Instruction Set Description - Advance Information

1.0 INSTRUCTION SET SUMMARY

The dsPIC™ is a full 16-bit data path MCU and DSP architecture. The dsPIC30F architecture represents the initial product offering for Microchip's DSC (Digital Signal Controller) solutions for the controls market. The dsPIC instruction set adds many enhancements to the previous PICmicro® microcontroller (MCU) instruction sets, while maintaining an easy migration from these PICmicro MCU instruction sets.

1.1 Instruction Set Overview

Table 1-3 lists the instruction set in alphabetical order using the mnemonics as will be presented in the data book. There are 94 instructions and 11 possible addressing modes, as shown in Table 1-1.

Many of the instructions have several addressing modes and consequently, different execution flows. Table 1-3 also lists the syntax of each instruction and applicable addressing mode and gives an example usage of that syntax. Symbols used in the definitions of Tables 1-3 and 1-4 are defined in Table 1-2.

1.2 Functional Overview

Table 1-4 lists the instruction set in a functional grouping. The instruction set is divided into the following nine functional groups, with subgroups as needed. Each group (or subgroup) contains instructions, arranged alphabetically.

Functional Groups/Subgroup .....Page

TABLE 1-1: dsPIC30F ADDRESSING MODE

	Addressing Mode Syntax	Addressing Mode Description
1	f	Memory Direct (8K byte address range)
2	Slit	Signed literal (constant)
3	Wn	Register Direct
4	[Wn]	Register Indirect
5	[Wn]--	Register Indirect with Post-Decrement
6	[Wn]++	Register Indirect with Post-Increment
7	[Wn--]	Register Indirect with Pre-Decrement
8	[Wn++]	Register Indirect with Pre-Increment
9	[Wn + Wb]	Register Indirect with Register Offset
10	[Wn + Slit]	Register Indirect with Signed Literal Offset
11	Slit16	Relative (16-bit signed offset)

FIGURE 1-1: PROGRAMMERS MODEL FOR dsPIC30

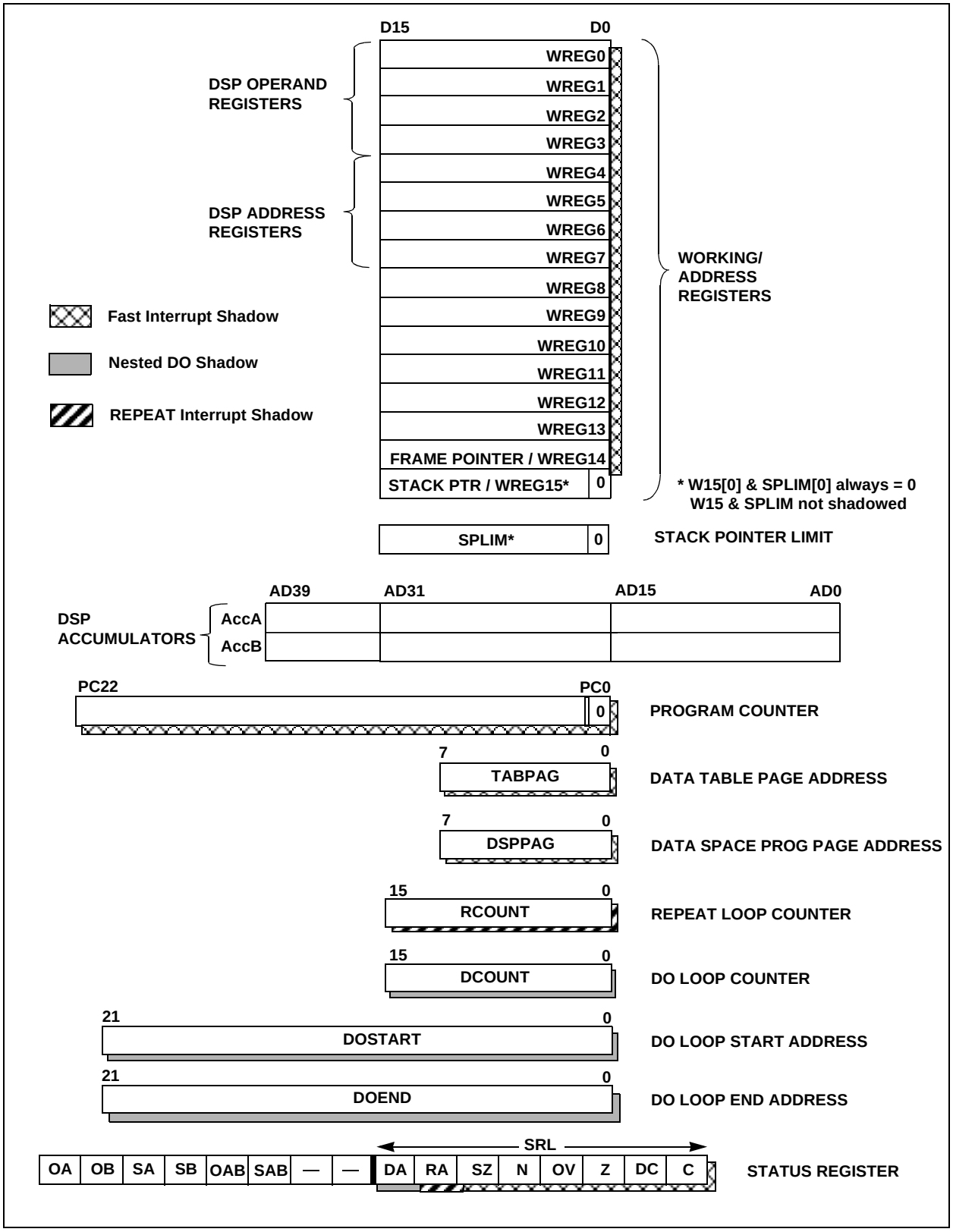


TABLE 1-2: SYMBOLS USED IN dsPIC30F OPCODE DESCRIPTIONS

Field	Description
{ }	Optional field or operation
[text]	Means "the location addressed by text"
(text)	Means "content of text"
#text	Means literal defined by "text"
text1 ∈ {text2, text3, ...}	text1 must be in the set of text2, text3, ...
none	field does not require an entry, may be blank
{label:}	Optional Label name
label	Translates to a literal representing the location of the label name
<n:m>	Register bit field
lit1	1-bit unsigned literal ∈ {0,1}
lit4	4-bit unsigned literal ∈ {0...15}
lit5	5-bit unsigned literal ∈ {0...31}
Slit5	5-bit signed literal ∈ {-16...15}
Slit10	10-bit signed literal ∈ {-512...511}
lit14	14-bit unsigned literal ∈ {0...16383}
lit16	16-bit unsigned literal ∈ {0...65535}
Slit16	16-bit signed literal ∈ {-32768...32767}
lit23	23-bit unsigned literal ∈ {0...8388607}; LSB must be 0
bit3	3-bit bit selection field (used in byte addressed instructions) ∈ {0...7}
bit4	4-bit bit selection field (used in word addressed instructions) ∈ {0...15}
.w	Word mode selection (default)
.b	Byte mode selection
.S	Shadow register select
f	File register address ∈ {0000h...1FFFh}
d	File register destination d ∈ {Ww, none}
Ww	Default W working register (used in file register instructions)
Wn	One of 16 working registers ∈ {W0..W15}
Wns	One of 16 source working registers ∈ {W0..W15}
Wnd	One of 16 destination working registers ∈ {W0..W15}
Wb	Base W register ∈ {W0..W15}
Ws	Source W register ∈ { Ws, [Ws], [Ws]++, [Ws]--, [Ws++] }
Wd	Destination W register ∈ { Wd, [Wd], [Wd]++, [Wd]--, [Wd++] }
Wso	Source W register ∈ { Wns, [Wns], [Wns]++, [Wns]--, [Wns--], [Wns+Wb], [Wns+slit5] }
Wdo	Destination W register ∈ { Wnd, [Wnd], [Wnd]++, [Wnd]--, [Wnd--], [Wnd+Wb], [Wnd+slit5] }
A	Accumulators ∈ {A, B}

