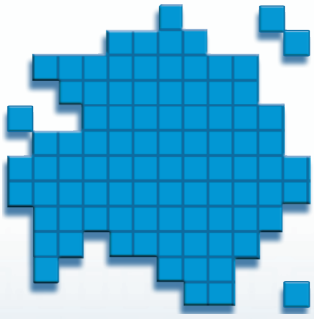


dsPIC30F Language Tools Quick Reference Card

MPLAB CERTIFIED™

dsPIC®

MPLAB® ASM30 Assembler

Assembler Command Line Options

pic30-as [options|sourcefiles]...

Listing Output

Option	Description
-a[*sub-option*] [=*file*] c d h l m n s i	The -a option supports the following suboptions: Omit false conditionals Omit debugging directives Include high-level source Include assembly Include macro expansions Omit forms processing Include symbols 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 *objfile*	Set name of output to *objfile*
-omf={coff|elf}	Set output file type. Coff is default.
--relax	Substitute short branches where possible
--no-relax	Don't substitute short branches (default)
-Z	Generate object file even after errors
-MD *file*	Write dependency information to *file*

Other Options

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

Special Operators

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

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
`.section` *name* `[,`*type*`]` `[,`*attr1*`,`*attr2*`,...,`*attrn*`]`	Set section *name* with *type* and *attr* *type:* code, data, bss, persist, psv, eedata *attr:* address(a), near, xmemory, ymemory, reverse(n), align(n)
`.text`	.section .text,code
`.bss`	.section .bss,bss
`.data`	.section .data,data

Constant Initialization

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

Symbols

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

Section Location

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

Conditional

Directive	Description
.if *expr*	Start of conditional
.ifdef *sym*	Conditional if *sym* defined
.ifndef *sym*	Conditional if *sym* not defined
.else	Alternative to conditional
.endif	End of conditional
.err *msg*	Print message

MPLAB LINK30 Object Linker

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

Command Line Options

Option	Description
-l *libname*	Search for library *libname*
-M	Print map file on standard output
-r	Generate relocatable output
-S	Strip debug symbols
-T *file*	Read linker script
-u *sym*	Start with undefined reference to *sym*
-v	Print version information
-(-l*lib1* ... -l*libn* -)	Resolve circular references in libraries
--no-check-sections	Do not check section addresses overlap
--cref	Output cross reference table
--defsym *sym=expr*	Define symbol
--heap *size*	Set heap to *size* bytes
--help	Print list of command line options
-Map *file*	Write a map file
--no-data-init	Don't create initialized data template
--no-handles	Don't create handle jump table
--no-pack-data	Use only lower 16-bits for data storage
-omf={coff|elf}	Set output file type. Coff is default.
--report-mem	Write memory report to standard output
--stack *size*	Set minimum stack to *size* bytes (default=16)
--warn-once	Warn only once per undefined symbol
--warn-section-align	Warn if alignment changes start of section

Object File Utilities

Executables

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

pic30-objdump Utility

pic30-objdump [*-options*] *file*

Options

Option	Description
-a	Display archive header information
-d	Disassemble sections that contain instructions
-D	Disassemble all sections
-z	Disassemble blocks, even if zeroes
-f	Display summary information from headers
-g	Display debugging information
-h	Display summary information from section headers
-j *name*	Display information only for section *name*
-l	Label display with file names and line numbers
-S	Display source intermixed with disassembly
-M symbolic	Perfom symbolic disassembly
-omf={coff|elf}	Set output file type. Coff is default.
-r	Print relocation entries of file
-s	Display full contents of section(s)
-t	Print symbol table entries of file
-x	Display all available header information same as: -a -f -h -r -t
--start-address=*adr*	Start displaying data at *adr* for -d, -r, -s
--stop-address=*adr*	Stop displaying data at *adr* for -d, -r, -s
-V	Print the version number
-H	Print summary of options

MPLAB LIB30 Archiver/Librarian

pic30-ar *option* [*mods [relpos*] [*count*]] *archive* [*member...*]

