

PIC16F882/883/884/886/887

15.0 INSTRUCTION SET SUMMARY

The PIC16F882/883/884/886/887 instruction set is highly orthogonal and is comprised of three basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal and control** operations

Each PIC16 instruction is a 14-bit word divided into an **opcode**, which specifies the instruction type and one or more **operands**, which further specify the operation of the instruction. The formats for each of the categories is presented in [Figure 15-1](#), while the various opcode fields are summarized in [Table 15-1](#).

[Table 15-2](#) lists the instructions recognized by the MPASM™ assembler.

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator, which selects the bit affected by the operation, while 'f' represents the address of the file in which the bit is located.

For **literal and control** operations, 'k' represents an 8-bit or 11-bit constant, or literal value.

One instruction cycle consists of four oscillator periods; for an oscillator frequency of 4 MHz, this gives a normal instruction execution time of 1 μ s. All instructions are executed within a single instruction cycle, unless a conditional test is true, or the program counter is changed as a result of an instruction. When this occurs, the execution takes two instruction cycles, with the second cycle executed as a NOP.

All instruction examples use the format '0xhh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

15.1 Read-Modify-Write Operations

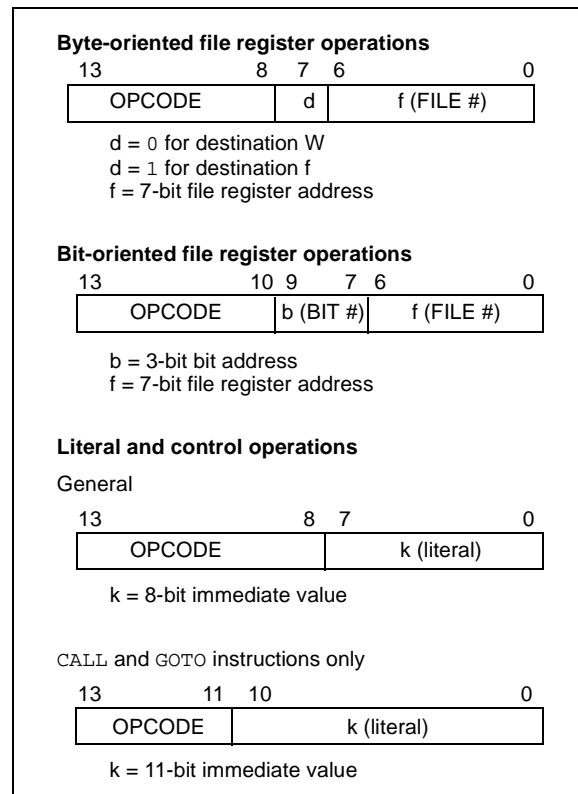
Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (RMW) operation. The register is read, the data is modified, and the result is stored according to either the instruction, or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

For example, a `CLRF PORTA` instruction will read PORTA, clear all the data bits, then write the result back to PORTA. This example would have the unintended consequence of clearing the condition that set the RAIF flag.

TABLE 15-1: OPCODE FIELD DESCRIPTIONS

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1.
PC	Program Counter
TO	Time-out bit
C	Carry bit
DC	Digit carry bit
Z	Zero bit
PD	Power-down bit

FIGURE 15-1: GENERAL FORMAT FOR INSTRUCTIONS



PIC16F882/883/884/886/887

TABLE 15-2: PIC16F882/883/884/886/887 INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status Affected	Notes
				MSb		LSb			
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C, DC, Z	1, 2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1, 2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRWF	—	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1, 2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1, 2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1, 2, 3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1, 2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1, 2, 3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1, 2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1, 2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		
NOP	—	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1, 2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1, 2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C, DC, Z	1, 2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1, 2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1, 2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		1, 2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		1, 2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	111x	kkkk	kkkk	C, DC, Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
CALL	k	Call Subroutine	2	10	0kkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	00	0000	0110	0100	\overline{TO} , \overline{PD}	
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLW	k	Move literal to W	1	11	00xx	kkkk	kkkk		
RETFIE	—	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk		
RETURN	—	Return from Subroutine	2	00	0000	0000	1000		
SLEEP	—	Go into Standby mode	1	00	0000	0110	0011	\overline{TO} , \overline{PD}	
SUBLW	k	Subtract w from literal	1	11	110x	kkkk	kkkk	C, DC, Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

- Note 1:** When an I/O register is modified as a function of itself (e.g., `MOVF GPIO, 1`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
- 2:** If this instruction is executed on the TMR0 register (and where applicable, $d = 1$), the prescaler will be cleared if assigned to the Timer0 module.
- 3:** If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

PIC16F882/883/884/886/887

15.2 Instruction Descriptions

ADDLW Add literal and W

Syntax: [*label*] ADDLW *k*
Operands: $0 \leq k \leq 255$
Operation: $(W) + k \rightarrow (W)$
Status Affected: C, DC, Z
Description: The contents of the W register are added to the 8-bit literal 'k' and the result is placed in the W register.