Field	Description
Wm*Wm	Multiplicand and Multiplier W register for Square instructions ∈ {W0*W0,W1*W1,W2*W2,W3*W3}
Wm*Wn	Multiplicand and Multiplier W register for DSP instructions ∈ {W0*W1,W0*W2,W0*W3,W1*W2,W1*W3,W2*W3}
Wx	X data space prefetch address register for DSP instructions ∈ {[W4]+=6, [W4]+=4, [W4]+=2, [W4], [W4]-=6, [W4]-=4, [W4]-=2, [W5]+=6, [W5]+=4, [W5]+=2, [W5], [W5]-=6, [W5]-=4, [W4]-=2, [W5+W8],none}
Wy	Y data space prefetch address register for DSP instructions ∈ {[W6]+=8, [W6]+=4, [W6]+=2, [W6], [W6]-=6, [W6]-=4, [W6]-=2, , [W7]+=8, [W7]+=4, [W7]+=2, [W7], [W7]-=6, [W7]-=4, [W7]-=2, , [W7+W8],none}
Wxp	X data space prefetch destination register for DSP instructions ∈ {W0..W3}
Wyp	Y data space prefetch destination register for DSP instructions ∈ {W0..W3}
AWB	Accumulator write back destination address register ∈ {W9,[W9]++}
PC	Program Counter
PCL	Program Counter Low Byte
PCH	Program Counter High Byte
PCU	Program Counter Upper Byte
PCLATH	Program Counter High Byte Latch
PCLATU	Program Counter Upper Byte Latch
OA, OB, SA, SB	DSC status bits: ACCA Overflow, ACCB Overflow, ACCA Saturate, ACCB Saturate
C, DC, N, OV, SZ, Z	ALU status bits: Carry, Digit Carry, Negative, Overflow, Sticky-Zero, Zero

TABLE 1-3: dsPIC30F INSTRUCTION SET - APHABETICALLY

Instr #	Assembly Mnemonic	Assembly Syntax		Example		Description
1	ADD	ADD	A	ADD	A	Add Accumulators
		ADD	f	ADD	RAM100	f = f + Ww
		ADD	f,Ww	ADD	RAM100,Ww	Ww = f + Ww
		ADD	Slit10,Wn	ADD	#0xAA,W11	Wd = Slit10 + Wd
		ADD	Wb,Ws,Wd	ADD	W7,[W12++],[W11]--	Wd = Wb + Ws
		ADD	Wb,lit5,Wd	ADD	W7,#25,[W11]--	Wd = Wb + lit5
		ADD	A,Wso,Slit4	ADD	A,[W12++],#6	16-bit Signed Add to Accumulator
2	ADDC	ADDC	f	ADDC	RAM100	f = f + Ww + (C)
		ADDC	f,Ww	ADDC	RAM100,Ww	Ww = f + Ww + (C)
		ADDC	Slit10,Wn	ADDC	#0xAA,W11	Wd = Slit10 + Wd + (C)
		ADDC	Wb,Ws,Wd	ADDC	W7,[W12++],[W11]--	Wd = Wb + Ws + (C)
		ADDC	Wb,lit5,Wd	ADDC	W7,#25,[W11]--	Wd = Wb + lit5 + (C)
3	AND	AND	f	AND	RAM100	f = f .AND. Ww
		AND	f,Ww	AND	RAM100,Ww	Ww = f .AND. Ww
		AND	Slit10,Wn	AND	#0xAA,W11	Wd = Slit10 .AND. Wd
		AND	Wb,Ws,Wd	AND	W7,[W12++],[W11]--	Wd = Wb .AND. Ws
		AND	Wb,lit5,Wd	AND	W7,#25,[W11]--	Wd = Wb .AND. lit5
4	ASR	ASR	f	ASR	RAM100	f = Arithmetic Right Shift f
		ASR	f,Ww	ASR	RAM100,Ww	Ww = Arithmetic Right Shift f
		ASR	Ws,Wd	ASR	[W12++],[W11]--	Wd = Arithmetic Right Shift Ws
		ASR	Wd,Wns,Wnd	ASR	W0,W2,W1	Wnd = Arithmetic Right Shift Wb by Wns
		ASR	Wd,lit5,Wnd	ASR	W0,#23,W1	Wnd = Arithmetic Right Shift Ws by lit5
5	BCLR	BCLR.b	f,bit3	BCLR.b	RAM100,#5	Bit Clear f
		BCLR	Ws,bit4	BCLR	[W12++],#9	Bit Clear Ws

TABLE 1-3: dsPIC30F INSTRUCTION SET - APHABETICALLY (CONTINUED)

Instr #	Assembly Mnemonic	Assembly Syntax		Example		Description
6	BRA	BRA	C,Slit16	BRA	C,label	Branch if Carry
		BRA	GE,Slit16	BRA	GE,label	Branch if greater than or equal
		BRA	GEU,Slit16	BRA	GEU,label	Branch if unsigned greater than or equal
		BRA	GT,Slit16	BRA	GT,label	Branch if greater than
		BRA	GTU,Slit16	BRA	GTU,label	Branch if unsigned greater than
		BRA	LE,Slit16	BRA	LE,label	Branch if less than or equal
		BRA	LEU,Slit16	BRA	LEU,label	Branch if unsigned less than or equal
		BRA	LT,Slit16	BRA	LT,label	Branch if less than
		BRA	LTU,Slit16	BRA	LTU,label	Branch if unsigned less than
		BRA	N,Slit16	BRA	N,label	Branch if Negative
		BRA	NC,Slit16	BRA	NC,label	Branch if Not Carry
		BRA	NN,Slit16	BRA	NN,label	Branch if Not Negative
		BRA	NOV,Slit16	BRA	NOV,label	Branch if Not Overflow
		BRA	NZ,Slit16	BRA	NZ,label	Branch if Not Zero
		BRA	OA,Slit16	BRA	OA,label	Branch if accumulator A overflow
		BRA	OB,Slit16	BRA	OB,label	Branch if accumulator B overflow
		BRA	OV,Slit16	BRA	OV,label	Branch if Overflow
		BRA	SA,Slit16	BRA	SA,label	Branch if accumulator A saturated
		BRA	SB,Slit16	BRA	SB,label	Branch if accumulator B saturated
		BRA	Slit16	BRA	label	Branch Unconditionally
		BRA	Z,Slit16	BRA	Z,label	Branch if Zero
		BRA	Wn	BRA	W11	Computed Branch
7	BSET	BSET.b	f,bit3	BSET.b	RAM100,#5	Bit Set f
		BSET	Ws,bit4	BSET	[W12++],#9	Bit Set Ws
8	BSW	BSW.C	Ws,Wb	BSW.C	[W12++],W7	Write C or Z bit to Ws<Wb>
		BSW.Z	Ws,Wb	BSW.Z	[W12++],W7	Write C or Z bit to Ws<Wb>
9	BTG	BTG.b	f,bit3	BTG.b	RAM100,#5	Bit Toggle f
		BTG	Ws,bit4	BTG	[W12++],#9	Bit Toggle Ws
10	BTSC	BTSC.b	f,bit3	BTSC.b	RAM100,#5	Bit Test f, Skip if Clear
		BTSC	Ws,bit4	BTSC	[W12++],#9	Bit Test Ws, Skip if Clear
11	BTSS	BTSS.b	f,bit3	BTSS.b	RAM100,#5	Bit Test f, Skip if Set
		BTSS	Ws,bit4	BTSS	[W12++],#9	Bit Test Ws, Skip if Set
12	BTST	BTST.b	f,bit3	BTST.b	RAM100,#5	Bit Test f
		BTST.C	Ws,bit4	BTST.C	[W12++],#9	Bit Test Ws to C or Z
		BTST.Z	Ws,bit4	BTST.Z	[W12++],#9	Bit Test Ws to C or Z
		BTST.C	Ws,Wb	BTST.C	[W12++],W7	Bit Test Ws<Wb> to C or Z
		BTST.Z	Ws,Wb	BTST.Z	[W12++],W7	Bit Test Ws<Wb> to C or Z

TABLE 1-3: dsPIC30F INSTRUCTION SET - APHABETICALLY (CONTINUED)