Options

Option	Description
d	Delete modules from the archive
p	Print specified members of archive
r	Insert the files *member...* into *archive* (with replacement)
t	Display a table listing the contents of *archive* or files in *member*
x	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 *filename* [*adr*]	**L**oad **P**rogram memory
LF *filename* [*adr*]	**L**oad **F**ile registers
DM [*start*] [*end*]	**D**isplay program **M**emory
DF [*start*] [*end*]	**D**isplay **F**ile registers
DA	**D**isplay **A**ccumulators
DB	**D**isplay **B**reakpoints
DW	**D**isplay **W** registers
DC	**D**isplay p**C** disassembled
DP	**D**isplay **P**rofile
DH	**D**isplay **H**elp on all
DS	**D**isplay **S**tatus register fields
AF *freq*	**A**lter clock **F**requency
MS *start end/val*[*val*]	Program **M**emory **S**et
MC *start* [*end*]	Program **M**emory **C**lear
FS *start end/val* [*val*]	**F**ile register **S**et
FC *start* [*end*]	**F**ile register **C**lear
BS *adr1 ...*[*adrN*]	**B**reakpoint **S**et
BC *adr1 ...*[*adrN*]	**B**reakpoint **C**lear
RP	**R**eset **P**rocessor
RC	**R**eset **C**lock
VO	**V**erbose **O**n
VF	**V**erbose of**F**
IO	Turn simulated **I**/o file **O**n
IF	Turn simulated **I**/o file of**F**
PS *value*	**P**C **S**et
LD *devicename*	**L**oad **D**evice
LC *filename*	**L**oad **C**OFF or ELF file
LS *filename*	**L**oad **S**CL file
Q	**Q**uit
S	**S**tep
E	**E**xecute
H	**H**alt
HE	**H**alt on **E**rror
HW	**H**alt on **W**arning

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 *name*. Example: int x __attribute__ ((section("foo")))=0;
near	Variable is allocated in first 8 KB of data memory
far	Variable can be located anywhere in data memory
space(xmemory)	Variable placed in X memory
space(ymemory)	Variable placed in Y memory
space(auto_psv)	Variable placed in compiler managed PSV memory
space(psv)	Variable placed in PSV memory
space(eedata)	Variable placed in EEDATA memory
address(addr)	Specify absolute address for variable
reverse(n)	Align end on n- aligned boundary
persistent	uninitialized variable not zeroed at startup
unordered	allocate space on best-fit

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
address(addr)	Specify absolute address for function

General Options

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

Memory Model Command Line Options

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

Options for Debugging

Option	Description
`-g`	Produce debugging information
`-omf={coff|elf}`	Set output file type. Coff is default.
`-save-temps`	Do not delete intermediate files

Options for Controlling Optimization

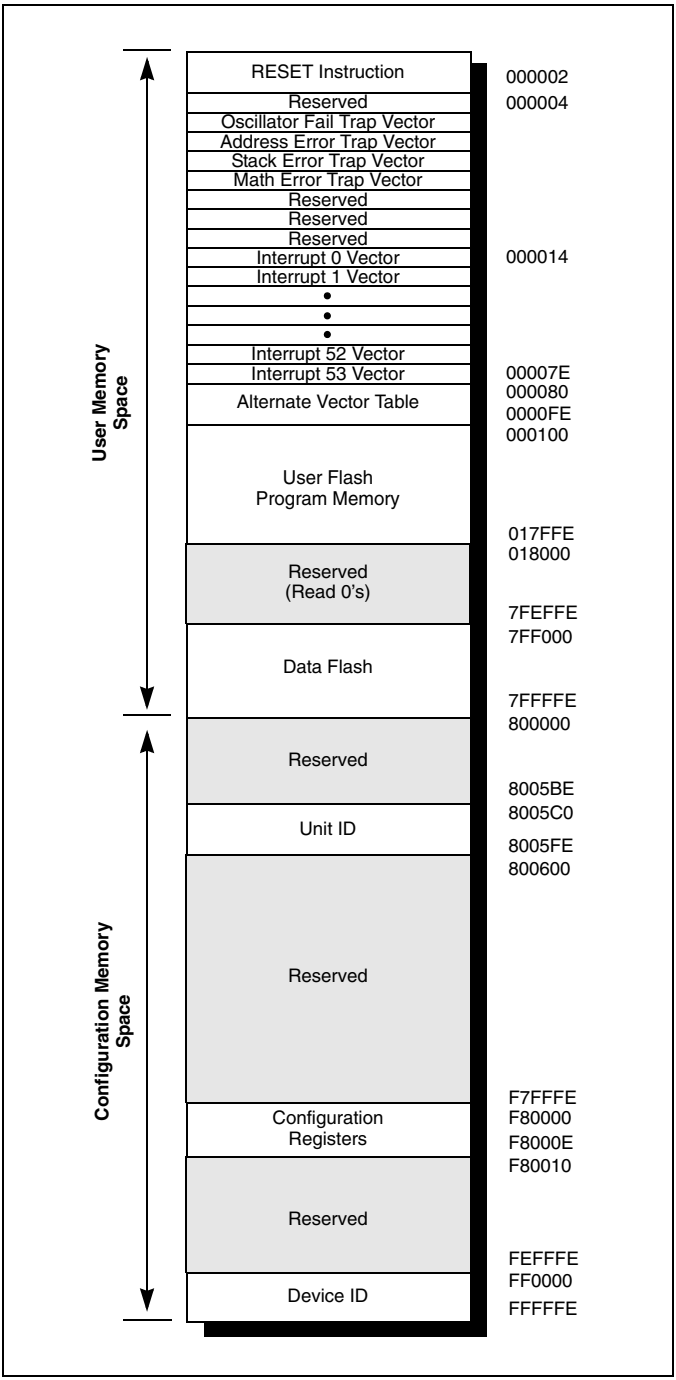
NOTE: Enabling optimizations may affect debugging

