

# dsPIC30F Language Tools Quick Reference Card



***dsPIC***®

# MPLAB® ASM30 Assembler

## Assembler Command Line Options

pic30-as [options|sourcefiles]...

### Listing Output

Option	Description
-a[ <i>sub-option</i> ] [=file]	The -a option supports the following suboptions:
c	Omit false conditionals
d	Omit debugging directives
h	Include high-level source
l	Include assembly
m	Include macro expansions
n	Omit forms processing
s	Include symbols
i	Include section information

### Informational Output

Option	Description
--fatal-warnings	Warnings treated as errors
--no-warn	Suppress warnings
--warn	Warnings issued (default)
-J	No signed overflow warnings
--help	Show help
--target-help	Show target-specific help
--statistics	Show statistics after assembly
--version	Show version number
--verbose	Maximum message output

### Output File Creation

Option	Description
-o <i>objfile</i>	Set name of output to <i>objfile</i>
--relax	Substitute short branches where possible
--no-relax	Don't substitute short branches (default)
-Z	Generate object file even after errors
-MD <i>file</i>	Write dependency information to <i>file</i>

### Other Options

Option	Description
--defsym <i>sym</i> = <i>value</i>	Define <i>sym</i> with <i>value</i>
-I <i>dir</i>	Add <i>dir</i> to .include directories
-p= <i>proc</i>	Set target processor to <i>proc</i>

## Special Operators

Operator	Description
tblpage( <i>name</i> )	Get page for table read/write operations
tbloffset( <i>name</i> )	Get pointer for table read/write operations
psvpage( <i>name</i> )	Get page for PSV data window operations
psvoffset( <i>name</i> )	Get pointer for PSV data window operations
paddr( <i>label</i> )	Get 24-bit address of <i>label</i> in program memory
handle( <i>label</i> )	Get 16-bit reference to <i>label</i> in program memory
.sizeof.( <i>name</i> )	Get size of section <i>name</i> in address units
.startof.( <i>name</i> )	Get starting address of section <i>name</i>

## Reserved Symbol Names

Note: Reserved symbol names can be upper or lower case

W0	W1	W2	W3	W4	W5	W6	W7
W8	W9	W10	W11	W12	W13	W14	W15
WREG	A	B	OV	C	Z	N	GE
LT	GT	LE	NOV	NC	NZ	NN	GEU
LTU	GTU	LEU	OA	OB	SA	SB	

## Assembler Directives

### Sections

Directive	Description
<code>.bss</code>	Uninitialized data
<code>.data</code>	Initialized data
<code>.section name , "flags"</code>	Set section <i>name</i> with <i>flags</i> : <b>b</b> bss section <b>n</b> Section is not loaded <b>d</b> Data section <b>r</b> Read only section (PSV) <b>x</b> Executable section
<code>.text</code>	Executable code

### Constant Initialization

Directive	Description
<code>.ascii</code>	String with no trailing zero byte
<code>.asciz</code>	String with trailing zero byte
<code>.byte</code>	8-bit value
<code>.pbyte</code>	8-bit value including upper byte of program memory
<code>.word</code>	16-bit value
<code>.pword</code>	24-bit value including upper byte of program memory
<code>.long</code>	32-bit value
<code>.float</code>	32-bit single precision floating point
<code>.double</code>	64-bit double precision floating point

### Symbols

Directive	Description
<code>.bss sym, len</code>	Reserve <i>len</i> bytes for uninitialized symbols
<code>.comm sym, len</code>	Declare common
<code>.extern sym</code>	Defined as global in another module
<code>.global sym</code>	Symbol available in other modules
<code>.weak sym</code>	Mark symbol as weak
<code>.equ sym, expr</code>	Set value of <i>sym</i> to <i>expr</i>
<code>.equiv sym, expr</code>	Like <code>.equ</code> , but issues error if <i>sym</i> already defined

## Section Location

Directive	Description
<code>.align <i>algn</i> [, <i>byte</i> [, <i>max-skip</i>]]</code>	Pad to <i>algn</i> boundary
<code>.palign <i>algn</i> [, <i>byte</i> [, <i>max-skip</i>]]</code>	Pad to <i>algn</i> boundary in program memory specifying upper byte
<code>.fill <i>repeat</i> [, <i>size</i> [, <i>byte</i>]]</code>	Fill <i>repeat</i> bytes
<code>.pfill <i>repeat</i> [, <i>size</i> [, <i>byte</i>]]</code>	Fill <i>repeat</i> bytes in program memory, specifying upper byte
<code>.org <i>new-lc</i> [, <i>byte</i>]</code>	Set location counter
<code>.porg <i>new-lc</i> [, <i>byte</i>]</code>	Set location counter in code section specifying upper byte
<code>.skip <i>size</i> [, <i>byte</i>]</code> <code>.space <i>size</i> [, <i>byte</i>]</code>	Fill bytes
<code>.pskip <i>size</i> [, <i>byte</i>]</code> <code>.pspace <i>size</i> [, <i>byte</i>]</code>	Fill bytes in program memory specifying upper byte

