28
Conditional Branch Instructions	. 4-25
Core Control Register	2-9
Core Special Function Register Map	6-3

D

Data Addressing Mode Tree	
Data Addressing Modes	
Data Memory Map	
DCOUNT Register	
Default Working Register (WREG)	2-3
Default Working Register WREG	
Development Support	
DOEND Register	2-6
DOSTART Register	2-6
DSP Accumulator Instructions	
DSP Data Formats	
DSP MAC Indirect Addressing Modes	
DSP MAC Instructions	
dsPIC30F Overview	
F	

File I	Register	Addressing	4-	2	
--------	----------	------------	----	---	--

L

Immediate Addressing
Operands in the Instruction Set
Implied DSP Operands
Implied Frame and Stack Pointer 4-27
Instruction Bit Map
Instruction Description Example 5-6
Instruction Descriptions5-7
ADD (16-bit Signed Add to Accumulator) 5-12
ADD (Add Accumulators) 5-11
ADD (Add f to WREG)5-7
ADD (Add Literal to Wn)5-8
ADD (Add Wb to Short Literal)5-9
ADD (Add Wb to Ws)5-10
ADDC (Add f to WREG with Carry) 5-14
ADDC (Add Literal to Wn with Carry)
ADDC (Add Wb to Short Literal with Carry) 5-16
ADDC (Add Wb to Ws with Carry)
AND (AND f and WREG)
AND (AND Literal and Wd)
AND (AND Wb and Short Literal)
AND (AND Wb and Ws)
ASR (Arithmetic Shift Right by Short Literal)
ASR (Arithmetic Shift Right by Wns)
ASR (Arithmetic Shift Right f)5-24 ASR (Arithmetic Shift Right Ws)5-25
BCLR (Bit Clear in Ws)
BCLR.B (Bit Clear f)
BRA (Branch Unconditionally)
BRA (Computed Branch)
BRA C (Branch if Carry)
BRA GE (Branch if Signed Greater Than
or Equal)
BRA GEU (Branch if Unsigned Greater Than
or Equal)
BRA GT (Branch if Signed Greater Than)
BRA GTU (Branch if Unsigned Greater Than) 5-38
BRA LE (Branch if Signed Less Than or Equal) 5-39
BRA LEU (Branch if Unsigned Less Than
or Equal)5-40
BRA LT (Branch if Signed Less Than) 5-41
BRA LTU (Branch if Not Carry) 5-44
BRA LTU (Branch if Unsigned Less Than) 5-42
BRA N (Branch if Negative) 5-43
BRA NN (Branch if Not Negative) 5-45
BRA NOV (Branch if Not Overflow)5-46
BRA NZ (Branch if Not Zero)5-47
BRA OA (Branch if Overflow Accumulator A)
BRA OB (Branch if Overflow Accumulator B)
BRA OV (Branch if Overflow) 5-50
BRA SA (Branch if Saturation Accumulator A) 5-51
BRA SA (Branch if Saturation Accumulator A) 5-51 BRA SB (Branch if Saturation Accumulator B) 5-52
BRA SA (Branch if Saturation Accumulator A) 5-51 BRA SB (Branch if Saturation Accumulator B) 5-52 BRA Z (Branch if Zero)
BRA SA (Branch if Saturation Accumulator A) 5-51 BRA SB (Branch if Saturation Accumulator B) 5-52 BRA Z (Branch if Zero)