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

Built-In Functions

Function
`int __builtin_divsd(const long num, const int den);`
`unsigned __builtin_divud(const unsigned long num, const unsigned den);`
`long __builtin_mulss(const int p0, const int p1);`
`long __builtin_mulsu(const int p0, const unsigned p1);`
`long __builtin_mulus(const unsigned p0, const int p1);`
`unsigned long __builtin_muluu(const unsigned p0, const unsigned p1);`
`unsigned __builtin_tblpage(const void *p);`
`unsigned __builtin_tbloffset(const void *p);`
`unsigned __builtin_psvpage(const void *p);`
`unsigned __builtin_psvoffset(const void *p);`

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

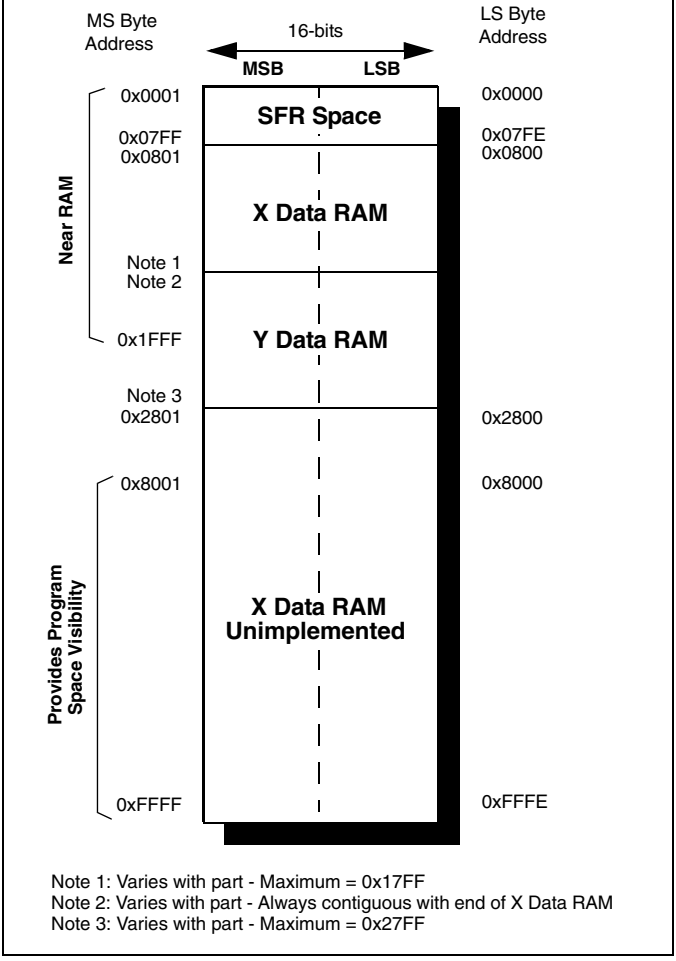
Instruction Set (Continued)

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
MOVSAC	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
MPY	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
PWRSAV	#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



Note 1: Varies with part - Maximum = 0x17FF
Note 2: Varies with part - Always contiguous with end of X Data RAM
Note 3: Varies with part - Maximum = 0x27FF

Microchip Technology Inc.
2355 W. Chandler Blvd. · Chandler, AZ 85224 U.S.A.
Phone: 480-792-7200 · Fax: 480-792-7277
www.microchip.com