ADDWF Add W and f

Syntax: [*label*] ADDWF *f*,*d*
Operands: $0 \leq f \leq 127$
 $d \in [0,1]$
Operation: $(W) + (f) \rightarrow (\text{destination})$
Status Affected: C, DC, Z
Description: Add the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

ANDLW AND literal with W

Syntax: [*label*] ANDLW *k*
Operands: $0 \leq k \leq 255$
Operation: $(W) .\text{AND}. (k) \rightarrow (W)$
Status Affected: Z
Description: The contents of W register are AND'ed with the 8-bit literal 'k'. The result is placed in the W register.

ANDWF AND W with f

Syntax: [*label*] ANDWF *f*,*d*
Operands: $0 \leq f \leq 127$
 $d \in [0,1]$
Operation: $(W) .\text{AND}. (f) \rightarrow (\text{destination})$
Status Affected: Z
Description: AND the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

BCF Bit Clear f

Syntax: [*label*] BCF *f*,*b*
Operands: $0 \leq f \leq 127$
 $0 \leq b \leq 7$
Operation: $0 \rightarrow (f)$
Status Affected: None
Description: Bit 'b' in register 'f' is cleared.

BSF Bit Set f

Syntax: [*label*] BSF *f*,*b*
Operands: $0 \leq f \leq 127$
 $0 \leq b \leq 7$
Operation: $1 \rightarrow (f)$
Status Affected: None
Description: Bit 'b' in register 'f' is set.

BTFSC Bit Test f, Skip if Clear

Syntax: [*label*] BTFSC *f*,*b*
Operands: $0 \leq f \leq 127$
 $0 \leq b \leq 7$
Operation: skip if $(f) = 0$
Status Affected: None
Description: If bit 'b' in register 'f' is '1', the next instruction is executed. If bit 'b' in register 'f' is '0', the next instruction is discarded, and a NOP is executed instead, making this a 2-cycle instruction.

PIC16F882/883/884/886/887

BTFSS **Bit Test f, Skip if Set**

Syntax: [*label*] BTFSS f,b

Operands: $0 \leq f \leq 127$
 $0 \leq b < 7$

Operation: skip if (f) = 1

Status Affected: None

Description: If bit 'b' in register 'f' is '0', the next instruction is executed.
 If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction.

CLRWDT **Clear Watchdog Timer**

Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT
 0 → WDT prescaler,
 1 → \overline{TO}
 1 → \overline{PD}

Status Affected: \overline{TO} , \overline{PD}

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits \overline{TO} and \overline{PD} are set.

CALL **Call Subroutine**

Syntax: [*label*] CALL k

Operands: $0 \leq k \leq 2047$

Operation: (PC)+1 → TOS,
 k → PC<10:0>,
 (PCLATH<4:3>) → PC<12:11>

Status Affected: None

Description: Call Subroutine. First, return address (PC + 1) is pushed onto the stack. The 11-bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a 2-cycle instruction.

COMF **Complement f**

Syntax: [*label*] COMF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: (\bar{f}) → (destination)

Status Affected: Z

Description: The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

CLRF **Clear f**

Syntax: [*label*] CLRF f

Operands: $0 \leq f \leq 127$

Operation: 00h → (f)
 1 → Z

Status Affected: Z

Description: The contents of register 'f' are cleared and the Z bit is set.

DECF **Decrement f**

Syntax: [*label*] DECF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: (f) - 1 → (destination)

Status Affected: Z

Description: Decrement register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

CLRW **Clear W**

Syntax: [*label*] CLRW

Operands: None

Operation: 00h → (W)
 1 → Z

Status Affected: Z

Description: W register is cleared. Zero bit (Z) is set.

PIC16F882/883/884/886/887

DECFSZ Decrement f, Skip if 0

Syntax: [*label*] DECFSZ f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(f) - 1 \rightarrow (\text{destination});$
 skip if result = 0

Status Affected: None

Description: The contents of register 'f' are decremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.
 If the result is '1', the next instruction is executed. If the result is '0', then a NOP is executed instead, making it a 2-cycle instruction.

INCFSZ Increment f, Skip if 0

Syntax: [*label*] INCFSZ f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(f) + 1 \rightarrow (\text{destination}),$
 skip if result = 0

Status Affected: None

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.
 If the result is '1', the next instruction is executed. If the result is '0', a NOP is executed instead, making it a 2-cycle instruction.

GOTO Unconditional Branch

Syntax: [*label*] GOTO k

Operands: $0 \leq k \leq 2047$

Operation: $k \rightarrow \text{PC}<10:0>$
 $\text{PCLATH}<4:3> \rightarrow \text{PC}<12:11>$

Status Affected: None

Description: GOTO is an unconditional branch. The 11-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a 2-cycle instruction.

IORLW Inclusive OR literal with W

Syntax: [*label*] IORLW k

Operands: $0 \leq k \leq 255$

Operation: $(W) .\text{OR. } k \rightarrow (W)$

Status Affected: Z

Description: The contents of the W register are OR'ed with the 8-bit literal 'k'. The result is placed in the W register.