BRA SA (Branch if Saturation Accumulator A) 5-51 BRA SB (Branch if Saturation Accumulator B) 5-52 BRA Z (Branch if Zero)
BRA SA (Branch if Saturation Accumulator A)5-51BRA SB (Branch if Saturation Accumulator B)5-52BRA Z (Branch if Zero)5-53BSET (Bit Set f)5-54BSET (Bit Set in Ws)5-55BSW (Bit Write in Ws)5-56
BRA SA (Branch if Saturation Accumulator A)5-51BRA SB (Branch if Saturation Accumulator B)5-52BRA Z (Branch if Zero)5-53BSET (Bit Set f)5-54BSET (Bit Set in Ws)5-55BSW (Bit Write in Ws)5-56BTG (Bit Toggle f)5-58
BRA SA (Branch if Saturation Accumulator A)5-51BRA SB (Branch if Saturation Accumulator B)5-52BRA Z (Branch if Zero)5-53BSET (Bit Set f)5-54BSET (Bit Set in Ws)5-55BSW (Bit Write in Ws)5-56BTG (Bit Toggle f)5-58BTG (Bit Toggle in Ws)5-59
BRA SA (Branch if Saturation Accumulator A)5-51BRA SB (Branch if Saturation Accumulator B)5-52BRA Z (Branch if Zero)5-53BSET (Bit Set f)5-54BSET (Bit Set in Ws)5-55BSW (Bit Write in Ws)5-56BTG (Bit Toggle f)5-58BTG (Bit Toggle in Ws)5-59BTSC (Bit Test f, Skip if Clear)5-60
BRA SA (Branch if Saturation Accumulator A)5-51BRA SB (Branch if Saturation Accumulator B)5-52BRA Z (Branch if Zero)5-53BSET (Bit Set f)5-54BSET (Bit Set in Ws)5-55BSW (Bit Write in Ws)5-56BTG (Bit Toggle f)5-58BTG (Bit Toggle in Ws)5-59BTSC (Bit Test f, Skip if Clear)5-60BTSC (Bit Test Ws, Skip if Clear)5-62
BRA SA (Branch if Saturation Accumulator A)5-51BRA SB (Branch if Saturation Accumulator B)5-52BRA Z (Branch if Zero)5-53BSET (Bit Set f)5-54BSET (Bit Set in Ws)5-55BSW (Bit Write in Ws)5-56BTG (Bit Toggle f)5-58BTG (Bit Toggle in Ws)5-59BTSC (Bit Test f, Skip if Clear)5-60BTSC (Bit Test Ws, Skip if Clear)5-62BTSS (Bit Test f, Skip if Set)5-64
BRA SA (Branch if Saturation Accumulator A)5-51BRA SB (Branch if Saturation Accumulator B)5-52BRA Z (Branch if Zero)5-53BSET (Bit Set f)5-54BSET (Bit Set in Ws)5-55BSW (Bit Write in Ws)5-56BTG (Bit Toggle f)5-58BTG (Bit Toggle in Ws)5-59BTSC (Bit Test f, Skip if Clear)5-60BTSC (Bit Test Ws, Skip if Clear)5-62BTSS (Bit Test f, Skip if Set)5-64BTSS (Bit Test Ws, Skip if Set)5-65
BRA SA (Branch if Saturation Accumulator A)5-51BRA SB (Branch if Saturation Accumulator B)5-52BRA Z (Branch if Zero)5-53BSET (Bit Set f)5-54BSET (Bit Set in Ws)5-55BSW (Bit Write in Ws)5-56BTG (Bit Toggle f)5-58BTG (Bit Toggle in Ws)5-59BTSC (Bit Test f, Skip if Clear)5-60BTSC (Bit Test Ws, Skip if Clear)5-62BTSS (Bit Test f, Skip if Set)5-64

