MPASM™ and MPLINK™ PICmicro® MCU Quick Reference Guide

MPLAB CERTIFIED™

MPASM Quick Reference Guide

This Quick Reference Guide gives all the instructions, directives and command line options for the Microchip MPASM Assembler.

MPASM Directive Language Summary

Directive	Description	Syntax
CONTROL DIRECTIVES		
CONSTANT	Declare Symbol Constant	constant <label> [= <expr>, ...,<label> [= <expr>]]
#DEFINE	Define Text Substitution	#define <name> [[(<arg>,...,<arg>)]<value>]
END	End Program Block	end
EQU	Define Assembly Constant	<label> equ <expr>
ERROR	Issue an Error Message	error "<text_string>"
ERROR-LEVEL	Set Messge Level	errorlevel 0|1|2|<+-><msg>
#INCLUDE	Include Source File	include <<include_file>> include "<include_file>"
LIST	Listing Options	list [<option>[,...,<option>]]
MESSG	User Defined Message	messg "<message_text>"
NOLIST	Turn off Listing Output	nolist
ORG	Set Program Origin	<label> org <expr>
PAGE	Insert Listing Page Eject	page
PROCESSOR	Set Processor Type	processor <processsor_type>
RADIX	Specify Default Radix	radix <default_radix>
SET	Assign Value to Variable	<label> set <expr>
SPACE	Insert Blank Listing Lines	space [<expr>]
SUBTITLE	Specify Program Subtitle	subtitl "<sub_text>"
TITLE	Specify Program Title	title "<title_text>"
#UNDEFINE	Delete a Substitution Label	#undefine <label>
VARIABLE	Declare Symbol Variable	variable <label> [= <expr>,...,]
CONDITIONAL ASSEMBLY		
ELSE	Begin Alternative Assembly to IF	else
ENDIF	End Conditional Assembly	endif
ENDW	End a While Loop	endw
IF	Begin Conditional ASM Code	if <expr>
IFDEF	Execute If Symbol Defined	ifdef <label>
IFNDEF	Execute If Symbol Not Defined	ifndef <label>
WHILE	Perform Loop While True	while <expr>

MPASM Directive Language Summary (Continued)

Directive	Description	Syntax
DATA		
_ _BADRAM	Specify invalid RAM locations	_ _badram <expr>
_ _BADROM	Specify invalid ROM locations	_ _badrom <expr>
CBLOCK	Define Block of Constants	cblock [<expr>]
_ _CONFIG	Set configuration bits	_ _config <expr> OR _ _config <addr>, <expr>
DA	Pack Strings in 14-bit Memory	[<label>] da <expr> [, <expr2>, ..., <exprn>]
DATA	Create Numeric/Text Data	data <expr>,[,<expr>,...,<expr>] data "<text_string>"[,"<text_string>",...]
DB	Declare Data of One Byte	db <expr>[,<expr>,...,<expr>]
DE	Declare EEPROM Data	de <expr>[,<expr>,...,<expr>]
DT	Define Table	dt <expr>[,<expr>,...,<expr>]
DW	Declare Data of One Word	dw <expr> [,<expr>,...,<expr>]
ENDC	End CBlock	endc
FILL	Specify Memory Fill Value	fill <expr>, <count>
_ _ IDLOCS	Set ID locations	_ _idlocs <expr>
_ _MAXRAM	Specify max RAM adr	_ _maxram <expr>
RES	Reserve Memory	res <mem_units>
MACROS		
ENDM	End a Macro Definition	endm
EXITM	Exit from a Macro	exitm
EXPAND	Expand Macro Listing	expand
LOCAL	Declare Local Macro Variable	local <label> [,<label>]
MACRO	Declare Macro Definition	<label> macro [<arg>,...,<arg>]
NOEXPAND	Turn off Macro Expansion	noexpand
OBJECT FILE DIRECTIVES		
BANKISEL	Select Bank for indirect	bankisel <label>
BANKSEL	Select RAM bank	banksel <label>
CODE	Executable code section	[<name>] code [<address>]
CODE_PACK	Packed data in program memory	[<name>] code_pack [<address>]
EXTERN	Declare external label	extern <label> [,<label>]
GLOBAL	Export defined label	extern <label> [.<label>]
IDATA	Initialized data section	[<name>] idata [<address>]
PAGESEL	Select ROM page	pagesel <label>
UDATA	Uninitialized data section	[<name>] udata [<address>]
UDATA_ACS	Access uninit data sect	[<name>] udata_acs [<address>]
UDATA_OVR	Overlay uninit data sect	[<name>] udata_ovr [<address>]
UDATA_SHR	Shared uninit data sect	[<name>] udata_shr [<address>]