## Conditional

Directive	Description
<code>.if <i>expr</i></code>	Start of conditional
<code>.ifdef <i>sym</i></code>	Conditional if <i>sym</i> defined
<code>.ifndef <i>sym</i></code>	Conditional if <i>sym</i> not defined
<code>.else</code>	Alternative to conditional
<code>.endif</code>	End of conditional
<code>.err <i>msg</i></code>	Print message

# MPLAB LINK30 Object Linker

`pic30-ld [options] file...`

## Command Line Options

Option	Description
<code>-l <i>libname</i></code>	Search for library <i>libname</i>
<code>-M</code>	Print map file on standard output
<code>-r</code>	Generate relocatable output
<code>-S</code>	Strip debug symbols
<code>-T <i>file</i></code>	Read linker script
<code>-u <i>sym</i></code>	Start with undefined reference to <i>sym</i>
<code>-v</code>	Print version information
<code>-( -llib1 ... -llibn -)</code>	Resolve circular references in libraries
<code>--no-check-sections</code>	Do not check section addresses overlap
<code>--cref</code>	Output cross reference table
<code>--defsym <i>sym</i>=<i>expr</i></code>	Define symbol
<code>--heap <i>size</i></code>	Set heap to <i>size</i> bytes
<code>--help</code>	Print list of command line options
<code>-Map <i>file</i></code>	Write a map file
<code>--no-data-init</code>	Don't create initialized data template
<code>--no-handles</code>	Don't create handle jump table
<code>--no-pack-data</code>	Use only lower 16-bits for data storage
<code>--stack <i>size</i></code>	Set minimum stack to <i>size</i> bytes (default=16)
<code>--warn-once</code>	Warn only once per undefined symbol
<code>--warn-section-align</code>	Warn if alignment changes start of section

# Object File Utilities

## Executables

Name	Description
<code>pic30-ar</code>	Manage libraries
<code>pic30-bin2hex</code>	Convert executable to Intel HEX file
<code>pic30-nm</code>	List symbols from an object file
<code>pic30-objdump</code>	Display information about object files
<code>pic30-ranlib</code>	Generate library index
<code>pic30-strings</code>	Print character sequences from file
<code>pic30-strip</code>	Discard all symbols from an object or archive file
<code>pic30-lm</code>	License manager

## pic30-objdump Utility

`pic30-objdump [ -options ] file`

### Options

Option	Description
<code>-a</code>	Display archive header information
<code>-d</code>	Disassemble sections that contain instructions
<code>-D</code>	Disassemble all sections
<code>-z</code>	Disassemble blocks, even if zeroes
<code>-f</code>	Display summary information from headers
<code>-g</code>	Display debugging information
<code>-h</code>	Display summary information from section headers
<code>-j name</code>	Display information only for section <i>name</i>
<code>-l</code>	Label display with file names and line numbers
<code>-S</code>	Display source intermixed with disassembly
<code>-M symbolic</code>	Perform symbolic disassembly
<code>-r</code>	Print relocation entries of file
<code>-s</code>	Display full contents of section(s)
<code>-t</code>	Print symbol table entries of file
<code>-x</code>	Display all available header information same as: <code>-a -f -h -r -t</code>
<code>--start-address=adr</code>	Start displaying data at <i>adr</i> for <code>-d</code> , <code>-r</code> , <code>-s</code>
<code>--stop-address=adr</code>	Stop displaying data at <i>adr</i> for <code>-d</code> , <code>-r</code> , <code>-s</code>
<code>-V</code>	Print the version number
<code>-H</code>	Print summary of options

## MPLAB LIB30 Archiver/Librarian

`pic30-ar option [mods [reldpos] [count]] archive [member...]`

### Options

Option	Description
<code>d</code>	Delete modules from the archive
<code>p</code>	Print specified members of archive
<code>r</code>	Insert the files <i>member...</i> into <i>archive</i> (with replacement)
<code>t</code>	Display a table listing the contents of <i>archive</i> or files in <i>member</i>
<code>x</code>	Extract members from archive

## Modifiers

Modifier	Description
a	Add new files after existing member of archive
b	Add new files before existing member of archive
c	Create the archive
i	Insert new files before existing member of archive
o	Preserve original dates of members when extracting
P	Use full path
S	Do not generate archive symbol table
U	Insert only those newer than existing
v	Verbose version
V	Show version number