BTSTS (Bit Test/Set f)	5-71
BTSTS (Bit Test/Set in Ws)	5-72
CALL (Call Indirect Subroutine)	
CALL (Call Subroutine)	5-73
CLR (Clear Accumulator, Pre-Fetch Operands)	
CLR (Clear f or WREG)	
CLR (Clear Wd)	5-76
CLRWDT (Clear Watchdog Timer)	
COM (Complement f)	
COM (Complement Ws)	5-81
CP (Compare f with WREG, Set Status Flags)	
OP (Compare 1 With WITEC, Oct Otatus 1 lags)	
CP (Compare Wb with lit5, Set Status Flags)	
CP (Compare Wb with Ws, Set Status Flags)	5-84
CP0 (Compare f with 0x0, Set Status Flags)	
CP0 (Compare Ws with 0x0, Set Status Flags)	5-86
CPB (Compare f with WREG using Borrow,	
Set Status Flags)	E 07
CPB (Compare Wb with lit5 using Borrow,	
Set Status Flags)	5-88
CPB (Compare Ws with Wb using Borrow,	
Set Status Flags)	5-89
CPSEQ (Compare Wb with Wn, Skip if Equal)	
CPSGT (Signed Compare Wb with Wn,	
Skip if Greater Than)	5-92
CPSLT (Signed Compare Wb with Wn,	
Skip if Less Than)	5-93
CPSNE (Signed Compare Wb with Wn,	
Skip if Not Equal)	F 04
Skip if Not Equal)	5-94
DAW.B (Decimal Adjust Wn)	5-95
DEC (Decrement f)	
DEC (Decrement Ws)	
DEC2 (Decrement f by 2)	5-98
DEC2 (Decrement Ws by 2)	
DISI (Disable Interrupts Temporarily)	
DIV.S (Signed Integer Divide)	.5-101
DIV.U (Unsigned Integer Divide)	
DIVF (Fractional Divide)	.5-105
DO (Initialize Hardware Loop Literal)	.5-107
DO (Initialize Hardware Loop Wn)	
	.5-109
ED (Euclidean Distance, No Accumulate)	
EDAC (Euclidean Distance)	.5-113
EXCH (Exchange Wns and Wnd)	
FBCL (Find First Bit Change from Left)	.5-116
FF1L (Find First One from Left)	5-118
FF1R (Find First One from Right)	E 100
GOTO (Unconditional Indirect Jump)	.5-123
GOTO (Unconditional Jump)	.5-122
INC (Increment f)	
INC (Increment Ws)	.5-125
INC2 (Increment f by 2)	5-126
INC2 (Increment Ws by 2)	
IOR (Inclusive OR f and WREG)	.5-128
IOR (Inclusive OR Literal and Wn)	
IOR (Inclusive OR Wb and Short Literal)	
IOR (Inclusive OR Wb and Ws)	.5-131
LAC (Load Accumulator)	
LNK (Allocate Stack Frame)	
LSR (Logical Shift Right by Short Literal)	.5-139
LSR (Logical Shift Right by Wns)	
LSR (Logical Shift Right f)	.5-140
1 CD (1 a min all Ch; ft D; mht M/a)	.5-140 .5-136
LSR (Logical Shift Right Ws)	.5-140 .5-136
	.5-140 .5-136 .5-137
MAC (Multiply and Accumulate)	.5-140 .5-136 .5-137 .5-141
MAC (Multiply and Accumulate) MAC (Square and Accumulate)	.5-140 .5-136 .5-137 .5-141 .5-143
MAC (Multiply and Accumulate) MAC (Square and Accumulate)	.5-140 .5-136 .5-137 .5-141 .5-143
MAC (Multiply and Accumulate) MAC (Square and Accumulate) MOV (Move 16-bit Literal to Wn)	.5-140 .5-136 .5-137 .5-141 .5-143 .5-150
MAC (Multiply and Accumulate) MAC (Square and Accumulate) MOV (Move 16-bit Literal to Wn) MOV (Move f to Destination)	.5-140 .5-136 .5-137 .5-141 .5-143 .5-150 .5-145
MAC (Multiply and Accumulate) MAC (Square and Accumulate) MOV (Move 16-bit Literal to Wn)	.5-140 .5-136 .5-137 .5-141 .5-143 .5-150 .5-145 .5-147