Instr #	Assembly Mnemonic	Assembly Syntax		Example		Description
13	BTSTS	BTSTS.b	f,bit3	BTSTS.b	RAM100,#5	Bit Test then Set f
		BTSTS.C	Ws,bit4	BTSTS.C	[W12++],#9	Bit Test Ws to C or Z then Set
		BTSTS.Z	Ws,bit4	BTSTS.Z	[W12++],#9	Bit Test Ws to C or Z then Set
14	CALL	CALL	lit23	CALL	label	Call subroutine
		CALL.S	lit23	CALL.S	label	Call subroutine
		CALL	Wn	CALL	W11	Call indirect subroutine
		CALL.S	Wn	CALL.S	W11	Call indirect subroutine
15	CLR	CLR	f	CLR	RAM100	f = 0x0000
		CLR	Ww	CLR	Ww	Ww = 0x0000
		CLR	Ws	CLR	[W11]--	Ws = 0x0000
		CLR	A,Wxp,Wx,Wyp,Wy,AWB	CLR	A,W0,[W5]+=4,W1,[W7]-=2,W9	Clear Accumulator
16	CLRWDT	CLRWDT		CLRWDT		Clear Watchdog Timer
17	COM	COM	f	COM	RAM100	f = $\overline{f}$
		COM	f,Ww	COM	RAM100,Ww	Ww = $\overline{f}$
		COM	Ws,Wd	COM	[W12++],[W11]--	Wd = $\overline{Ws}$
18	CP	CP	f	CP	RAM100	Compare f with Ww
		CP	Wb,lit5	CP	W7,#25	CompareWb with lit5
		CP	Wb,Ws	CP	W7,[W12++]	Compare Wb with Ws
19	CP0	CP0	f	CP0	RAM100	Compare f with 0x0000
		CP0	Ws	CP0	[W11]--	Compare Ws with 0x0000
20	CP1	CP1	f	CP1	RAM100	Compare f with 0xFFFF
		CP1	Ws	CP1	[W11]--	Compare Ws with 0xFFFF
21	CPB	CPB	f	CPB	RAM100	Compare Borrow f with Ww
		CPB	Wb,lit5	CPB	W7,#25	Compare Borrow Wb with lit5
		CPB	Wb,Ws	CPB	W7,[W12++]	Compare Borrow Wb with Ws
22	CPFSEQ	CPFSEQ	f	CPFSEQ	RAM100	Compare f with Ww, skip if =
23	CPFSGT	CPFSGT	f	CPFSGT	RAM100	Compare f with Ww, skip if >
24	CPFSLT	CPFSLT	f	CPFSLT	RAM100	Compare f with Ww, skip if <
25	CPFSNE	CPFSNE	f	CPFSNE	RAM100	Compare f with Ww, skip if ≠
26	DAW.B	DAW.B	Wn	DAW.B	W11	Wn = decimal adjust Wn
27	DEC	DEC	f	DEC	RAM100	f = f -1
		DEC	f,Ww	DEC	RAM100,Ww	Ww = f -1
		DEC	Ws,Wd	DEC	[W12++],[W11]--	Wd = Ws - 1
28	DEC2	DEC2	Ws,Wd	DEC2	[W12++],[W11]--	Wd = Ws - 2
29	DECSNZ	DECSNZ	f	DECSNZ	RAM100	f = f-1, Skip if Not 0
		DECSNZ	f,Ww	DECSNZ	RAM100,Ww	Ww = f-1, Skip if Not 0
30	DECSZ	DECSZ	f	DECSZ	RAM100	f = f-1, Skip if 0
		DECSZ	f,Ww	DECSZ	RAM100,Ww	Ww = f-1, Skip if 0

TABLE 1-3: dsPIC30F INSTRUCTION SET - APHABETICALLY (CONTINUED)

Instr #	Assembly Mnemonic	Assembly Syntax		Example		Description
31	DISI	DISI	lit14	DISI	#157	Disable Interrupts for k instruction cycles
32	DIV	DIV		DIV		Divide Helper
33	DO	DO	Slit16,lit14	DO	label,#157	Do code to PC+Slit16, lit14 times
		DO	Slit16,Wn	DO	label,W3	Do code to PC+Slit16, (Wn) times
34	ED	ED	A,Wm*Wm,Wxp,Wx,Wy	ED	A,W2*W2,W0,[W5]+=4,[W7]-=2	Euclidean Distance
35	EDAC	EDAC	A,Wm*Wm,Wxp,Wx,Wy,AWB	EDAC	A,W2*W2,W0,[W5]+=4,[W7]-=2,[W9]++	Euclidean Distance Accumulate
36	EXCH	EXCH	Wns,Wnd	EXCH	W12,W11	Swap Wns with Wnd
37	FBCL	FBCL	Ws,Wd	FBCL	[W12++],[W11]--	Find Bit Change from Left (MSb) Side
38	FBCR	FBCR	Ws,Wd	FBCR	[W12++],[W11]--	Find Bit Change from Right (LSb) Side
39	FF0L	FF0L	Ws,Wd	FF0L	[W12++],[W11]--	Find First Zero from Left (MSb) Side
40	FF0R	FF0R	Ws,Wd	FF0R	[W12++],[W11]--	Find First Zero from Right (LSb) Side
41	FF1L	FF1L	Ws,Wd	FF1L	[W12++],[W11]--	Find First One from Left (MSb) Side
42	FF1R	FF1R	Ws,Wd	FF1R	[W12++],[W11]--	Find First One from Right (LSb) Side
43	GOTO	GOTO	lit23	GOTO	label	Go to address
		GOTO	Wn	GOTO	W11	Go to indirect
44	HALT	HALT		HALT		No Operation/ HALT
45	INC	INC	f	INC	RAM100	f = f + 1
		INC	f,Ww	INC	RAM100,Ww	Ww = f + 1
		INC	Ws,Wd	INC	[W12++],[W11]--	Wd = Ws + 1
46	INC2	INC2	Ws,Wd	INC2	[W12++],[W11]--	Wd = Ws + 2
47	INCSNZ	INCSNZ	f	INCSNZ	RAM100	f = f+1, Skip if Not 0
		INCSNZ	f,Ww	INCSNZ	RAM100,Ww	Ww = f+1, Skip Not if 0
48	INCSZ	INCSZ	f	INCSZ	RAM100	f = f+1, Skip if 0
		INCSZ	f,Ww	INCSZ	RAM100,Ww	Ww = f+1, Skip if 0
49	IOR	IOR	f	IOR	RAM100	f = f .IOR. Ww
		IOR	f,Ww	IOR	RAM100,Ww	Ww = f .IOR. Ww
		IOR	Slit10,Wn	IOR	#0xAA,W11	Wd = Slit10 .IOR. Wd
		IOR	Wb,Ws,Wd	IOR	W7,[W12++],[W11]--	Wd = Wb .IOR. Ws
		IOR	Wb,lit5,Wd	IOR	W7,#25,[W11]--	Wd = Wb .IOR. lit5
50	LAC	LAC	A,Wso,Slit4	LAC	A,[W12+6],#5	Load Accumulator
51	LNK	LNK	lit14	LNK	#157	Link frame pointer
52	LSR	LSR	f	LSR	RAM100	f = Logical Right Shift f
		LSR	f,Ww	LSR	RAM100,Ww	Ww = Logical Right Shift f
		LSR	Ws,Wd	LSR	[W12++],[W11]--	Wd = Logical Right Shift Ws
		LSR	Wb,Wns,Wnd	LSR	W0,W2,W1	Wnd = Logical Right Shift Wb by Wns
		LSR	Wb,lit5,Wnd	LSR	W0,#23,W1	Wnd = Logical Right Shift Wb by lit5
53	MAC	MAC	A,Wm*Wn,Wxp,Wx,Wyp,Wy,AWB	MAC	A,W2*W3,W0,[W5]+=4,W1,[W7]-=2,W9	Multiply and Accumulate
		MAC	A,Wm*Wm,Wxp,Wx,Wyp,Wy,AWB	MAC	A,W2*W2,W0,[W5]+=4,W1,[W7]-=2,W9	Square and Accumulate

TABLE 1-3: dsPIC30F INSTRUCTION SET - APHABETICALLY (CONTINUED)