## MPLAB SIM30 Simulator

`sim30 [command-file-name]`

## Commands

Command	Description
LP <i>filename</i> [ <i>adr</i> ]	Load Program memory
LF <i>filename</i> [ <i>adr</i> ]	Load File registers
DM [ <i>start</i> ] [ <i>end</i> ]	Display program Memory
DF [ <i>start</i> ] [ <i>end</i> ]	Display File registers
DA	Display Accumulators
DB	Display Breakpoints
DW	Display W registers
DC	Display pC disassembled
DP	Display Profile
DH	Display Help on all
DS	Display Status register fields
AF <i>freq</i>	Alter clock Frequency
MS <i>start end/val</i> [ <i>val</i> ]	Program Memory Set
MC <i>start</i> [ <i>end</i> ]	Program Memory Clear
FS <i>start end/val</i> [ <i>val</i> ]	File register Set
FC <i>start</i> [ <i>end</i> ]	File register Clear
BS <i>adr1</i> ... [ <i>adrN</i> ]	Breakpoint Set
BC <i>adr1</i> ... [ <i>adrN</i> ]	Breakpoint Clear
RP	Reset Processor
RC	Reset Clock
VO	Verbose On
VF	Verbose off
IO	Turn simulated I/o file On
IF	Turn simulated I/o file off
PS <i>value</i>	PC Set
LD <i>devicename</i>	Load Device
LC <i>filename</i>	Load COFF file
LS <i>filename</i>	Load SCL file
Q	Quit
S	Step
E	Execute
H	Halt
HE	Halt on Error
HW	Halt on Warning

# MPLAB C30 Compiler

`pic30-gcc [options] files`

## Integer Data Types

Type	Bits	Min	Max
char, signed char	8	-128	127
unsigned char	8	0	255
short, signed short	16	-32768	32767
unsigned short	16	0	65535
int, signed int	16	-32768	32767
unsigned int	16	0	65535
long, signed long	32	$-2^{31}$	$2^{31} - 1$
unsigned long	32	0	$2^{32} - 1$
long long, signed long long	64	$-2^{63}$	$2^{63} - 1$
unsigned long long	64	0	$2^{64} - 1$

## IEEE Floating Point Data Types

Type	Bits	E min	E max	N min	N max (approx)
float	32	-126	127	$2^{-126}$	$2^{128}$
double*	32	-126	127	$2^{-126}$	$2^{128}$
long double	64	-1022	1023	$2^{-1022}$	$2^{1024}$

\* double is equivalent to long double if -fno-short-double is used.

## Attributes of Variables

Option	Description
section (name)	Specify that variable is located in section <i>name</i> . Example: <code>int x __attribute__((section(".xdata")))=0;</code>
near	Variable is allocated in first 8 KB of data memory
far	Variable can be located anywhere in data memory

## Attributes of Functions

Option	Description
noreturn	Indicate the function cannot return
weak	Declaration emitted as weak symbol rather than global
near	Function within 32K range
far	Function anywhere in memory
shadow	Use shadow registers rather than stack for saving registers W0-W3
interrupt	Specify function is an interrupt handler

## Section Names

Name	Description
<code>.text</code>	Executable code
<code>.xbss</code>	X memory that is not initialized
<code>.xdata</code>	X memory that has initial values
<code>.nbss</code>	Near memory that is not initialized
<code>.ndata</code>	Near memory that has initial values
<code>.ndconst</code>	Constants in near memory
<code>.pbss</code>	Uninitialized persistent data memory
<code>.bss</code>	Non-initialized general memory
<code>.data</code>	Initialized general memory
<code>.dconst</code>	Constants in data memory
<code>.ybss</code>	Non-initialized Y memory
<code>.ydata</code>	Initialized Y memory
<code>.const</code>	Constants in program memory

## General Options

Option	Description
<code>-c</code>	Stop after assembly
<code>-E</code>	Stop after preprocessing stage
<code>-o file</code>	Place output in <i>file</i>
<code>-S</code>	Stop after compilation
<code>-v</code>	Print commands executed during compilation
<code>--help</code>	Description of the command line options
<code>-D name [=value]</code>	Define symbol <i>name</i>
<code>-U name</code>	Undefine symbol <i>name</i>
<code>-Wl,option1,...,optionN</code>	Pass options to linker

## Memory Model Command Line Options

Option	Memory Definition
<code>-msmall-data</code>	Put all data in first 8 KB of data memory (default)
<code>-msmall-scalar</code>	Put all scalars in first 8 KB of data memory (default)
<code>-mlarge-data</code>	Greater than 8 KB of data memory
<code>-msmall-code</code>	Up to 32 Kwords of program memory (default)
<code>-mlarge-code</code>	Greater than 32 Kwords of program memory
<code>-mconst-in-data</code>	Local constants in data memory
<code>-mconst-in-code</code>	Local constants in program memory (PSV) (default)

## Options for Debugging

Option	Description
<code>-g</code>	Produce debugging information
<code>-save-temps</code>	Do not delete intermediate files