MOV (Move Wns to f)	5-148
MOV (Move WREG to f)	5-146
MOV (Move Ws to Wd)	
MOV (Move Ws with offset to Wnd)	
MOV.B (Move 8-bit Literal to Wnd)	
MOV.D (Double-Word Move from Source	
to Wnd)	5-155
MOV D (Dauble Word Mayo from Wra	5-155
MOV.D (Double-Word Move from Wns	- 4-7
to Destination)	5-157
MOVSAC (Pre-Fetch Operands and	
Store Accumulator)	
MPY (Multiply Wm by Wn to Accumulator)	5-161
MPY (Square to Accumulator)	
MPY.N (Multiply -Wm by Wn to Accumulator)	
MSC (Multiply and Subtract from Accumulator)	
MUL (Integer Unsigned Multiply f and WREG)	
MUL.SS (Integer 16x16-bit Signed Multiply)	5-170
MUL.SU (Integer 16x16-bit	
Signed-Unsigned Multiply)	5-174
MUL.SU (Integer 16x16-bit	
Signed-Unsigned Short Literal Multiply)	5-172
MUL.US (Integer 16x16-bit	
Unsigned-Signed Multiply)	E 176
MUL.UU (Integer 16x16-bit Unsigned Multiply)	5-179
MUL.UU (Integer 16x16-bit Unsigned	
Short Literal Multiply)	5-178
NEG (Negate Accumulator)	5-183
NEG (Negate f)	
NEG (Negate Ws)	5-182
NOP (No Operation)	5 1 9 /
NOPR (No Operation)	5-185
POP (Pop TOS to f)	5-186
POP (Pop TOS to Wd)	5-187
POP.D (Double Pop TOS to Wnd/	
Wnd+1)	5-188
Wnd+1) POP S (Pop Shadow Registers)	
POP.S (Pop Shadow Registers)	5-189
POP.S (Pop Shadow Registers) PUSH (Push f to TOS)	5-189 5-190
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS)	5-189 5-190
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/	5-189 5-190 5-191
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS)	5-189 5-190 5-191 5-192
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/	5-189 5-190 5-191 5-192
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers)	5-189 5-190 5-191 5-192 5-193
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode)	5-189 5-190 5-191 5-192 5-193 5-194
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call)	5-189 5-190 5-191 5-192 5-193 5-194 5-196
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call)	5-189 5-190 5-191 5-192 5-193 5-194 5-196 5-195
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times).	5-189 5-190 5-191 5-192 5-193 5-193 5-194 5-196 5-195 5-197
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times)	5-189 5-190 5-191 5-192 5-193 5-193 5-194 5-196 5-195 5-197 5-198
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RESET (Reset)	5-189 5-190 5-191 5-192 5-193 5-194 5-196 5-195 5-197 5-198 5-200
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RESET (Reset) RETFIE (Return from Interrupt)	5-189 5-190 5-191 5-192 5-193 5-193 5-194 5-195 5-195 5-197 5-198 5-200 5-201
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RESET (Reset) RETFIE (Return from Interrupt)	5-189 5-190 5-191 5-192 5-193 5-193 5-194 5-195 5-195 5-197 5-198 5-200 5-201
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RESET (Reset) RETFIE (Return from Interrupt) RETLW (Return with Literal in Wn)	5-189 5-190 5-191 5-192 5-193 5-193 5-194 5-195 5-195 5-197 5-198 5-200 5-201 5-202
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RESET (Reset) RESET (Reset) RETFIE (Return from Interrupt) RETLW (Return with Literal in Wn) RETURN (Return)	5-189 5-190 5-191 5-191 5-193 5-193 5-194 5-195 5-195 5-197 5-198 5-200 5-201 5-203