INCF Increment f

Syntax: [*label*] INCF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(f) + 1 \rightarrow (\text{destination})$

Status Affected: Z

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.

IORWF Inclusive OR W with f

Syntax: [*label*] IORWF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(W) .\text{OR. } (f) \rightarrow (\text{destination})$

Status Affected: Z

Description: Inclusive OR the W register with register 'f'. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.

PIC16F882/883/884/886/887

MOVF	Move f
Syntax:	[<i>label</i>] MOVF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(f) \rightarrow (\text{dest})$
Status Affected:	Z
Description:	The contents of register 'f' is moved to a destination dependent upon the status of 'd'. If $d = 0$, destination is W register. If $d = 1$, the destination is file register 'f' itself. $d = 1$ is useful to test a file register since status flag Z is affected.
Words:	1
Cycles:	1
Example:	<pre>MOVF FSR, 0</pre> <p>After Instruction</p> <p>W = value in FSR register</p> <p>Z = 1</p>

MOVWF	Move W to f
Syntax:	[<i>label</i>] MOVWF f
Operands:	$0 \leq f \leq 127$
Operation:	$(W) \rightarrow (f)$
Status Affected:	None
Description:	Move data from W register to register 'f'.
Words:	1
Cycles:	1
Example:	<pre>MOVWF OPTION F</pre> <p>Before Instruction</p> <p>OPTION = 0xFF</p> <p>W = 0x4F</p> <p>After Instruction</p> <p>OPTION = 0x4F</p> <p>W = 0x4F</p>

MOVLW	Move literal to W
Syntax:	[<i>label</i>] MOVLW k
Operands:	$0 \leq k \leq 255$
Operation:	$k \rightarrow (W)$
Status Affected:	None
Description:	The 8-bit literal 'k' is loaded into W register. The "don't cares" will assemble as '0's.
Words:	1
Cycles:	1
Example:	<pre>MOVLW 0x5A</pre> <p>After Instruction</p> <p>W = 0x5A</p>

NOP	No Operation
Syntax:	[<i>label</i>] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Description:	No operation.
Words:	1
Cycles:	1
Example:	<pre>NOP</pre>

PIC16F882/883/884/886/887

RLF Rotate Left f through Carry

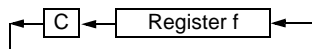
Syntax: [label] RLF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: See description below

Status Affected: C

Description: The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is stored back in register 'f'.



Words:

1

Cycles:

1

Example:

```
RLF    REG1,0

Before Instruction
REG1   = 1110 0110
C      = 0

After Instruction
REG1   = 1110 0110
W      = 1100 1100
C      = 1
```

RRF Rotate Right f through Carry

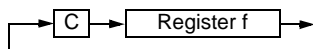
Syntax: [label] RRF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: See description below

Status Affected: C

Description: The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed back in register 'f'.



SLEEP Enter Sleep mode

Syntax: [label] SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler,
1 → \overline{TO} ,
0 → \overline{PD}

Status Affected: \overline{TO} , \overline{PD}

Description: The power-down Status bit, \overline{PD} is cleared. Time-out Status bit, \overline{TO} is set. Watchdog Timer and its prescaler are cleared. The processor is put into Sleep mode with the oscillator stopped.

SUBLW Subtract W from literal

Syntax: [label] SUBLW k

Operands: $0 \leq k \leq 255$

Operation: $k - (W) \rightarrow (W)$

Status Affected: C, DC, Z

Description: The W register is subtracted (2's complement method) from the 8-bit literal 'k'. The result is placed in the W register.

C = 0	$W > k$
C = 1	$W \leq k$
DC = 0	$W<3:0> > k<3:0>$
DC = 1	$W<3:0> \leq k<3:0>$

SUBWF Subtract W from f

Syntax: [label] SUBWF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: $(f) - (W) \rightarrow (\text{destination})$

Status Affected: C, DC, Z

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.

C = 0	$W > f$
C = 1	$W \leq f$
DC = 0	$W<3:0> > f<3:0>$
DC = 1	$W<3:0> \leq f<3:0>$

PIC16F882/883/884/886/887

SWAPF **Swap Nibbles in f**

Syntax: [*label*] SWAPF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: ($f<3:0>$) \rightarrow (destination<7:4>),
 ($f<7:4>$) \rightarrow (destination<3:0>)

Status Affected: None

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed in register 'f'.

XORLW **Exclusive OR literal with W**

Syntax: [*label*] XORLW k

Operands: $0 \leq k \leq 255$

Operation: (W) .XOR. k \rightarrow (W)

Status Affected: Z

Description: The contents of the W register are XOR'ed with the 8-bit literal 'k'. The result is placed in the W register.

XORWF **Exclusive OR W with f**

Syntax: [*label*] XORWF f,d

Operands: $0 \leq f \leq 127$
 $d \in [0,1]$

Operation: (W) .XOR. (f) \rightarrow (destination)

Status Affected: Z

Description: Exclusive OR the contents of the W register with register 'f'. If 'd' is '0', the result is stored in the W register. If 'd' is '1', the result is stored back in register 'f'.