MPASM Radix Types Supported

Radix	Syntax	Example
Decimal	D‘<digits>’ .<digits>	D‘100’ .100
Hexadecimal (default)	H‘<hex_digits>’ 0x<hex_digits>	H‘9f’ 0x9f
Octal	O‘<octal_digits>’	O‘777’
Binary	B‘<binary_digits>’	B‘00111001’
Character (ASCII)	‘<character>’ A‘<Character>’	A‘C’ ‘C’

MPLINK Command Line Options

Option	Description
/o filename	Specify output file ‘filename’. Default is a.out.
/m filename	Create map file ‘filename’.
/l pathlist	Add directories to library search path.
/k pathlist	Add directories to linker script search path.
/n length	Specify number of lines per listing page.
/h, /?	Display help screen.
/a hexformat	Specify format of hex output file.
/q	Quiet mode.
/d	Don’t create an absolute listing file.

Key to 12, 14 and 16-bit PICmicro Family Instruction Sets

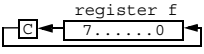
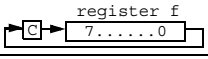
Field	Description
b	Bit address within an 8 bit file register
d	Destination select; d = 0 Store result in W (f0A). d = 1 Store result in file register f. Default is d = 1.
f	Register file address (0x00 to 0xFF)
k	Literal field, constant data or label
W	Working register (accumulator)
x	Don’t care location
i	Table pointer control; i = 0 Do not change. i = 1 Increment after instruction execution.
p	Peripheral register file address (0x00 to 0x1f)
t	Table byte select; t = 0 Perform operation on lower byte. t = 1 Perform operation on upper byte.
PH:PL	Multiplication results registers

12-Bit Core Instruction Set

12-Bit Core Literal and Control Operations

Hex	Mnemonic		Description	Function
Ekk	ANDLW	k	AND literal and W	k .AND. W → W
9kk	CALL	k	Call subroutine	PC + 1 → TOS, k → PC
004	CLRWDT		Clear watchdog timer	0 → WDT (and Prescaler if assigned)
Akk	GOTO	k	Goto address (k is 9 bits)	k → PC(9 bits)
Dkk	IORLW	k	Incl. OR literal and W	k .OR. W → W
Ckk	MOVLW	k	Move Literal to W	k → W
002	OPTION		Load OPTION Register	W → OPTION Register
8kk	RETLW	k	Return with literal in W	k → W, TOS → PC
003	SLEEP		Go into Standby Mode	0 → WDT, stop osc
00f	TRIS	f	Tristate port f	W → I/O control reg f
Fkk	XORLW	k	Exclusive OR literal and W	k .XOR. W → W

12-Bit Core Byte Oriented File Register Operations

Hex	Mnemonic		Description	Function
1Cf	ADDWF	f,d	Add W and f	W + f → d
14f	ANDWF	f,d	AND W and f	W .AND. f → d
06f	CLRF	f	Clear f	0 → f
040	CLRW		Clear W	0 → W
24f	COMF	f,d	Complement f	.NOT. f → d
0Cf	DECF	f,d	Decrement f	f - 1 → d
2Cf	DECFSZ	f,d	Decrement f, skip if zero	f - 1 → d, skip if zero
28f	INCF	f,d	Increment f	f + 1 → d
3Cf	INCFSZ	f,d	Increment f, skip if zero	f + 1 → d, skip if zero
10f	IORWF	f,d	Inclusive OR W and f	W .OR. f → d
20f	MOVF	f,d	Move f	f → d
02f	MOVWF	f	Move W to f	W → f
000	NOP		No operation	
34f	RLF	f,d	Rotate left f	
30f	RRF	f,d	Rotate right f	
08f	SUBWF	f,d	Subtract W from f	f - W → d
38f	SWAPF	f,d	Swap halves f	f(0:3) → f(4:7) → d
18f	XORWF	f,d	Exclusive OR W and f	W .XOR. f → d