Instr #	Assembly Mnemonic	Assembly Syntax		Example		Description
54	MOV	MOV	f,Wn	MOV	RAM100,W12	Move f to Wn
		MOV	f	MOV	RAM100	Move f to f
		MOV	f,Ww	MOV	RAM100,Ww	Move f to Ww
		MOV	lit16,Wn	MOV	#0x5A5A,W11	Move 16-bit literal to Wn
		MOV	Slit10,Wn	MOV	#0xAA,W11	Move 10-bit signed literal to Wn
		MOV	Wn,f	MOV	W12,RAM100	Move Wn to f
		MOV	Wso,Wdo	MOV	[W12+W3],[W11++]	Move Ws to Wd
		MOV	Ww,f	MOV	Ww,RAM100	Move Ww to f
		MOV.D	Wns,Wd	MOV.D	W12,[W11]--	Move W(ns):W(ns+1) to Wd
		MOV.Q	Wns,Wd	MOV.Q	W12,[W11]--	Move W(ns):W(ns+1):W(ns+2):W(ns+3) to Wd
		MOV.D	Ws,Wnd	MOV.D	[W14++],W12	Move Ws to W(nd+1):W(nd)
		MOV.Q	Ws,Wnd	MOV.Q	[W14++],W12	Move Ws to W(nd+3):W(nd+2):W(nd+1):W(nd)
55	MOVSAC	MOVSAC	A,Wxp,Wx,Wyp,Wy,AWB	MOVSAC	B,W3,[W5],W2,[W7],[W9]++	Move Special
56	MPY	MPY	A,Wm*Wn,Wxp,Wx,Wyp,Wy	MPY	A,W0*W1,W0,[W4]+=4	Multiply Wm by Wn to Accumulator
		MPY	A,Wm*Wm,Wxp,Wx,Wyp,Wy	MPY	A,W1*W1,W0,[W4]+=4	Square Wm to Accumulator
57	MPYN	MPYN	A,Wm*Wn,Wxp,Wx,Wyp,Wy	MPYN	B,W1*W2,W1,[W4],W2,[W6]+=2	-(Multiply Wm by Wn) to Accumulator
58	MSL	MSL	Wb,Wns,Wnd	MSL	W0,W2,W1	Wnd = Multi-word Left Shift Wb by Wns
		MSL	Wb,lit5,Wnd	MSL	W0,#23,W1	Wnd = Multi-word Left Shift Wb by lit5
59	MSR	MSR	Wb,Wns,Wnd	MSR	W0,W2,W1	Wnd = Multi-word Right Shift Wb by Wns
		MSR	Wb,lit5,Wnd	MSR	W0,#23,W1	Wnd = Multi-word Right Shift Wb by lit5
60	MSC	MSC	A,Wm*Wn,Wxp,Wx,Wyp,Wy,AWB	MSC	A,W2*W3,W0,[W5]+=4,W1,[W6],W9	Multiply and Subtract from Accumulator
61	MUL	MUL.SS	Wb,Ws,Wnd	MUL.SS	W7,[W12++],W11	{Wd+1, Wd} = signed(Wb) * signed(Ws)
		MUL.SU	Wb,Ws,Wnd	MUL.SU	W7,[W12++],W11	{Wd+1, Wd} = signed(Wb) * unsigned(Ws)
		MUL.US	Wb,Ws,Wnd	MUL.US	W7,[W12++],W11	{Wd+1, Wd} = unsigned(Wb) * signed(Ws)
		MUL.UU	Wb,Ws,Wnd	MUL.UU	W7,[W12++],W11	{Wd+1, Wd} = unsigned(Wb) * unsigned(Ws)
		MUL.SU	Wb,lit5,Wnd	MUL.US	W7,#25,W11	{Wd+1, Wd} = signed(Wb) * unsigned(lit5)
		MUL.UU	Wb,lit5,Wnd	MUL.UU	W7,#25,W11	{Wd+1, Wd} = unsigned(Wb) * unsigned(lit5)
		MUL	f	MUL	RAM100	W3:W2 = f * Ww
62	NEG	NEG	A	NEG	B	Negate Accumulator
		NEG	f	NEG	RAM100	$f = \overline{f} + 1$
		NEG	f,Ww	NEG	RAM100,Ww	$Ww = \overline{f} + 1$
		NEG	Ws,Wd	NEG	[W12++],[W11]--	$Wd = \overline{Ws} + 1$
63	NOP	NOP		NOP		No Operation
		NOPR		NOPR		No Operation

TABLE 1-3: dsPIC30F INSTRUCTION SET - APHABETICALLY (CONTINUED)

Instr #	Assembly Mnemonic	Assembly Syntax		Example		Description
64	POP	POP	f	POP	RAM100	Pop f from top of stack (TOS)
		POP.S		POP.S		Pop Shadow Registers
		POP	Wdo	POP	[W11+6]	Pop Wd Registers
		POP.D	Wnd	POP.D	W12	Pop W(nd+1):W(nd) Registers
		POP.Q	Wnd	POP.Q	W12	Pop W(nd+3):W(nd+2):W(nd+1):W(nd) Registers
65	PUSH	PUSH	f	PUSH	RAM100	Push f to top of stack (TOS)
		PUSH.S		PUSH.S		Push Shadow Registers
		PUSH	Wso	PUSH	[W12+W3]	Push Ws Registers
		PUSH.D	Wns	PUSH.D	W12	Push W(ns):W(ns+1) Registers
		PUSH.Q	Wns	PUSH.Q	W12	Push W(ns):W(ns+1):W(ns+2):W(ns+3) Registers
66	RCALL	RCALL	Slit16	RCALL	label	Relative Call
		RCALL	Wn	RCALL	W11	Computed Call
67	REPEAT	REPEAT	lit14	REPEAT	#157	Repeat Next Instruction lit14 times
		REPEAT	Wn	REPEAT	W11	Repeat Next Instruction (Wn) times
68	RESET	RESET		RESET		Software device RESET
69	RETFIE RETFIE.S	RETFIE RETFIE.S		RETFIE RETFIE.S		Return from interrupt enable
70	RETLW	RETLW RETLW.S	Slit10,Wn	RETLW	#0xAA,W11	Return with literal in Wn
71	RETURN RETURN.S	RETURN RETURN.S		RETURN RETURN.S		Return from Subroutine
72	RLC	RLC	f	RLC	RAM100	f = Rotate Left through Carry f
		RLC	f,Ww	RLC	RAM100,Ww	Ww = Rotate Left through Carry f
		RLC	Ws,Wd	RLC	[W12++],[W11]--	Wd = Rotate Left through Carry Ws
73	RLNC	RLNC	f	RLNC	RAM100	f = Rotate Left (No Carry) f
		RLNC	f,Ww	RLNC	RAM100,Ww	Ww = Rotate Left (No Carry) f
		RLNC	Ws,Wd	RLNC	[W12++],[W11]--	Wd = Rotate Left (No Carry) Ws
74	RRC	RRC	f	RRC	RAM100	f = Rotate Right through Carry f
		RRC	f,Ww	RRC	RAM100,Ww	Ww = Rotate Right through Carry f
		RRC	Ws,Wd	RRC	[W12++],[W11]--	Wd = Rotate Right through Carry Ws
75	RRNC	RRNC	f	RRNC	RAM100	f = Rotate Right (No Carry) f
		RRNC	f,Ww	RRNC	RAM100,Ww	Ww = Rotate Right (No Carry) f
		RRNC	Ws,Wd	RRNC	[W12++],[W11]--	Wd = Rotate Right (No Carry) Ws
76	SAC	SAC	A,Wdo,Slit4	SAC	A,[W11+W3],#5	Store Accumulator
		SAC.R	A,Wdo,Slit4	SAC.R	A,[W11+W3],#5	Store Rounded Accumulator
77	SE	SE	Ws,Wd	SE	[W12++],[W11]--	Wd = sign extended Ws
78	SETM	SETM	f	SETM	RAM100	f = 0xFFFF
		SETM	Ww	SETM	Ww	Ww = 0xFFFF
		SETM	Ws	SETM	[W11]--	Ws = 0xFFFF

TABLE 1-3: dsPIC30F INSTRUCTION SET - APHABETICALLY (CONTINUED)