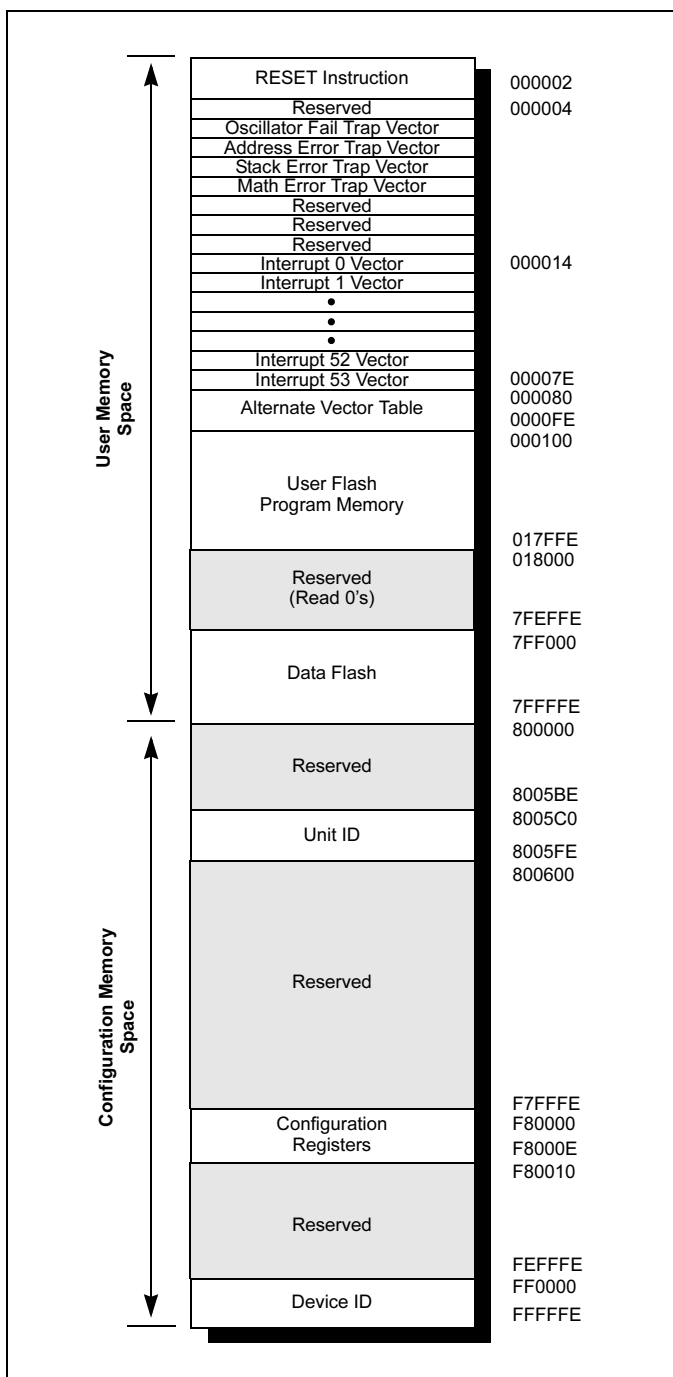
## Options for Controlling Optimization

NOTE: Enabling optimizations may affect debugging

Option	Description
<code>-O0</code>	Do not optimize (default)
<code>-O1</code>	Minimal optimization
<code>-O2</code>	Optimize to balance size and speed
<code>-O3</code>	Optimize for speed
<code>-Os</code>	Optimize for size
<code>-mpa</code>	Enable procedural abstraction optimization



# Program Memory Map



# dsPIC30F Architecture

## Indirect Mode Syntax

Function	Syntax	Byte Inst	Word Inst
No Modification	[Wn]	EA = [Wn]	EA = [Wn]
Pre-Increment	[++Wn]	EA = [Wn += 1]	EA = [Wn += 2]
Pre-Decrement	[--Wn]	EA = [Wn -= 1]	EA = [Wn -= 2]
Post-Increment	[Wn++]	EA = [Wn] += 1	EA = [Wn] += 2
Post-Decrement	[Wn--]	EA = [Wn] -= 1	EA = [Wn] -= 2
Register Offset	[Wn+Wb]	EA = [Wn + Wb]	EA = [Wn + Wb]

## DSP MAC Indirect Addressing Modes

Addressing Mode	X Memory	Y Memory
No Modification	EA = [Wx]	EA = [Wy]
Post-Inc by 2	EA = [Wx] += 2	EA = [Wy] += 2
Post-Inc by 4	EA = [Wx] += 4	EA = [Wy] += 4
Post-Inc by 6	EA = [Wx] += 6	EA = [Wy] += 6
Post-Dec by 2	EA = [Wx] -= 2	EA = [Wy] -= 2
Post-Dec by 4	EA = [Wx] -= 4	EA = [Wy] -= 4
Post-Dec by 6	EA = [Wx] -= 6	EA = [Wy] -= 6
Register Offset	EA = [W9 + W12]	EA = [W11 + W12]