12-Bit Core Bit Oriented File Register Operations

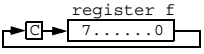
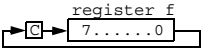
Hex	Mnemonic		Description	Function
4bf	BCF	f,b	Bit clear f	0 → f(b)
5bf	BSF	f,b	Bit set f	1 → f(b)
6bf	BTFSC	f,b	Bit test, skip if clear	skip if f(b) = 0
7bf	BTFSS	f,b	Bit test, skip if set	skip if f(b) = 1

14-Bit Core Instruction Set

14-Bit Core Literal and Control Operations

Hex	Mnemonic		Description	Function
3Ekk	ADDLW	k	Add literal to W	k + W → W
39kk	ANDLW	k	AND literal and W	k .AND. W → W
2kkk	CALL	k	Call subroutine	PC + 1 → TOS, k → PC
0064	CLRWDT	T	Clear watchdog timer	0 → WDT (and Prescaler)
2kkk	GOTO	k	Goto address (k is nine bits)	k → PC(9 bits)
38kk	IORLW	k	Incl. OR literal and W	k .OR. W → W
30kk	MOVLW	k	Move Literal to W	k → W
0062	OPTION		Load OPTION register	W → OPTION Register
0009	RETFIE		Return from Interrupt	TOS → PC, 1 → GIE
34kk	RETLW	k	Return with literal in W	k → W, TOS → PC
0008	RETURN		Return from subroutine	TOS → PC
0063	SLEEP		Go into Standby Mode	0 → WDT, stop oscillator
3Ckk	SUBLW	k	Subtract W from literal	k - W → W
006f	TRIS	f	Tristate port f	W → I/O control reg f
3Akk	XORLW	k	Exclusive OR literal and W	k .XOR. W → W

14-Bit Core Byte Oriented File Register Operations

Hex	Mnemonic		Description	Function
07ff	ADDWF	f,d	Add W and f	W + f → d
05ff	ANDWF	f,d	AND W and f	W .AND. f → d
018f	CLRF	f	Clear f	0 → f
0100	CLRW		Clear W	0 → W
09ff	COMF	f,d	Complement f	.NOT. f → d
03ff	DECF	f,d	Decrement f	f - 1 → d
0Bff	DECFSZ	f,d	Decrement f, skip if zero	f - 1 → d, skip if 0
0Aff	INCF	f,d	Increment f	f + 1 → d
0Fff	INCFSZ	f,d	Increment f, skip if zero	f + 1 → d, skip if 0
04ff	IORWF	f,d	Inclusive OR W and f	W .OR. f → d
08ff	MOVF	f,d	Move f	f → d
008f	MOVWF	f	Move W to f	W → f
0000	NOP		No operation	
0Dff	RLF	f,d	Rotate left f	
0Cff	RRF	f,d	Rotate right f	
02ff	SUBWF	f,d	Subtract W from f	f - W → d
0Eff	SWAPF	f,d	Swap halves f	f(0:3) → f(4:7) → d
06ff	XORWF	f,d	Exclusive OR W and f	W .XOR. f → d
1bff	BCF	f,b	Bit clear f	0 → f(b)
1bff	BSF	f,b	Bit set f	1 → f(b)
1bff	BTFSC	f,b	Bit test, skip if clear	skip if f(b) = 0
1bff	BTFSS	f,b	Bit test, skip if set	skip if f(b) = 1