Instr #	Assembly Mnemonic	Assembly Syntax		Example		Description
79	SFTAC	SFTAC	A,Wn	SFTAC	A,W12	Arithmetic Shift by (Wn) Accumulator
		SFTAC	A,Slit5	SFTAC	A,#5	Arithmetic Shift by Slit5 Accumulator
80	SL	SL	f	SL	RAM100	f = Left Shift f
		SL	f,Ww	SL	RAM100,Ww	Ww = Left Shift f
		SL	Ws,Wd	SL	[W12++],[W11]--	Wd = Left Shift Ws
		SL	Wd,Wns,Wnd	SL	W0,W2,W1	Wnd = Left Shift Wb by Wns
		SL	Wd,lit5,Wnd	SL	W0,#23,W1	Wnd = Left Shift Ws by lit5
81	SLEEP	SLEEP	lit4	SLEEP	#5	Go into standby mode
82	SUB	SUB	A	SUB	B	Subtract Accumulators
		SUB	f	SUB	RAM100	f = f - Ww
		SUB	f,Ww	SUB	RAM100,Ww	Ww = f - Ww
		SUB	Slit10,Wn	SUB	#0xAA,W11	Wd = Slit10 - Wd
		SUB	Wb,Ws,Wd	SUB	W7,[W12++],[W11]--	Wd = Wb - Ws
		SUB	Wb,lit5,Wd	SUB	W7,#25,[W11]--	Wd = Wb - lit5
83	SUBB	SUBB	f	SUBB	RAM100	f = f - Ww - ($\overline{C}$)
		SUBB	f,Ww	SUBB	RAM100,Ww	Ww = f - Ww - ($\overline{C}$)
		SUBB	Slit10,Wn	SUBB	#0xAA,W11	Wd = Slit10 - Wd - ($\overline{C}$)
		SUBB	Wb,Ws,Wd	SUBB	W7,[W12++],[W11]--	Wd = Wb - Ws - ($\overline{C}$)
		SUBB	Wb,lit5,Wd	SUBB	W7,#25,[W11]--	Wd = Wb - lit5 - ($\overline{C}$)
84	SUBR	SUBR	f	SUBR	RAM100	f = Ww - f
		SUBR	f,Ww	SUBR	RAM100,Ww	Ww = Ww - f
		SUBR	Wb,Ws,Wd	SUBR	W7,[W12++],[W11]--	Wd = Wb - Ws
		SUBR	Wb,lit5,Wd	SUBR	W7,#25,[W11]--	Wd = Wb - lit5
85	SUBBR	SUBBR	f	SUBBR	RAM100	f = Ww - f - ($\overline{C}$)
		SUBBR	f,Ww	SUBBR	RAM100,Ww	Ww = Ww -f - ($\overline{C}$)
		SUBBR	Wb,Ws,Wd	SUBBR	W7,[W12++],[W11]--	Wd = Ws - Wb - ($\overline{C}$)
		SUBBR	Wb,lit5,Wd	SUBBR	W7,#25,[W11]--	Wd = lit5 - Wb - ($\overline{C}$)
86	SWAP	SWAP.b	Wn	SWAP.b	W11	Wn = nibble swap Wn
		SWAP	Wn	SWAP	W11	Wn = byte swap Wn
87	TBLRDH	TBLRDH	Ws,Wd	TBLRDH	[W12++],[W11]--	Read Prog<23:16> to Wd
88	TBLRDL	TBLRDL	Ws,Wd	TBLRDL	[W12++],[W11]--	Read Prog<15:0> to Wd
89	TBLWTH	TBLWTH	Ws,Wd	TBLWTH	[W12++],[W11]--	Write Ws to Prog<23:16>
90	TBLWTL	TBLWTL	Ws,Wd	TBLWTL	[W12++],[W11]--	Write Ws to Prog<15:0>
91	TRAP	TRAP	lit1,lit16	TRAP	0,#157	Trap to vector with literal
92	ULNK	ULNK		ULNK		Unlink frame pointer

TABLE 1-3: dsPIC30F INSTRUCTION SET - APHABETICALLY (CONTINUED)

Instr #	Assembly Mnemonic	Assembly Syntax		Example		Description
93	XOR	XOR	f	XOR	RAM100	f = f .XOR. Ww
		XOR	f,Ww	XOR	RAM100,Ww	Ww = f .XOR. Ww
		XOR	Slit10,Wn	XOR	#0xAA,W11	Wd = Slit10 .XOR. Wd
		XOR	Wb,Ws,Wd	XOR	W7,[W12++],[W11]--	Wd = Wb .XOR. Ws
		XOR	Wb,lit5,Wd	XOR	W7,#25,[W11]--	Wd = Wb .XOR. lit5
94	ZE	ZE	Ws,Wd	ZE	[W12++],[W11]--	Wd = Zero Extend Ws

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Move Operations					
EXCH	Wns,Wnd	Swap Wns and Wnd	1	1	
MOV.D POP.D	Ws,Wnd Wnd	Move source or stack to W(nd+1):W(nd)	1	1	
MOV.Q POP.Q	Ws,Wnd Wnd	Move source or stack to W(nd+3):W(nd+2):W(nd+1):W(nd)	1	1	
MOV	f,Wn	Move f to Wn	1	1	
MOV PUSH POP	Wso,Wdo Wso Wdo	Move Ws to Wd Push Ws Pop Wd	1	1	1
MOV	f f,Ww	Move f to destination	1	1	1,2
MOV	lit16,Wn	Move 16-bit literal to Wn	1	1	
MOV	Slit10,Wn	Move 10-bit signed literal to Wn	1	1	1
MOV	Ww,f	Move Ww to f	1	1	1
MOV.D PUSH.D	Wns,Wd Wns	Move W(ns):W(ns+1) to destination or stack	1	1	
MOV.Q PUSH.Q	Wns,Wd Wns	Move W(ns):W(ns+1):W(ns+2):W(ns+3) to destination or stack	1	1	
MOV	Wn,f	Move Wn to f	1	1	
Table Operations					
TBLRDH	Ws,Wd	Read Prog<23:16> to Wd	1	2	1
TBLRDL	Ws,Wd	Read Prog<15:0> to Wd	1	2	1
TBLWTH	Ws,Wd	Write Ws to Prog<23:16>	1	2	1
TBLWTL	Ws,Wd	Write Ws to Prog<15:0>	1	2	1

Note 1: INST or INST.W is a word operation, B=0; INST.B is a byte operation, B=1
2: MOVF,MOVFW,TESTF are equivalent assembly mnemonics.

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Math Operations - W Registers					
ADD	Wb,Ws,Wd	Wd = Wb + Ws	1	1	1
ADDC	Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1	1
AND	Wb,Ws,Wd	Wd = Wb .AND. Ws	1	1	1
IOR	Wb,Ws,Wd	Wd = Wb .IOR. Ws	1	1	1
SUB	Wb,Ws,Wd	Wd = Wb - Ws	1	1	1
SUBB	Wb,Ws,Wd	Wd = Wb - Ws - ($\overline{C}$)	1	1	1
SUBR	Wb,Ws,Wd	Wd = Ws - Wb	1	1	1
SUBBR	Wb,Ws,Wd	Wd = Ws - Wb - ($\overline{C}$)	1	1	1
XOR	Wb,Ws,Wd	Wd = Wb .XOR. Ws	1	1	1
Math Operations - Short Unsigned Literals (literal 0..31**)**					
ADD	Wb,lit5,Wd	Wd = Wb + lit5	1	1	1
ADDC	Wb,lit5,Wd	Wd = Wb + lit5 + (C)	1	1	1
AND	Wb,lit5,Wd	Wd = Wb .AND. lit5	1	1	1
IOR	Wb,lit5,Wd	Wd = Wb .IOR. lit5	1	1	1
SUB	Wb,lit5,Wd	Wd = Wb - lit5	1	1	1
SUBB	Wb,lit5,Wd	Wd = Wb - lit5 - ($\overline{C}$)	1	1	1
SUBR	Wb,lit5,Wd	Wd = lit5 - Wb	1	1	1
SUBBR	Wb,lit5,Wd	Wd = lit5 - Wb - ($\overline{C}$)	1	1	1
XOR	Wb,lit5,Wd	Wd = Wb .XOR. lit5	1	1	1
Math Operations - W Registers Single Operand					
CLR	Ws	Ws = 0x0000	1	1	1
COM	Ws,Wd	Wd = $\overline{Ws}$	1	1	1
DEC	Ws,Wd	Wd = Ws - 1	1	1	1
DEC2	Ws,Wd	Wd = Ws - 2	1	1	1
INC	Ws,Wd	Wd = Ws + 1	1	1	1
INC2	Ws,Wd	Wd = Ws + 2	1	1	1
NEG	Ws,Wd	Wd = $\overline{Ws}$ + 1	1	1	1
SETM	Ws	Ws = 0xFFFF	1	1	1

Note 1: INST or INST.W is a word operation,B=0; INST.B is a byte operation,B=1.