## Registers

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	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 through W15	Working register array
WREG	Synonym for W0

## Symbols Used in Instruction Set Tables

Symbol	Description
f	File Register Address
Wb	Base Working Register
Wd	Destination Working Register (direct and indirect)
Wm,Wn	Working Register Pair (dividend, divisor)
Wn	Both Source and Destination Working Register (direct)
Wnd	Destination Working Register (direct)
Wns	Source Working Register (direct)
Ws	Source Working Register (direct and indirect)
Wx	Source Addressing Mode and Working Register for X Data
Wxd	Destination Working Register for X Data
Wy	Source Addressing Mode and Working Register for Y Data
Wyd	Destination Working Register for Y Data

# Instruction Set

Asm	Operand	Description	W	C
ADD	f{ , WREG}	Destination = f + WREG	1	1
	#lit10, Wn	Wn = lit10 + Wn	1	1
	Wb, #lit5, Wd	Wd = Wb + lit5	1	1
	Wb, Ws, Wd	Wd = Wb + Ws	1	1
	Acc	Add Accumulators	1	1
	Ws, #Slit4, Acc	Add signed lit4 to Accumulator	1	1
ADDC	f{ , WREG}	Dest = f + WREG + (C)	1	1
	#lit10, Wn	Wn = lit10 + Wn + (C)	1	1
	Wb, #lit5, Wd	Wd = Wb + lit5 + (C)	1	1
	Wb, Ws, Wd	Wd = Wb + Ws +(C)	1	1
AND	f{ , WREG}	Destination = f .AND. WREG	1	1
	#lit10, Wn	Wn = lit10 .AND. Wn	1	1
	Wb, #lit5, Wd	Wd = Wb .AND. lit5	1	1
	Wb, Ws, Wd	Wd = Wb .AND. Ws	1	1
ASR	f{ , WREG}	Dest = arith right shift f	1	1
	Ws, Wd	Wd = arith right shift Ws	1	1
	Wb, #lit4, Wnd	Wnd = arith right shift Wb by lit4	1	1
	Wb, Wns, Wnd	Wnd = arith right shift Wb by Wns	1	1
BCLR	f, #bit4	Bit clear f	1	1
	Ws, #bit4	Bit clear Ws	1	1
BRA	Expr	Branch always	1	2
	Wn	Computed branch	1	2
	C, Expr	Branch if carry	1	1-2
	GE, Expr	Branch if >= 1	1	1-2
	GEU, Expr	Branch if unsigned >=1	1	1-2
	GT, Expr	Branch if > 1	1	1-2
	GTU, Expr	Branch if unsigned > 1	1	1-2
	LE, Expr	Branch if <= 1	1	1-2
	LEU, Expr	Branch if unsigned <= 1	1	1-2
	LT, Expr	Branch if <1	1	1-2
	LTU, Expr	Branch if unsigned < 1	1	1-2
	N, Expr	Branch if negative	1	1-2
	NC, Expr	Branch if not carry	1	1-2
	NN, Expr	Branch if not negative	1	1-2
	NOV, Expr	Branch if not overflow	1	1-2
	NZ, Expr	Branch if not zero	1	1-2
	OA, Expr	Branch if Accumulator A overflow	1	1-2
	OB, Expr	Branch if Accumulator B overflow	1	1-2
	OV, Expr	Branch if overflow	1	1-2
	SA, Expr	Branch if Accumulator A saturated	1	1-2
	SB, Expr	Branch if Accumulator B saturated	1	1-2
	Z, Expr	Branch if zero	1	1-2
BSET	f, #bit4	Bit set f	1	1
	Ws, #bit4	Bit set Ws	1	1
BSW.C	Ws, Wb	Write C bit to Ws<Wb>	1	1
BSW.Z	Ws, Wb	Write Z bit to Ws<Wb>	1	1
BTG	f, #bit4	Bit toggle f	1	1
	Ws, #bit4	Bit toggle Ws	1	1
BTSC	f, #bit4	Bit test f, skip if clear	1	1-3
	Ws, #bit4	Bit test Ws, skip if clear	1	1-3
BTSS	f, #bit4	Bit test f, skip if set	1	1-3
	Ws, #bit4	Bit test Ws, skip if set	1	1-3
BTST	f, #bit4	Bit test f	1	1
BTST.C	Ws, #bit4	Bit test Ws to C	1	1
	Ws, Wb	Bit test Ws<Wb> to C	1	1