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RESET (Reset) RESET (Reset) RETFIE (Return from Interrupt) RETURN (Return) RLC (Rotate Left f through Carry)	5-189 5-190 5-191 5-191 5-193 5-193 5-194 5-195 5-195 5-197 5-198 5-200 5-201 5-202 5-203 5-204
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RESET (Reset) RESET (Reset) RETFIE (Return from Interrupt) RETLW (Return with Literal in Wn) RETURN (Return) RLC (Rotate Left f through Carry) RLC (Rotate Left Ws through Carry)	5-189 5-190 5-191 5-191 5-193 5-193 5-194 5-195 5-195 5-197 5-197 5-198 5-201 5-202 5-203 5-204 5-205
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RESET (Reset) RETFIE (Return from Interrupt) RETLW (Return with Literal in Wn) RETURN (Return) RLC (Rotate Left f through Carry) RLNC (Rotate Left f without Carry)	5-189 5-190 5-191 5-191 5-193 5-193 5-194 5-195 5-195 5-197 5-197 5-198 5-201 5-201 5-202 5-203 5-204 5-205 5-207
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RESET (Reset) RETFIE (Return from Interrupt) RETIRE (Return with Literal in Wn) RETURN (Return) RETURN (Return) RLC (Rotate Left f through Carry) RLNC (Rotate Left f without Carry) RLNC (Rotate Left Ws without Carry)	5-189 5-190 5-191 5-191 5-193 5-193 5-194 5-195 5-195 5-197 5-198 5-201 5-201 5-203 5-204 5-205 5-207 5-208
 POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) RESET (Reset) RETFIE (Return from Interrupt) RETUW (Return with Literal in Wn) RETURN (Return) RLC (Rotate Left f through Carry) RLNC (Rotate Left f ws without Carry) RRC (Rotate Left Ws without Carry) RRC (Rotate Left f through Carry) 	5-189 5-190 5-191 5-193 5-193 5-194 5-195 5-195 5-197 5-201 5-201 5-203 5-203 5-204 5-205 5-207 5-208 5-208 5-210
 POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call)	5-189 5-190 5-191 5-193 5-194 5-194 5-196 5-195 5-197 5-197 5-201 5-201 5-203 5-203 5-204 5-205 5-207 5-208 5-210 5-211
 POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) RESET (Reset) RETFIE (Return from Interrupt) RETUW (Return with Literal in Wn) RETURN (Return) RLC (Rotate Left f through Carry) RLNC (Rotate Left f ws without Carry) RRC (Rotate Left Ws without Carry) RRC (Rotate Left f through Carry) 	5-189 5-190 5-191 5-193 5-194 5-194 5-196 5-195 5-197 5-197 5-201 5-201 5-203 5-203 5-204 5-205 5-207 5-208 5-210 5-211
 POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) RETFIE (Return from Interrupt) RETFIE (Return with Literal in Wn) RETURN (Return) RLC (Rotate Left f through Carry) RLNC (Rotate Left f without Carry) RRNC (Rotate Right f without Carry) RRNC (Rotate Right f without Carry) 	5-189 5-190 5-191 5-193 5-193 5-194 5-195 5-195 5-197 5-197 5-201 5-202 5-202 5-203 5-204 5-205 5-207 5-208 5-211 5-213