12-Bit/14-Bit Core Special Instruction Mnemonics

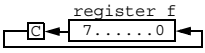
Mnemonic		Description	Equivalent Operation(s)		Status
ADDCF	f,d	Add Carry to File	BTFSC INCF	3,0 f,d	Z
ADDDCF	f,d	Add Digit Carry to File	BTFSC INCF	3,1 f,d	Z
B	k	Branch	GOTO	k	–
BC	k	Branch on Carry	BTFSC GOTO	3,0 k	–
BDC	k	Branch on Digit Carry	BTFSC GOTO	3,1 k	–
BNC	k	Branch on No Carry	BTFSS GOTO	3,0 k	–
BNDC	k	Branch on No Digit Carry	BTFSS GOTO	3,1 k	–
BNZ	k	Branch on No Zero	BTFSS GOTO	3,2 k	–
BZ	k	Branch on Zero	BTFSC GOTO	3,2 k	–
CLRC		Clear Carry	BCF	3,0	–
CLRDC		Clear Digit Carry	BCF	3,1	–
CLRZ		Clear Zero	BCF	3,2	–
LCALL	k	Long Call	BCF/BSF BCF/BSF CALL	0x0A,3 0x0A,4 k	–
LGOTO	k	Long GOTO	BCF/BSF BCF/BSF GOTO	0x0A,3 0x0A,4 k	–
MOVFW	f	Move File to W	MOVF	f,0	Z
NEGF	f,d	Negate File	COMF INCF	f,1 f,d	Z
SETC		Set Carry	BSF	3,0	–
SETDC		Set Digit Carry	BSF	3,1	–
SETZ		Set Zero	BSF	3,2	–
SKPC		Skip on Carry	BTFSS	3,0	–
SKPDC		Skip on Digit Carry	BTFSS	3,1	–
SKPNC		Skip on No Carry	BTFSC	3,0	–
SKPNDC		Skip on No Digit Carry	BTFSC	3,1	–
SKPNZ		Skip on Non Zero	BTFSC	3,2	–
SKPZ		Skip on Zero	BTFSS	3,2	–
SUBCF	f,d	Subtract Carry from File	BTFSC DECF	3,0 f,d	Z
SUBDCF	f,d	Subtract Digit Carry from File	BTFSC DECF	3,1 f,d	Z
TSTF	f	Test File	MOVF	f,1	Z

16-Bit Core Instruction Set

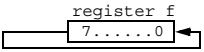
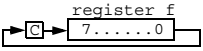
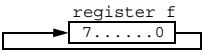
16-Bit Core Data Movement Instructions

Hex	Mnemonic		Description	Function
6pff	MOVFP	f,p	Move f to p	f → p
b8kk	MOVLB	k	Move literal to BSR	k → BSR (3:0)
bakx	MOVLP	k	Move literal to RAM page select	k → BSR (7:4)
4pff	MOVPF	p,f	Move p to f	p → W
01ff	MOVWF	f	Move W to F	W → f
a8ff	TABLRD	t,i,f	Read data from table latch into file f, then update table latch with 16-bit contents of memory location addressed by table pointer	TBLATH → f if t=1, TBLATL → f if t=0; ProgMem(TBLPTR) → TBLAT; TBLPTR + 1 → TBLPTR if i=1
acff	TABLWT	t,i,f	Write data from file f to table latch and then write 16-bit table latch to program memory location addressed by table pointer	f → TBLATH if t = 1, f → TBLATL if t = 0; TBLAT → ProgMem(TBLPTR); TBLPTR + 1 → TBLPTR if i=1
a0ff	TLRD	t,f	Read data from table latch into file f (table latch unchanged)	TBLATH → f if t = 1 TBLATL → f if t = 0
a4ff	TLWT	t,f	Write data from file f into table latch	f → TBLATH if t = 1 f → TBLATL if t = 0

16-Bit Core Arithmetic and Logical Instruction

Hex	Mnemonic		Description	Function
b1kk	ADDLW	k	Add literal to W	(W + k) → W
0eff	ADDWF	f,d	Add W to F	(W + f) → d
10ff	ADDWFC	f,d	Add W and Carry to f	(W + f + C) → d
b5kk	ANDLW	k	AND Literal and W	(W .AND. k) → W
0aff	ANDWF	f,d	AND W with f	(W .AND. f) → d
28ff	CLRF	f,d	Clear f and Clear d	0x00 → f,0x00 → d
12ff	COMF	f,d	Complement f	.NOT. f → d
2eff	DAW	f,d	Dec. adjust W, store in f,d	W adjusted → f and d
06ff	DECF	f,d	Decrement f	(f - 1) → f and d
14ff	INCF	f,d	Increment f	(f + 1) → f and d
b3kk	IORLW	k	Inclusive OR literal with W	(W .OR. k) → W
08ff	IORWF	f,d	Inclusive or W with f	(W .OR. f) → d
b0kk	MOVLW	k	Move literal to W	k → W
bckk	MULLW	k	Multiply literal and W	(k x W) → PH:PL
34ff	MULWF	f	Multiply W and f	(W x f) → PH:PL
2cff	NEGW	f,d	Negate W, store in f and d	(W + 1) → f,(W + 1) → d
1aff	RLCF	f,d	Rotate left through carry	