# Instruction Set (Continued)

Asm	Operand	Description	W	C
BTST.Z	Ws, #bit4	Bit test Ws to Z	1	1
	Ws, Wb	Bit test Ws<Wb> to Z	1	1
BTSTS	f, #bit4	Bit test f then set f	1	1
BTSTS.C	Ws, #bit4	Bit test Ws to C then set Ws	1	1
BTSTS.Z	Ws, #bit4	Bit test Ws to Z then set Ws	1	1
CALL	Expr	Call subroutine	2	2
	Wn	Call indirect subroutine	1	2
CLR	f	f = 0x0000	1	1
	WREG	WREG = 0x0000	1	1
	Wd	Wd = 0x0000	1	1
	Acc, Wx, Wxd, Wy, Wyd, AWB	Clear Accumulator	1	1
CLRWDT		Clear watchdog timer	1	1
COM	f{, WREG}	Destination = f	1	1
	Ws, Wd	Wd = Ws	1	1
CP	f	Compare (f – WREG)	1	1
	Wb, #lit5	Compare (Wb – lit5)	1	1
	Wb, Ws	Compare (Wb – Ws)	1	1
CP0	f	Compare (f – 0)	1	1
	Ws	Compare (Ws – 0)	1	1
CPB	f	Compare w/ Borrow (f – WREG – C)	1	1
	Wb, #lit5	Compare w/ Borrow (Wb – lit5 – C)	1	1
	Wb, Ws	Compare w/ Borrow (Wb – Ws – C)	1	1
CPSEQ	Wb, Ws	Compare (Wb - Ws), skip if =	1	1-3
CPSGT	Wb, Ws	Compare (Wb - Ws), skip if >	1	1-3
CPSLT	Wb, Ws	Compare (Wb - Ws), skip if <	1	1-3
CPSNE	Wb, Ws	Compare (Wb - Ws), skip if <>	1	1-3
DAW.B	Wn	Wn = decrement and adjust Wn	1	1
DEC	f{, WREG}	Destination = f – 1	1	1
	Ws, Wd	Wd = Ws – 1	1	1
DEC2	f{, WREG}	Destination = f – 2	1	1
	Ws, Wd	Wd = Ws – 2	1	1
DISI	#lit14	Disable interrupt for (lit14+1) cycles	1	1
DIV.SW	Wm, Wn	Signed 16/16-bit integer divide	1	18
DIV.SD	Wm, Wn	Signed 32/16-bit integer divide	1	18
DIV.UW	Wm, Wn	Unsigned 16/16-bit integer divide	1	18
DIV.UD	Wm, Wn	Unsigned 32/16-bit integer divide	1	18
DIVF	Wm, Wn	Signed 16/16-bit fractional divide	1	18
DO	#lit14, Expr	Do thru PC+Expr(lit14+1) times	2	2
	Wn, Expr	Do thru PC+Expr(Wn+1) times	2	2
ED	Wm*Wm, Acc, Wx, Wy, Wxd	Euclidean distance (no accumulate)	1	1
EDAC	Wm*Wm, Acc, Wx, Wy, Wxd	Euclidean distance	1	1
EXCH	Wns, Wnd	Swap Wns and Wnd	1	1
FBCL	Ws, Wnd	Find bit change from left (MSb)	1	1
FF1L	Ws, Wnd	Find first 1 from left (MSb)	1	1
FF1R	Ws, Wnd	Find first 1 from right (LSb)	1	1
GOTO	Expr	Go to address	2	2
	Wn	Go to address indirectly	1	2
INC	f{, WREG}	Destination = f + 1	1	1
	Ws, Wd	Wd = Ws + 1	1	1
INC2	f{, WREG}	Destination = f + 2	1	1
	Ws, Wd	Wd = Ws + 2	1	1
IOR	f{, WREG}	Destination = f .IOR. WREG	1	1

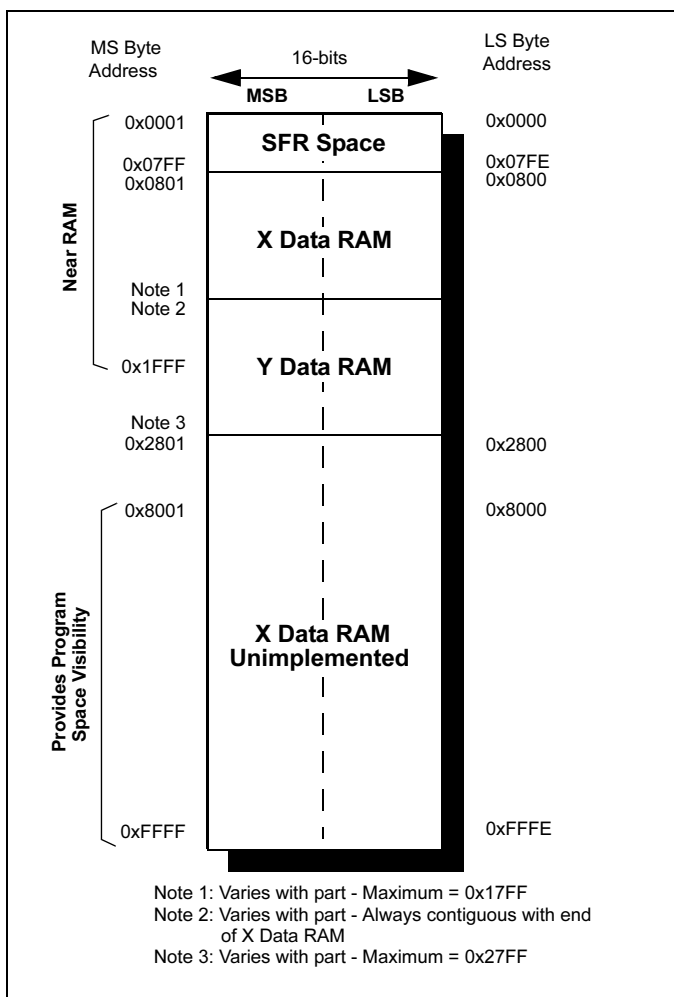
# Instruction Set (Continued)