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) REFEAT (Repeat Next Instruction Wn Times) REFEAT (Repeat Next Instruction Wn Times) REFIE (Return from Interrupt) RETIR (Return with Literal in Wn) RETURN (Return) RLC (Rotate Left f through Carry) RLNC (Rotate Left Ws without Carry) RRC (Rotate Right f through Carry) RRC (Rotate Right f without Carry) RRNC (Rotate Right f without Carry) RNNC (Rotate Right f without Carry) RNNNC (Rotate Right f without Carry)	5-189 5-192 5-193 5-194 5-194 5-196 5-195 5-195 5-197 5-198 5-200 5-201 5-203 5-204 5-205 5-203 5-208 5-211 5-213 5-214
POP.S (Pop Shadow Registers) PUSH (Push f to TOS) PUSH (Push Ws to TOS) PUSH.D (Double Push Wns/ Wns+1 to TOS) PUSH.S (Push Shadow Registers) PWRSAV (Enter Power Saving Mode) RCALL (Computed Relative Call) RCALL (Relative Call) RCALL (Relative Call) REPEAT (Repeat Next Instruction 'lit14' Times) REPEAT (Repeat Next Instruction Wn Times) REFEAT (Repeat Next Instruction Wn Times) REFEAT (Repeat Next Instruction Wn Times) RETFIE (Return from Interrupt) RETTIR (Return with Literal in Wn) RETURN (Return) RLC (Rotate Left f through Carry) RLC (Rotate Left Ws through Carry) RLNC (Rotate Left Ws without Carry) RRC (Rotate Right f through Carry) RRNC (Rotate Right f without Carry) RRNC (Rotate Right f without Carry) RRNC (Rotate Right Ws without Carry) RRNC (Rotate Right f without Carry) RRNC (Rotate Right Ws without Carry) RRNC (Rotate Right Ws without Carry) RRNC (Rotate Right f without Carry) RRNC (Rotate Right Ws without Carry) RRNC (Rotate Right Ws without Carry) RRNC (Rotate Right Ws without Carry) RRNC (Rotate Right f without C	5-189 5-192 5-193 5-194 5-194 5-194 5-195 5-195 5-195 5-197 5-198 5-200 5-203 5-203 5-203 5-205 5-208 5-211 5-213 5-214 5-216
 POP.S (Pop Shadow Registers)	5-189 5-192 5-193 5-194 5-194 5-194 5-196 5-195 5-195 5-197 5-198 5-200 5-203 5-203 5-203 5-203 5-208 5-208 5-211 5-213 5-214 5-218
 POP.S (Pop Shadow Registers)	5-189 5-192 5-193 5-194 5-194 5-196 5-195 5-195 5-197 5-198 5-200 5-201 5-203 5-202 5-203 5-203 5-200 5-210 5-210 5-213 5-214 5-218 5-218 5-220
 POP.S (Pop Shadow Registers)	5-189 5-192 5-193 5-194 5-194 5-195 5-195 5-195 5-195 5-197 5-198 5-200 5-201 5-203 5-203 5-203 5-204 5-210 5-210 5-213 5-214 5-218 5-221
 POP.S (Pop Shadow Registers)	5-189 5-192 5-191 5-193 5-194 5-195 5-195 5-195 5-195 5-197 5-198 5-200 5-201 5-203 5-203 5-203 5-203 5-203 5-203 5-203 5-203 5-203 5-210 5-213 5-214 5-218 5-221 5-222
 POP.S (Pop Shadow Registers)	5-189 5-192 5-193 5-194 5-194 5-195 5-195 5-195 5-195 5-195 5-197 5-198 5-200 5-201 5-203 5-203 5-203 5-207 5-203 5-203 5-203 5-204 5-210 5-213 5-214 5-218 5-222 5-223
 POP.S (Pop Shadow Registers)	5-189 5-192 5-193 5-194 5-194 5-195 5-195 5-195 5-195 5-195 5-197 5-198 5-200 5-201 5-203 5-203 5-203 5-207 5-203 5-203 5-203 5-204 5-210 5-213 5-214 5-218 5-222 5-223
 POP.S (Pop Shadow Registers)	5-189 5-190 5-191 5-191 5-193 5-194 5-195 5-195 5-195 5-195 5-197 5-198 5-200 5-201 5-203 5-203 5-203 5-204 5-203 5-203 5-204 5-203 5-204 5-213 5-214 5-214 5-221 5-223 5-223 5-224
 POP.S (Pop Shadow Registers)	5-189 5-190 5-191 5-191 5-193 5-194 5-195 5-195 5-195 5-195 5-197 5-198 5-200 5-201 5-203 5-204 5-203 5-203 5-204 5-203 5-204 5-205 5-201 5-213 5-214 5-218 5-228 5-223 5-224 5-228