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Math Operations - File Registers					
ADD	f f,Ww	f = f + Ww Ww = f + Ww	1	1	1
ADDC	f f,Ww	f = f + Ww + (C) Ww = f + Ww + (C)	1	1	1
AND	f f,Ww	f = f .AND. Ww Ww = f .AND. Ww	1	1	1
IOR	f f,Ww	f = f .IOR. Ww Ww = f .IOR. Ww	1	1	1
SUBR	f f,Ww	f = Ww - f Ww = Ww - f	1	1	1
SUBRB	f f,Ww	f = Ww - f - $(\overline{C})$ Ww = Ww - f - $(\overline{C})$	1	1	1
SUB	f f,Ww	f = f - Ww Ww = f - Ww	1	1	1
SUBB	f f,Ww	f = f - Ww - $(\overline{C})$ Ww = f - Ww - $(\overline{C})$	1	1	1
XOR	f f,Ww	f = f .XOR. Ww Ww = f .XOR. Ww	1	1	1
Math Operations - File Registers Single Operand					
CLR	f Ww	f = 0x0000 Ww = 0x0000	1	1	1
COM	f f,Ww	f = $\overline{f}$ Ww = $\overline{f}$	1	1	1
DEC	f f,Ww	f = f -1 Ww = f -1	1	1	1
INC	f f,Ww	f = f + 1 Ww = f + 1	1	1	1
NEG	f f,Ww	f = $\overline{f}$ + 1 Ww = $\overline{f}$ + 1	1	1	1
SETM	f Ww	f = 0xFFFF Ww = 0xFFFF	1	1	1
Note 1: INST or INST.W is a word operation,B=0; INST.B is a byte operation,B=1.					
Math Operations - Literals (literal -512..511**)**					
ADD	Slit10,Wn	Wn = Slit10 + Wn	1	1	1
ADDC	Slit10,Wn	Wn = Slit10 + Wn + (C)	1	1	1
AND	Slit10,Wn	Wn = Slit10 .AND. Wn	1	1	1
IOR	Slit10,Wn	Wn = Slit10 .IOR. Wn	1	1	1
SUB	Slit10,Wn	Wn = Slit10 - Wn	1	1	1
SUBB	Slit10,Wn	Wn = Slit10 - Wn - $(\overline{C})$	1	1	1
XOR	Slit10,Wn	Wn = Slit10 .XOR. Wn	1	1	1
Math Operations - Multiply,Adjust					
DAW.B	Wn	Wn = decimal adjust Wn	1	1	
DIV		Divide Helper	1	1	2
MUL.SS	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * sign(Ws)	1	1	
MUL.SU	Wb,Ws,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(Ws)	1	1	
MUL.SU	Wb,lit5,Wnd	{Wnd+1,Wnd} = sign(Wb) * unsign(lit5)	1	1	
MUL.UU	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(Ws)	1	1	
MUL.UU	Wb,lit5,Wnd	{Wnd+1,Wnd} = unsign(Wb) * unsign(lit5)	1	1	
MUL.US	Wb,Ws,Wnd	{Wnd+1,Wnd} = unsign(Wb) * sign(Ws)	1	1	
MUL	f	W3:W2 = f * Ww	1	1	1
SE	Ws,Wd	Wd = sign extended Ws	1	1	
ZE	Ws,Wd	Wd = zero extended Ws	1	1	
SWAP	Wn	Wn = byte or nibble swap Wn	1	1	1
Note 1: INST or INST.W is a word operation,B=0; INST.B is a byte operation,B=1. **Note 2:** Iterative Divide - 32/16, 16/16 - total cycle count TBD.					

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Rotate/Shift Operations - W Registers					
ASR	Ws,Wd	Wd = Arithmetic Right Shift Ws	1	1	1
LSR	Ws,Wd	Wd = Logical Right Shift Ws	1	1	1
RLC	Ws,Wd	Wd = Rotate Left through Carry Ws	1	1	1
RLNC	Ws,Wd	Wd = Rotate Left (No Carry) Ws	1	1	1
RRC	Ws,Wd	Wd = Rotate Right through Carry Ws	1	1	1
RRNC	Ws,Wd	Wd = Rotate Right (No Carry) Ws	1	1	1
SL	Ws,Wd	Wd = Arithmetic Left Shift Ws	1	1	1
Rotate/Shift Operations - File Registers					
ASR	f f,Ww	f = Arithmetic Right Shift f Ww = Arithmetic Right Shift f	1	1	1
LSR	f f,Ww	f = Logical Right Shift f Ww = Logical Right Shift f	1	1	1
RLC	f f,Ww	f = Rotate Left through Carry f Ww = Rotate Left through Carry f	1	1	1
RLNC	f f,Ww	f = Rotate Left (No Carry) f Ww = Rotate Left (No Carry) f	1	1	1
RRC	f f,Ww	f = Rotate Right through Carry f Ww = Rotate Right through Carry f	1	1	1
RRNC	f f,Ww	f = Rotate Right (No Carry) f Ww = Rotate Right (No Carry) f	1	1	1
SL	f f,Ww	f = Arithmetic Left Shift f Ww = Arithmetic Left Shift f	1	1	1

Note 1: INST or INST.W is a word operation, B=0; INST.B is a byte operation, B=1.

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Barrel Shift Operations - W Registers (shift range 0..31)					
ASR	Wb,Wns,Wnd	Wnd = Arithmetic Right Shift Wb by Wns	1	1	
LSR	Wb,Wns,Wnd	Wnd = Logical Right Shift Wb by Wns	1	1	
MSL	Wb,Wns,Wnd	Wnd = Multi-word Left Shift Wb by Wns	1	1	1
MSR	Wb,Wns,Wnd	Wnd = Multi-word Right Shift Wb by Wns	1	1	1
SL	Wb,Wns,Wnd	Wnd = Left Shift Wb by Wns	1	1	
Barrel Shift Operations - Short Literals (shift range 0..31)					
ASR	Wb,lit5,Wnd	Wnd = Arithmetic Right Shift Wb by lit5	1	1	
LSR	Wb,lit5,Wnd	Wnd = Logical Right Shift Wb by lit5	1	1	
MSL	Wb,lit5,Wnd	Wnd = Multi-word Left Shift Wb by lit5	1	1	
MSR	Wb,lit5,Wnd	Wnd = Multi-word Right Shift Wb by lit5	1	1	
SL	Wb,lit5,Wnd	Wnd = Left Shift Wb by lit5	1	1	
Note 1: Single word shift instruction to support multi-word operations. .					

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
DSP OPERATIONS - Accumulator Ops					
ADD	A	Add Accumulators	1	1	1
ADD	A,Wso,Slit4	16-bit Signed Add to Accumulator	1	1	1
LAC	A,Wso,Slit4	Load Accumulator	1	1	1
NEG	A	Negate Accumulators	1	1	1
SAC	A,Wdo,Slit4	Store Accumulator	1	1	1
SFTAC	A,Wn	Arithmetic Shift by (Wn) Accumulator	1	1	
SFTACK	A,Slit5	Arithmetic Shift by Slit5 Accumulator	1	1	
SAC.R	A,Wdo,Slit4	Store Rounded Accumulator	1	1	1
SUB	A	Subtract Accumulators	1	1	1
DSP OPERATIONS - MAC Ops					
CLR	A,Wxp,Wx,Wyp,Wy,AWB	Clear Accumulator	1	1	
ED	A,Wm*Wm,Wxp,Wx,Wy	Euclidean Distance	1	1	
EDAC	A,Wm*Wm,Wxp,Wx,Wy,AWB	Euclidean Distance Accumulate	1	1	
MAC	A,Wm*Wn,Wxp,Wx,Wyp,Wy,AWB	Multiply and Accumulate	1	1	
MOVSAC	A,Wxp,Wx,Wyp,Wy,AWB	Move Special	1	1	
MPY	A,Wm*Wn,Wxp,Wx,Wyp,Wy	Multiply Wn by Wm to Accumulator	1	1	
MPYN	A,Wm*Wn,Wxp,Wx,Wyp,Wy	-(Multiply Wn by Wm) to Accumulator	1	1	
MSC	A,Wm*Wn,Wxp,Wx,Wyp,Wy,AWB	Multiply and Subtract from Accumulator	1	1	
MPY	A,Wm*Wm,Wxp,Wx,Wyp,Wy	Square to Accumulator	1	1	
MAC	A,Wm*Wm,Wxp,Wx,Wyp,Wy,AWB	Square and Accumulate	1	1	

Note 1: lit4 translates to the rrrr field that specifies a shift count.