Asm	Operand	Description	W	C
IOR	#lit10, Wn	Wn = lit10 .IOR. Wn	1	1
	Wb, #lit5, Wd	Wd = Wb .IOR. lit5	1	1
	Wb, Ws, Wd	Wd = Wb .IOR. Ws	1	1
LAC	Ws, #Slit4, Acc	Load Accumulator	1	1
LNK	#lit14	Link frame pointer	1	1
LSR	f{, WREG}	Destination = log right shift f	1	1
	Ws, Wd	Wd = log right shift Ws	1	1
	Wb, #lit4, Wnd	Wn = log right shift Wb by lit4	1	1
	Wb, Wns, Wnd	Wnd = log right shift Wb by Wns	1	1
MAC	Wm*Wn, Acc, Wx, Wxd, Wy, Wyd, AWB	Multiply Wm by Wn to Accumulator	1	1
	Wm*Wm, Acc, Wx, Wxd, Wy, Wyd	Square Wm to Accumulator	1	1
MOV	f{, WREG}	Move f to destination	1	1
	WREG, f	Move WREG to f	1	1
	f, Wnd	Move f to Wnd	1	1
	Wns, f	Move Wns to f	1	1
	#lit16, Wnd	Move 16-bit literal to Wnd	1	1
	[Ws+Slit10], Wnd	Move [Ws + offset] to Wnd	1	1
	Wns, [Wd+Slit10]	Move Wns to [Wd + offset]	1	1
	Ws, Wd	Move Ws to Wd	1	1
MOV.B	#lit8, Wnd	Move 8-bit literal to Wnd	1	1
MOV.D	Ws, Wnd	Move double Ws to Wnd:Wnd+1	1	2
	Wns, Wd	Move double Wns:Wns+1 to Wd	1	2
MOV.SAC	Acc, Wx, Wxd, Wy, Wyd, AWB	Move Wx to Wxd and Wy to Wyd	1	1
MPY	Wm*Wn, Acc, Wx, Wxd, Wy, Wyd	Multiply Wn by Wm to Accumulator	1	1
	Wm*Wm, Acc, Wx, Wxd, Wy, Wyd	Square Wm to Accumulator	1	1
MPY.N	Wm*Wn, Acc, Wx, Wxd, Wy, Wyd	Negative Multiply Wn by Wm to Accumulator	1	1
MSC	Wm*Wn, Acc, Wx, Wxd, Wy, Wyd, AWB	Multiply Wm by Wn and subtract from Accumulator	1	1
MUL	f	W3:W2 = f * WREG	1	1
MUL.SS	Wb, Ws, Wnd	{Wnd+1, Wnd} = sign(Wb)*sign(Ws)	1	1
MUL.SU	Wb, #lit5, Wnd	{Wnd+1, Wnd} = sign(Wb)*uns(lit5)	1	1
	Wb, Ws, Wnd	{Wnd+1, Wnd} = sign(Wb)*uns(Ws)	1	1
MUL.US	Wb, Ws, Wnd	{Wnd+1, Wnd} = uns(Wb)*sign(Ws)	1	1
MUL.UU	Wb, #lit5, Wnd	{Wnd+1, Wnd} = uns(Wb)*uns(lit5)	1	1
	Wb, Ws, Wnd	{Wnd+1, Wnd} = uns(Wb)*uns(Ws)	1	1
NEG	f{, WREG}	Destination = f + 1	1	1
	Ws, Wd	Wd = Ws + 1	1	1
	Acc	Negate Accumulator	1	1
NOP		No operation	1	1
NOPR		No operation	1	1
POP	f	Pop top of stack to f	1	1
	Wd	Pop top of stack to Wd	1	1
POP.D	Wnd	Double pop from top of stack to Wnd:Wnd+1	1	2
POP.S		Pop shadow registers	1	1
PUSH	f	Push f to top of stack	1	1
	Ws	Push Ws to top of stack	1	1
PUSH.D	Wns	Push double Wns:Wns+1 to TOS	1	2
PUSH.S		Push shadow registers	1	1
PWRSV	#lit1	Enter power saving mode lit1	1	1