16-Bit Core Arithmetic and Logical Instruction (Continued)

Hex	Mnemonic		Description	Function
22ff	RLNCF	f,d	Rotate left (no carry)	
18ff	RRCF	f,d	Rotate right through carry	
20ff	RRNCF	f,d	Rotate right (no carry)	
2aff	SETF	f,d	Set f and Set d	0xff → f,0xff → d
b2kk	SUBLW	k	Subtract W from literal	(k - W) → W
04ff	SUBWF	f,d	Subtract W from f	(f - W) → d
02ff	SUBWFB	f,d	Subtract from f with borrow	(f - W - c) → d
1cff	SWAPF	f,d	Swap f	f(0:3) → d(4:7), f(4:7) → d(0:3)
b4kk	XORLW	k	Exclusive OR literal with W	(W .XOR. k) → W
0cff	XORWF	f,d	Exclusive OR W with f	(W .XOR. f) → d

16-Bit Core Bit Handling Instructions

Hex	Mnemonic		Description	Function
8bff	BCF	f,b	Bit clear f	0 → f(b)
8bff	BSF	f,b	Bit set f	1 → f(b)
9bff	BTFSC	f,b	Bit test, skip if clear	skip if f(b) = 0
9bff	BTFSS	f,b	Bit test, skip if set	skip if f(b) = 1
3bff	BTG	f,b	Bit toggle f	.NOT. f(b) → f(b)

16-Bit Core Program Control Instructions

Hex	Mnemonic		Description	Function
ekkk	CALL	k	Subroutine call (within 8k page)	PC+1 → TOS,k → PC(12:0), k(12:8) → PCLATH(4:0), PC(15:13) → PCLATH(7:5)
31ff	CPFSEQ	f	Compare f/w, skip if f = w	f-W, skip if f = W
32ff	CPFSGT	f	Compare f/w, skip if f > w	f-W, skip if f > W
30ff	CPFSLT	f	Compare f/w, skip if f < w	f-W, skip if f < W
16ff	DECFSZ	f,d	Decrement f, skip if 0	(f-1) → d, skip if 0
26ff	DCFSNZ	f,d	Decrement f, skip if not 0	(f-1) → d, skip if not 0
ckkk	GOTO	k	Unconditional branch (within 8k)	k → PC(12:0) k(12:8) → PCLATH(4:0), PC(15:13) → PCLATH(7:5)
1eff	INCFSZ	f,d	Increment f, skip if zero	(f+1) → d, skip if 0
24ff	INFSNZ	f,d	Increment f, skip if not zero	(f+1) → d, skip if not 0
b7kk	LCALL	k	Long Call (within 64k)	(PC+1) → TOS; k → PCL, (PCLATH) → PCH
0005	RETFIE		Return from interrupt, enable interrupt	(PCLATH) → PCH:k → PCL 0 → GLINTD
b6kk	RETLW	k	Return with literal in W	k → W, TOS → PC, (PCLATH unchanged)

16-Bit Core Program Control Instructions (Continued)

Hex	Mnemonic	Description	Function
0002	RETURN	Return from subroutine	TOS → PC (PCLATH unchanged)
33ff	TSTFSZ f	Test f, skip if zero	skip if f = 0

16-Bit Core Special Control Instructions

Hex	Mnemonic	Description	Function
0004	CLRWDT	Clear watchdog timer	0 → WDT, 0 → WDT prescaler, 1 → PD, 1 → TO
0003	SLEEP	Enter Sleep Mode	Stop oscillator, power down, 0 → WDT, 0 → WDT Prescaler 1 → PD, 1 → TO

Key to Enhanced 16-Bit Core Instruction Set

Field	Description
FILE ADDRESSES	
f	8-bit file register address
fs	12-bit source file register address
fd	12-bit destination file register address
dest	W register if d = 0; file register if d = 1
r	0, 1 or 2 for FSR number
LITERALS	
kk	8-bit literal value
kb	4-bit literal value
kc	bits 8-11 of 12-bit literal value
kd	bits 0-7 of 12-bit literal value
OFFSETS, ADDRESSES	
nn	8-bit relative offset (signed, 2's complement)
nd	11-bit relative offset (signed, 2's complement)
ml	bits 0-7 of 20-bit program memory address
mm	bits 8-19 of 20-bit program memory address
BITS	
b	bits 9-11; bit address
d	bit 9; 0=W destination; 1=f destination
a	bit 8; 0=access block; 1=BSR selects bank
s	bit 0 (bit 8 for CALL); 0=no update; 1(fast)=update/save W, STATUS, BSR