SL (Shift Left f)	5-225
SL (Shift Left Ws)	
SUB (Subtract Accumulators)	5-235
SUB (Subtract Literal from Wn)	
SUB (Subtract Short Literal from Wb)	
SUB (Subtract WREG from f)	
SUB (Subtract Ws from Wb)	5-233
SUBB (Subtract Short Literal from	
Wb with Borrow)	5-238
SUBB (Subtract Wn from Literal with Borrow)	
SUBB (Subtract WREG and Carry bit from f)	
SUBB (Subtract WKEC and Carly bit norm)	
SUBBR (Subtract f from WREG with Borrow)	5-241
SUBBR (Subtract Wb from	
Short Literal with Borrow)	
SUBBR (Subtract Wb from Ws with Borrow)	5-243
SUBR (Subtract f from WREG)	
SUBR (Subtract Wb from Short Literal)	
SUBR (Subtract Wb from Ws)	
SWAP (Byte or Nibble Swap Wn)	
TBLRDH (Table Read High)	
TBLRDL (Table Read Low)	
TBLWTH (Table Write High)	5-254
TBLWTL (Table Write Low)	5-256
ULNK (De-allocate Stack Frame)	
XOR (Exclusive OR f and WREG)	
XOR (Exclusive OR Literal and Wn)	
XOR (Exclusive OR Wb and Short Literal)	
XOR (Exclusive OR Wb and Ws)	5-262
ZE (Zero-Extend Wn)	5-264
Instruction Encoding Field Descriptors Introduction	
Instruction Set Overview	
Bit Instructions Compare/Skip Instructions	
Compare/Skip Instructions	
Control Instructions	3-10
	3-10
Control Instructions DSP Instructions	3-10 3-10
Control Instructions DSP Instructions dsPIC30F Instruction Groups	3-10 3-10 3-2
Control Instructions DSP Instructions dsPIC30F Instruction Groups Logic Instructions	3-10 3-10 3-2 3-5
Control Instructions DSP Instructions dsPIC30F Instruction Groups Logic Instructions Math Instructions	3-10 3-10 3-2 3-5 3-4
Control Instructions DSP Instructions dsPIC30F Instruction Groups Logic Instructions Math Instructions Move Instructions	3-10 3-10 3-2 3-5 3-4 3-3
Control Instructions DSP Instructions dsPIC30F Instruction Groups Logic Instructions Math Instructions Move Instructions Program Flow Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9
Control Instructions DSP Instructions dsPIC30F Instruction Groups Logic Instructions Math Instructions Move Instructions Program Flow Instructions Rotate/Shift Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6
Control Instructions DSP Instructions dsPIC30F Instruction Groups Logic Instructions Math Instructions Move Instructions Program Flow Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6
Control Instructions DSP Instructions dsPIC30F Instruction Groups Logic Instructions Math Instructions Move Instructions Program Flow Instructions Rotate/Shift Instructions Shadow/Stack Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4 1-4 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4 1-4 1-4 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4 1-4 1-4 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4 1-4 1-4 1-4 1-4 1-4 1-4
Control Instructions	$\begin{array}{c} & & & & & & & & & & & \\ & & & & & & & $
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4 1-4 1-4 1-4 1-4 1-4 1-4 1-4
Control Instructions	3-10 3-10 3-2 3-5 3-4 3-3 3-3 3-9 3-6 3-10 6-9 1-4 1-4 1-4 1-4 1-4 1-4 1-4 1-4 1-4 1-4
Control Instructions	$\begin{array}{c} \begin{tabular}{c} & & \end{tabular} & & $
Control Instructions	$\begin{array}{c} \begin{tabular}{lllllllllllllllllllllllllllllllllll$