# Instruction Set (Continued)

Asm	Operand	Description	W	C
RCALL	Expr	Relative call	1	2
	Wn	Computed call	1	2
REPEAT	#lit14	Repeat next inst (lit14+1) times	1	1
	Wn	Repeat next inst (Wn+1) times	1	1
RESET		Software RESET	1	1
RETFIE		Return with interrupt enabled	1	3
RETLW	#lit10, Wn	Return with literal10 in Wn	1	3
RETURN		Return from subroutine	1	3
RLC	f{, WREG}	Destination = rotate left thru Carry f	1	1
	Ws, Wd	Wd = rotate left thru Carry Ws	1	1
RLNC	f{, WREG}	Destination = rotate left (no Carry) f	1	1
	Ws, Wd	Wd = rotate left (no Carry) Ws	1	1
RRC	f{, WREG}	Destination = rotate right thru Carry f	1	1
	Ws, Wd	Wd = rotate right thru Carry Ws	1	1
RRNC	f{, WREG}	Destination = rotate right (no Carry) f	1	1
	Ws, Wd	Wd = rotate right (no Carry) Ws	1	1
SAC	Acc, #Slit4, Wd	Store Accumulator	1	1
SAC.R	Acc, #Slit4, Wd	Store rounded Accumulator	1	1
SE	Ws, Wnd	Wnd = sign-extend Ws	1	1
SETM	f	Set f = 0xFFFF	1	1
	WREG	Set WREG = 0xFFFF	1	1
	Wd	Set Wd = 0xFFFF	1	1
SFTAC	Acc, #Slit6	Arith shift Accumulator by Slit6	1	1
	Acc, Wn	Arith shift Accumulator by (Wn)	1	1
SL	f{, WREG}	Destination = left shift f	1	1
	Ws, Wd	Wd = left shift Ws	1	1
	Wb, #lit4, Wnd	Wnd = left shift Wb by lit4	1	1
	Wb, Wns, Wnd	Wnd = left shift Wb by Wns	1	1
SUB	f{, WREG}	Destination = f – WREG	1	1
	#lit10, Wn	Wn = Wn – lit10	1	1
	Wb, #lit5, Wd	Wd = Wb – lit5	1	1
	Wb, Ws, Wd	Wd = Wb – Ws	1	1
	Acc	Subtract Accumulators	1	1
SUBB	f{, WREG}	Destination = f–WREG–(C)	1	1
	#lit10, Wn	Wn = Wn – lit10–(C)	1	1
	Wb, #lit5, Wd	Wd = Wb – lit5–(C)	1	1
	Wb, Ws, Wd	Wd = Wb – Ws–(C)	1	1
SUBBR	f{, WREG}	Destination = WREG–f–(C)	1	1
	Wb, #lit5, Wd	Wd = lit5 – Wb–(C)	1	1
	Wb, Ws, Wd	Wd = Ws – Wb–(C)	1	1
SUBR	f{, WREG}	Destination = WREG–f	1	1
	Wb, #lit5, Wd	Wd = lit5 – Wb	1	1
	Wb, Ws, Wd	Wd = Ws – Wb	1	1
SWAP	Wn	Wn = byte/nibble swap Wn	1	1
TBLRDH	Ws, Wd	Read high program word to Wd	1	2
TBLRDL	Ws, Wd	Read low program word to Wd	1	2
TBLWTH	Ws, Wd	Write Ws to high program word	1	2
TBLWTL	Ws, Wd	Write Ws to low program word	1	2
ULNK		Unlink frame pointer	1	1
XOR	f{, WREG}	Destination = f .XOR. WREG	1	1
	#lit10, Wn	Wn = lit10 .XOR. Wn	1	1
	Wb, #lit5, Wd	Wd = Wb .XOR. lit5	1	1
	Wb, Ws, Wd	Wd = Wb .XOR. Ws	1	1
ZE	Ws, Wnd	Wnd = zero extend Ws	1	1

# Data Memory Map





# **MICROCHIP**

Microchip Technology Inc.

2355 W. Chandler Blvd.

Chandler, AZ 85224 U.S.A.

Phone: 480-792-7200 • Fax: 480-792-7277

[www.microchip.com](http://www.microchip.com)

The Microchip name and logo, the Microchip logo, dsPIC, MPLAB, PIC and PICmicro are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

© 2003 Microchip Technology Incorporated.

All rights reserved. Printed in the U.S.A. 11/03 DS51322C