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
BIT OPERATIONS - W Registers					
BCLR	Ws,bit4	Bit Clear Ws	1	1	2
BSET	Ws,bit4	Bit Set Ws	1	1	2
BSW.C BSW.Z	Ws,Wb	Write C or Z bit to Ws<Wb>	1	1	2
BTG	Ws,bit4	Bit Toggle Ws	1	1	2
BTST.C BTST.Z	Ws,bit4	Bit Test Ws to C or Z	1	1	2
BTSTS.C BTSTS.Z	Ws,bit4	Bit Test Ws to C or Z then Set	1	1	2
BTST.C BTST.Z	Ws,Wb	Bit Test Ws<Wb> to C or Z	1	1	2
BIT OPERATIONS - File Registers					
BCLR.b	f,bit3	Bit Clear f	1	1	3
BSET.b	f,bit3	Bit Set f	1	1	3
BTG.b	f,bit3	Bit Toggle f	1	1	3
BTST.b	f,bit3	Bit Test f	1	1	3
BTSTS.b	f,bit3	Bit Test then Set f	1	1	3
BIT FIND OPERATIONS					
FBCL	Ws,Wd	Find Bit Change from Left (MSb) Side	1	1	1
FBCR	Ws,Wd	Find Bit Change from Right (LSb) Side	1	1	1
FF0L	Ws,Wd	Find First Zero from Left (MSb) Side	1	1	1
FF0R	Ws,Wd	Find First Zero from Right (LSb) Side	1	1	1
FF1L	Ws,Wd	Find First One from Left (MSb) Side	1	1	1
FF1R	Ws,Wd	Find First One from Right (LSb) Side	1	1	1

Note 1: INST or INST.W is a word operation,B=0; INST.B is a byte operation, B=1.

Note 2: bbbb field selects bit position 1111=MSb(15) 0000=LSb(0)

Note 3: bbb field selects bit position 111=MSb(7) 000=LSb(0)

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Skip OPERATIONS - W Registers					
BTSC	Ws,bit4	Bit Test Ws, Skip if Clear	1	1(2/3)	2
BTSS	Ws,bit4	Bit Test Ws, Skip if Set	1	1(2/3)	2
Skip OPERATIONS - File Registers					
BTSC.b	f,bit3	Bit Test f, Skip if Clear	1	1(2/3)	3
BTSS.b	f,bit3	Bit Test f, Skip if Set	1	1(2/3)	3
Inc/Dec Skip OPERATIONS - File Registers					
DECSNZ	f f,Ww	f = f-1, Skip if Not 0 Ww = f-1, Skip if Not 0	1	1(2/3)	1
DECSZ	f f,Ww	f = f-1, Skip if 0 Ww = f-1, Skip if 0	1	1(2/3)	1
INCSNZ	f f,Ww	f = f+1, Skip if Not 0 Ww = f+1, Skip if Not 0	1	1(2/3)	1
INCSZ	f f,Ww	f = f+1, Skip if 0 Ww = f+1, Skip if 0	1	1(2/3)	1

Note 1: INST or INST.W is a word operation,B=0; INST.B is a byte operation, B=1.

Note 2: bbbb field selects bit position 1111=MSb(15) 0000=LSb(0)

Note 3: bbb field selects bit position 111=MSb(7) 000=LSb(0)

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Compare OPERATIONS - W Registers					
CP0	Ws	Compare (Ws - 0x0000)	1	1	1
CP1	Ws	Compare (Ws - 0xFFFF)	1	1	1
CP	Wb,Ws	Compare (Ws - Wb)	1	1	1
CPB	Wb,Ws	Compare Borrow (Ws - Wb - $\overline{C}$)	1	1	1
Compare OPERATIONS - Short Literals (literal 0...31)					
CP	Wb,lit5	Compare (lit5 - Wb)	1	1	1
CPB	Wb,lit5	Compare Borrow (lit5 - Wb - $\overline{C}$)	1	1	1
Compare OPERATIONS - File Registers					
CP0	f	Compare (f - 0x0000)	1	1	1
CP1	f	Compare (f - 0xFFFF)	1	1	1
CP	f	Compare (f - Ww)	1	1	1
CPB	f	Compare Borrow (f - Ww - $\overline{C}$)	1	1	1
Compare Skip OPERATIONS - File Registers					
CPFSEQ	f	Compare (f - Ww), skip if =	1	1(2/3)	1
CPFSGT	f	Compare (f - Ww), skip if >	1	1(2/3)	1
CPFSLT	f	Compare (f - Ww), skip if <	1	1(2/3)	1
CPFSNE	f	Compare (f - Ww), skip if ≠	1	1(2/3)	1

Note 1: INST or INST.W is a word operation,B=0; INST.B is a byte operation, B=1.

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Branch Operations					
BRA BRA	C,Slit16 GEU,Slit16	Branch if Carry	1	2	1
BRA	GE,Slit16	Branch if greater than or equal	1	2	1
BRA	GT,Slit16	Branch if greater than	1	2	1
BRA	GTU,Slit16	Branch if unsigned greater than	1	2	1
BRA	LE,Slit16	Branch if less than or equal	1	2	1
BRA	LEU,Slit16	Branch if unsigned less than or equal	1	2	1
BRA	LT,Slit16	Branch if less than	1	2	1
BRA	N,Slit16	Branch if Negative	1	2	1
BRA BRA	NC,Slit16 LTU,Slit16	Branch if Not Carry	1	2	1
BRA	NN,Slit16	Branch if Not Negative	1	2	1
BRA	NOV,Slit16	Branch if Not Overflow	1	2	1
BRA	NZ,Slit16	Branch if Not Zero	1	2	1
BRA	OA,Slit16	Branch if accumulator A overflow	1	2	1
BRA	OB,Slit16	Branch if accumulator B overflow	1	2	1
BRA	OV,Slit16	Branch if Overflow	1	2	1
BRA	Slit16	Branch Unconditionally	1	2	1
BRA	SA,Slit16	Branch if accumulator A saturated	1	2	1
BRA	SB,Slit16	Branch if accumulator B saturated	1	2	1
BRA	Z,Slit16	Branch if Zero	1	2	1

Note 1: 16-bit signed literal allows jump range of PC-32768 to PC+32767.

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Jump / Call / Return Operations					
BRA	Wn	Computed branch	1	2	
CALL CALL.S	lit23	Call subroutine	2	2	3
CALL CALL.S	Wn	Call indirect subroutine	1	2	
GOTO	lit23	Go to address	2	2	3
GOTO	Wn	Go to indirect	1	2	
RCALL	Slit16	Relative Call	1	2	2
RCALL	Wn	Computed Call	1	2	
RETFIE RETFIE.S		Return from interrupt enable	1	2	
RETLW RETLW.S	Slit10,Wn	Return with Slit10 in Wn	1	2	1
RETURN RETURN.S		Return from Subroutine	1	2	
TRAP	lit1,lit16	Trap to vector(lit1) with lit16	1	2	
Looping Operations					
DO	Slit16,lit14	Do code to PC+Slit16, lit14 times	2	2+ loop	2
DO	Slit16,Wn	Do code to PC+Slit16, (Wn) times	2	2+ loop	2
REPEAT	lit14	Repeat Next Instruction lit14 times	1	1 + lit14	
REPEAT	Wn	Repeat Next Instruction (Wn) times	1	1 + (Wn)	

Note 1: INST or INST.W is a word operation,B=0; INST.B is a byte operation, B=1.

Note 2: 16-bit signed literal allows jump range of PC-32768 to PC+32767.

Note 3: 23-bit literal coded in 2 words,1st word contains n<15:0>, n<0>=0, 2nd word contains n<22:16>.