Enhanced 16-Bit Core Instruction Set

Enhanced 16-Bit Core Literal Operations

Hex	Mnemonic	Description	Function
0F*kk*	ADDLW kk	ADD literal to WREG	W+kk → W
0B*kk*	ANDLW kk	AND literal with WREG	W .AND. kk → W
0004	CLRWD	Clear watchdog timer	0 → WDT, 0 → WDT postscaler, 1 → TO,1 → PD
0007	DAW	Decimal Adjust WREG	if W<3:0> >9 or DC=1, W<3:0>+6→W<3:0>, else W<3:0> → W<3:0>; if W<7:4> >9 or C=1, W<7:4>+6→W<7:4>, else W<7:4> → W<7:4>;
09*kk*	IORLW kk	Inclusive OR literal with WREG	W .OR. kk → W
EE*kc* F0*kd*	LFSR r,kd:kc	Load 12-bit Literal to FSR (second word)	kd:kc → FSRr
01*kb*	MOVLB kb	Move literal to low nibble in BSR	kb → BSR
0E*kk*	MOVLW kk	Move literal to WREG	kk → W
0D*kk*	MULLW kk	Multiply literal with WREG	W * kk → PRODH:PRODL
08*kk*	SUBLW kk	Subtract W from literal	kk–W → W
0A*kk*	XORLW kk	Exclusive OR literal with WREG	W .XOR. kk → W

Enhanced 16-Bit Core Memory Operations

Hex	Mnemonic	Description	Function
0008	TBLRD*	Table Read (no change to TBLPTR)	Prog Mem (TBLPTR) → TABLAT
0009	TBLRD*+	Table Read (post-increment TBLPTR)	Prog Mem (TBLPTR) → TABLAT TBLPTR +1 → TBLPTR
000A	TBLRD*-	Table Read (post-decrement TBLPTR)	Prog Mem (TBLPTR) → TABLAT TBLPTR -1 → TBLPTR
000B	TBLRD+*	Table Read (pre-increment TBLPTR)	TBLPTR +1 → TBLPTR Prog Mem (TBLPTR) → TABLAT
000C	TBLWT*	Table Write (no change to TBLPTR)	TABLAT → Prog Mem(TBLPTR)
000D	TBLWT*+	Table Write (post-increment TBLPTR)	TABLAT → Prog Mem(TBLPTR) TBLPTR +1 → TBLPTR
000E	TBLWT*-	Table Write (post-decrement TBLPTR)	TABLAT → Prog Mem(TBLPTR) TBLPTR -1 → TBLPTR
000F	TBLWT+*	Table Write (pre-increment TBLPTR)	TBLPTR +1 → TBLPTR TABLAT → Prog Mem(TBLPTR)

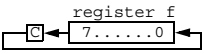
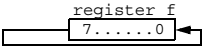
Enhanced 16-Bit Core Control Operations