Slit51-4
TOS 1-4
Wb 1-4
Wd 1-4
Wm*Wm 1-4
Wm*Wn 1-4
Wm, Wn1-4
Wn 1-4
Wnd 1-4
Wns 1-4
WREG 1-4
Ws 1-4
Wx 1-4
Wxd 1-4
Wy 1-4
Wyd 1-4
Instruction Stalls 4-12
DO/REPEAT Loops 4-13
Exceptions 4-13
Instructions that Change Program Flow 4-13
PSV 4-13
RAW Dependency Detection 4-12
Instruction Symbols
Integer and Fractional Data 4-30
Representation 4-31
Interrupt Priority Level
Introduction 1-2

М

MAC	
Operations	4-34
Pre-Fetch Register Updates	4-34
Pre-Fetches	4-33
Syntax	4-34
Write Back	4-34
MAC Accumulator Write Back Selection	5-5
MAC or MPY Source Operands	
(Different Working Register)	5-5
MAC or MPY Source Operands	
(Same Working Register)	5-5
Manual Objective	1-2
Microchip Documentation	
Modulo and Bit-Reversed Addressing Modes	4-8
Multi-Cycle Instructions	3-2
Multi-Word Instructions	3-3

Ν

Normalizing the Accumulator with the	
FBCL Instruction	4-39

Offset Addressing Modes for Wd Destination Register (with Register Offset)	
Offset Addressing Modes for Ws Source Register (with Register Offset)	
Ρ	
PICmicro® Microcontroller Compatibility PRODH	. 4-28
PRODL Register Pair	. 4-28
Program Addressing Modes	. 4-11
Mathada of Madifuing Elow	4 4 4

т

Programmer's Model	2-3
Diagram	
Register Descriptions	2-3
PSVPAG Register	2-5

R

RCOUNT Register	2-6
Register Direct Addressing	4-4
Register Indirect Addressing	4-5
Modes	4-5
Register Indirect Addressing and the Instruction Set.	4-8
Registers	
CORCON (Core Control) Register	2-12
SR (Status) Register	2-10
Related Documents	1-5

S

Scaling Data with the FBCL Instruction	4-37
Scaling Examples	
Shadow Registers	
Automatic Usage	2-9
Software Stack Frame Pointer	2-3, 4-22
Example	4-23
Overflow	4-24
Underflow	4-24
Software Stack Pointer	
Example	4-21
Stack Pointer Limit Register (SPLIM)	2-5
Status Register	2-7
DSP ALU Status Bits	2-8
Loop Status Bits	2-7
MCU ALU Status Bits	2-7
Style and Symbol Conventions	1-3
Document Conventions	1-3

TBLPAG Register
U
Using 10-bit Literal Operands
w
Word Move Operations 4-16 Data Alignment in Memory 4-16 Working Register Array 2-3
x
X Data Space Pre-Fetch Destination5-4 X Data Space Pre-Fetch Operation5-4
Y
Y Data Space Pre-Fetch Destination5-5 Y Data Space Pre-Fetch Operation5-4
Z
Z Status Bit 4-26

NOTES:



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

Atlanta

3780 Mansell Road, Suite 130 Alpharetta, GA 30022 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, IN 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

Phoenix

2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7966 Fax: 480-792-4338

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 Marketing Support Division 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. 2401-2402, 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 - 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

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. 1812, 18/F, Building A, United Plaza No. 5022 Binhe Road, Futian District Shenzhen 518033, China Tel: 86-755-82901380 Fax: 86-755-82966626

China - Qingdao

Rm. B505A, Fullhope Plaza, No. 12 Hong Kong Central Rd. Qingdao 266071, China Tel: 86-532-5027355 Fax: 86-532-5027205 India Microchip Technology Inc. India Liaison Office Marketing Support Division **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 Steinheilstrasse 10

D-85737 Ismaning, Germany Tel: 49-89-627-144-0 Fax: 49-89-627-144-44

Italy

Microchip Technology SRL Via Quasimodo, 12 20025 Legnano (MI) Milan, Italy Tel: 39-0331-742611 Fax: 39-0331-466781 United Kingdom Microchip Ltd. 505 Eskdale Road Winnersh Triangle Wokingham Berkshire, England RG41 5TU Tel: 44 118 921 5869 Fax: 44-118 921-5820

03/25/03