TABLE 1-4: dsPIC30F INSTRUCTION SET - FUNCTIONAL GROUPING

Assembly Syntax Mnemonic, Operands		Description	W	CY	Note
Stack Operations					
POP.S		Pop Shadow Registers	1	1	
LNK	lit14	Link frame pointer	1	1	
POP	f	Pop f from top of stack (TOS)	1	1	
PUSH	f	Push f to top of stack (TOS)	1	1	
PUSH.S		Push Shadow Registers	1	1	
ULNK		Unlink frame pointer	1	1	
Control Operations					
CLRWDT		Clear Watchdog Timer	1	1	
DISI	lit14	Disable Interrupts for lit14 instruction cycles	1	1	
HALT		No Operation/ HALT	1	1	
NOP		No Operation	1	1	
NOPR		No Operation	1	1	
RESET		Software device RESET	1	1	
SLEEP	lit4	Go into standby mode	1	1	

TABLE 1-5: ADDRESSING MODES FOR Ws SOURCE REGISTER (ADDRESS MODE 1)

ppp	Addressing Mode	Source Operand	Instruction Operation(3)	Effective Address
000	Register Direct	Ws	Wd = Ws op Wb	EAs = W register number
001	Indirect	[Ws]	Wd = [Ws] op Wb	EAs = Ws
010	Indirect with post-decrement	[Ws]--	Wd = [Ws]-- op Wb	EAs = Ws; Ws <- (Ws - 1)(1) - or - Ws <- (Ws - 2)(2)
011	Indirect with post-increment	[Ws]++	Wd = [Ws]++ op Wb	EAs = Ws; Ws <- (Ws + 1)(1) - or - Ws <- (Ws + 2)(2)
100	Indirect with pre-decrement	[Ws--]	Wd = [Ws--] op Wb	Ws <- (Ws - 1)(1); - or - Ws <- (Ws - 2)(2); EAs = Ws
101	Indirect with pre-increment	[Ws++]	Wd = [Ws++] op Wb	Ws <- (Ws + 1)(1); - or - Ws <- (Ws + 2)(2); EAs = Ws
11k	(Specifies Slit5 Source for Short Literal Instructions)			

Note 1: For byte operations, add or subtract 1.
2: For word operations, add or subtract 2.
3: **Wd assumed (but not required) to be in register direct mode**.

TABLE 1-6: ADDRESSING MODES FOR Wd DESTINATION REGISTER (ADDRESS MODE 2)

qqq	Addressing Mode	Destination Operand	Instruction Operation(3)	Effective Address
000	Register Direct	Wd	Wd = Ws op Wb	EAd = W register number
001	Indirect	[Wd]	[Wd] = Ws op Wb	EAd = Wd
010	Indirect with post-decrement	[Wd]--	[Wd]-- = Ws op Wb	EAd = Wd; Wd <- (Wd - 1)(1) - or - Wd <- (Wd - 2)(2)
011	Indirect with post-increment	[Wd]++	[Wd]++ = Ws op Wb	EAd = Wd; Wd <- (Wd + 1)(1) - or - Wd <- (Wd + 2)(2)
100	Indirect with pre-decrement	[Wd--]	[Wd--] = Ws op Wb	Wd <- (Wd - 1)(1); - or - Wd <- (Wd - 2)(2); EAd = Wd
101	Indirect with pre-increment	[Wd++]	[Wd++] = Ws op Wb	Wd <- (Wd + 1)(1); - or - Wd <- (Wd + 2)(2); EAd = Wd
11x	(Unused)			

Note 1: For byte operations, add or subtract 1.
2: For word operations, add or subtract 2.
3: **Ws assumed (but not required) to be in register direct mode**.

TABLE 1-7: OFFSET ADDRESSING MODES FOR WSO SOURCE REGISTER (MODE 3)

ggg	Addressing Mode	Source Operand	Effective Address
000	Register Direct	Wns	EA = W register number
001	Indirect	[Wns]	EA = Wns
010	Indirect with post-decrement	[Wns]--	EA = Wns; Wns <- (Wns - 1)**(1)** - or - Wns <- (Wns - 2)**(2)**
011	Indirect with post-increment	[Wns]++	EA = Wns; Wns <- (Wns + 1)**(1)** - or - Wns <- (Wns + 2)**(2)**
100	Indirect with pre-decrement	[Wns--]	Wns <- (Wns - 1)**(1)**; - or - Wns <- (Wns - 2)**(2)**; EA = Wns
101	Indirect with register offset	[Wns+Wb]	EA = Wns + Wb**(3)**
11g	Indirect with signed offset by short literal Slit5 ∈ (-16...15)	[Wns+Slit5]	EA = (Wns + gwwww)**(4)** - or - EA = (Wns + 2*gwwww)**(5)**

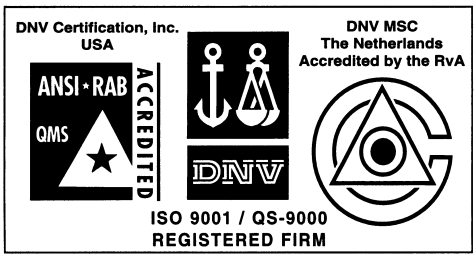
Note 1: For byte operations, add or subtract 1.
2: For word operations, add or subtract 2.
3: For byte and word operations, add 2's compliment Wb.
4: For byte operations, add or subtract gwwww.
5: For word operations, add or subtract (2 * gwwww) or gwwww0.

TABLE 1-8: OFFSET ADDRESSING MODES FOR WDO DESTINATION REGISTER (MODE 3)

hhh	Addressing Mode	Source Operand	Effective Address
000	Register Direct	Wnd	EA = W register number
001	Indirect	[Wnd]	EA = Wnd
010	Indirect with post-decrement	[Wnd]--	EA = Wnd; Wnd <- (Wnd - 1)**(1)** - or - Wnd <- (Wnd - 2)**(2)**
011	Indirect with post-increment	[Wnd]++	EA = Wnd; Wnd <- (Wnd + 1)**(1)** - or - Wnd <- (Wnd + 2)**(2)**
100	Indirect with pre-decrement	[Wnd--]	Wnd <- (Wnd - 1)**(1)**; - or - Wnd <- (Wnd - 2)**(2)**; EA = Wnd
101	Indirect with register offset	[Wnd+Wb]	EA = Wnd + Wb**(3)**
11h	Indirect with signed offset by short literal Slit5 ∈ (-16...15)	[Wnd+Slit5]	EA = (Wnd + hwwww)**(4)** - or - EA = (Wnd + 2*hwwww)**(5)**

Note 1: For byte operations, add or subtract 1.
2: For word operations, add or subtract 2.
3: For byte and word operations, add 2's compliment Wb.
4: For byte operations, add or subtract hwwww.
5: For word operations, add or subtract (2 * hwwww) or hwwww0.

Trademarks



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

Atlanta
500 Sugar Mill Road, Suite 200B
Atlanta, GA 30350
Tel: 770-640-0034 Fax: 770-640-0307

Austin
Analog Product Sales
8303 MoPac Expressway North
Suite A-201
Austin, TX 78759
Tel: 512-345-2030 Fax: 512-345-6085

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Boston
Analog Product Sales
Unit A-8-1 Millbrook Tarry Condominium
97 Lowell Road
Concord, MA 01742
Tel: 978-371-6400 Fax: 978-371-0050

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

Dayton
Two Prestige Place, Suite 130
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250 Fax: 248-538-2260

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

Mountain View
Analog Product Sales
1300 Terra Bella Avenue
Mountain View, CA 94043-1836
Tel: 650-968-9241 Fax: 650-967-1590

New York
150 Motor Parkway, Suite 202
Hauppauge, NY 11788
Tel: 631-273-5305 Fax: 631-273-5335

San Jose
Microchip Technology Inc.
2107 North First Street, Suite 590
San Jose, CA 95131
Tel: 408-436-7950 Fax: 408-436-7955

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Beijing Office
Unit 915
New China Hong Kong Manhattan Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Shanghai
Microchip Technology Shanghai Office
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

Hong Kong
Microchip Asia Pacific
RM 2101, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

Japan
Microchip Technology Intl. Inc.
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

ASIA/PACIFIC (continued)

Korea
Microchip Technology Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea
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: 65-334-8870 Fax: 65-334-8850

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

EUROPE

Denmark
Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45 4420 9895 Fax: 45 4420 9910

France
Arizona 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
Arizona Microchip Technology GmbH
Gustav-Heinemann Ring 125
D-81739 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Germany
Analog Product Sales
Lochhamer Strasse 13
D-82152 Martinsried, Germany
Tel: 49-89-895650-0 Fax: 49-89-895650-22

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleoni
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-039-65791-1 Fax: 39-039-6899883

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

01/30/01