Hex	Mnemonic		Description	Function
E2*nn*	BC	nn	Relative Branch if Carry	if C=1, PC+2+2*nn→PC, else PC+2→PC
E6*nn*	BN	nn	Relative Branch if Negative	if N=1, PC+2+2*nn→PC, else PC+2→PC
E3*nn*	BNC	nn	Relative Branch if Not Carry	if C=0, PC+2+2*nn→PC, else PC+2→PC
E7*nn*	BNN	nn	Relative Branch if Not Negative	if N=0, PC+2+2*nn→PC, else PC+2→PC
E5*nn*	BNOV	nn	Relative Branch if Not Overflow	if OV=0, PC+2+2*nn→PC, else PC+2→PC
E1*nn*	BNZ	nn	Relative Branch if Not Zero	if Z=0, PC+2+2*nn→PC, else PC+2→PC
E4*nn*	BOV	nn	Relative Branch if Overflow	if OV=1, PC+2+2*nn→PC, else PC+2→PC
D0*nd*	BRA	nd	Unconditional relative branch	PC+2+2*nd→PC
E0*nn*	BZ	nn	Relative Branch if Zero	if Z=1, PC+2+2*nn→PC, else PC+2→PC
EC*ml* F*mm*	CALL	mm:ml,s	Absolute Subroutine Call (second word)	PC+4 → TOS, mm:ml → PC<20:1>, if s=1, W → WS, STATUS → STATUSS, BSR → BSRS
EF*ml* F*mm*	GOTO	mm:ml	Absolute Branch (second word)	mm:ml → PC<20:1>
0000	NOP		No Operation	No operation
0006	POP		Pop Top/stack	TOS-1 → TOS
0005	PUSH		Push Top/stack	PC +2 → TOS
D8*nd*	RCALL	nd	Relative Subroutine Call	PC+2 → TOS, PC+2+2*nd→PC
00FF	RESET		Generate a Reset (same as $\overline{\text{MCLR}}$ reset)	same as $\overline{\text{MCLR}}$ reset
0010	RETFIE	s	Return from interrupt (and enable interrupts)	TOS → PC, 1 → GIE/GIEH or PEIE/GIEL, if s=1, WS → W, STATUSS → STATUS, BSRS → BSR, PCLATU/PCLATH are unchanged
0C*kk*	RETLW	kk	Return from subroutine, literal in W	kk → W
0012	RETURN	s	Return from subroutine	TOS → PC, if s=1, WS → W, STATUSS → STATUS, BSRS → BSR, PCLATU/PCLATH are unchanged
0003	SLEEP		Enter SLEEP Mode	0 → WDT, 0 → WDT postscaler, 1 → TO, 0 → PD

Enhanced 16-Bit Core Bit Operations

Hex	Mnemonic	Description	Function
9*bf*	BCF f,b,a	Bit Clear f	0 → f<b>
8*bf*	BSF f,b,a	Bit Set f	1 → f<b>
B*bf*	BTFSC f,b,a	Bit test f, skip if clear	if f<b>=0, PC+4→PC, else PC+2→PC
A*bf*	BTFSS f,b,a	Bit test f, skip if set	if f<b>=1, PC+4→PC, else PC+2→PC
7*bf*	BTG f,b,a	Bit Toggle f	f<b> → f<b>

Enhanced 16-Bit Core File Register Operation

Hex	Mnemonic	Description	Function
24*f*	ADDWF f,d,a	ADD WREG to f	W+f → dest
20*f*	ADDWFC f,d,a	ADD WREG and Carry bit to f	W+f+C → dest
14*f*	ANDWF f,d,a	AND WREG with f	W .AND. f → dest
6A*f*	CLRF f,a	Clear f	0 → f
1C*f*	COMF f,d,a	Complement f	f → dest
62*f*	CPFSEQ f,a	Compare f with WREG, skip if f=WREG	f–W, if f=W, PC+4 → PC else PC+2 → PC
64*f*	CPFSGT f,a	Compare f with WREG, skip if f > WREG	f–W, if f > W, PC+4 → PC else PC+2 → PC
60*f*	CPFSLT f,a	Compare f with WREG, skip if f < WREG	f–W, if f < W, PC+4 → PC else PC+2 → PC
04*f*	DECF f,d,a	Decrement f	f–1 → dest
2C*f*	DECFSZ f,d,a	Decrement f, skip if 0	f–1 → dest, if dest=0, PC+4 → PC else PC+2 → PC
4C*f*	DCFSNZ f,d,a	Decrement f, skip if not 0	f–1 → dest, if dest ≠ 0, PC+4 → PC else PC+2 → PC
28*f*	INCF f,d,a	Increment f	f+1 → dest
3C*f*	INCFSZ f,d,a	Increment f, skip if 0	f+1 → dest, if dest=0, PC+4 → PC else PC+2 → PC
48*f*	INFSNZ f,d,a	Increment f, skip if not 0	f+1 → dest, if dest ≠ 0, PC+4 → PC else PC+2 → PC
10*f*	IORWF f,d,a	Inclusive OR WREG with f	W .OR. f → dest
50*f*	MOVF f,d,a	Move f	f → dest
C*fs* F*fd*	MOVFF fs,fd	Move fs (first word) to fd (second word)	fs → fd
6E*f*	MOVWF f,a	Move WREG to f	W → f
02*f*	MULWF f,a	Multiply WREG with f	W * f → PRODH:PRODL
6C*f*	NEGF f,a	Negate f	f + 1 → PRODH:PRODL
34*f*	RLCF f,d,a	Rotate left f through Carry	
44*f*	RLNCF f,d,a	Rotate left f (no carry)	

Enhanced 16-Bit Core File Register Operation (Continued)

Hex	Mnemonic		Description	Function
30*f*	RRCF	f,d,a	Rotate right f through Carry	register f C → 7.....0
40*f*	RRNCF	f,d,a	Rotate right f (no carry)	register f 7.....0
68*f*	SETF	f,a	Set f	0xFF → f
54*f*	SUBFWB	f,d,a	Subtract f from WREG with Borrow	W - f - C → dest
5C*f*	SUBWF	f,d,a	Subtract WREG from f	f - W → dest
58*f*	SUBWFB	f,d,a	Subtract WREG from f with Borrow	f - W - C → dest
38*f*	SWAPF	f,d,a	Swap nibbbles of f	f<3:0> → dest<7:4>, f<7:4> → dest<3:0>
66*f*	TSTFSZ	f,a	Test f, skip if 0	PC+4 → PC, if f=0, else PC+2 → PC
18*f*	XORWF	f,d,a	Exclusive OR WREG with f	W .XOR. f → dest

PIC18CXXX Core Special Function Register Files

PRODH	FF4	INDF1	FE7
PRODL	FF3	POSTINC1	FE6
TOSU	FFF	POSTDEC1	FE5
TOSH	FFE	PREINC1	FE4
TOSL	FFD	PLUSW1	FE3
STKPTR	FFC	FSR1H	FE2
PCLATU	FFB	FSR1L	FE1
PCLATH	FFA	INDF2	FDF
PCL	FF9	POSTINC2	FDE
TBLPTRU	FF8	POSTDEC2	FDD
TBLPTRH	FF7	PREINC2	FDC
TBLPTRL	FF6	PLUSW2	FDB
TABLAT	FF5	FSR2H	FDA
INDF0	FEF	FSR2L	FD9
POSTINC0	FEE	WREG	FE8
POSTDEC0	FED	BSR	FE0
PREINC0	FEC	STATUS	FD8
PLUSW0	FEB	INTCON	FF2
FSR0H	FEA	INTCON2	FF1
FSR0L	FE9	INTCON3	FF0

ASCII Character Set

				Most Significant Character					
	Hex			2	3	4	5	6	7
Least Significant Character	0			Space	0	@	P	`	p
	1	SOH	DC1	!	1	A	Q	a	q
	2	STX	DC2	"	2	B	R	b	r
	3	ETX	DC3	#	3	C	S	c	s
	4	EOT	DC4	$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
	6	ACK	SYN	&	6	F	V	f	v
	7	Bell	ETB	'	7	G	W	g	w
	8	BS	CAN	(	8	H	X	h	x
	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L	\	l	|
	D	CR	GS	–	=	M	]	m	}
	E	SO	RS	.	>	N	^	n	~
	F	SI	US	/	?	O	_	o	DEL

MPLIB™ Usage Format

MPLIB object librarian is invoked with the following syntax:

```
mplib [/q] /{ctdrx} LIBRARY [MEMBER...]
```

options:

/c	create library;	creates a new LIBRARY with the listed MEMBER(s)
/t	list members;	prints a table showing the names of the members in the LIBRARY
/d	delete member;	deletes MEMBER(s) from the LIBRARY; if no MEMBER is specified the LIBRARY is not altered
/r	add/replace member;	if MEMBER(s) exist in the LIBRARY, then they are replaced, otherwise MEMBER is appended to the end of the LIBRARY
/x	extract member;	if MEMBER(s) exist in the LIBRARY, then they are extracted. If no MEMBER is specified, all members will be extracted
/q	quiet mode;	no output is displayed

MPLIB Usage Examples

Suppose a library named `dsp.lib` is to be created from three object modules named `fft.o`, `fir.o` and `iir.o`. The following command line would produce the desired results:

```
mplib /c dsp.lib fft.o fir.o iir.o
```

To display the names of the object modules contained in a library file names `dsp.lib`, the following command line would be appropriate:

```
mplib /t dsp.lib
```

Microchip Technology Inc.
2355 W. Chandler Blvd. • Chandler, AZ 85224 U.S.A.
Phone: 480-792-7200 • Fax: 480-792-9210
www.microchip.com

Printed in the U.S.A. 1/03 DS30400F