SET DE INSTRUCCIONES PIC16F84A

ADDLW Suma un literal

Sintaxis: [label] ADDLW k
Operandos: $0 \le k \le 255$
Operación: : (W) + (k)⇒ (W)
Flags afectados: C, DC, Z
Código OP: 11 111x kkkk kkkk

Descripción: Suma el contenido del registro W y k, guardando el resultado en W.

Ejemplo: ADDLW 0xC2

Antes: W = 0x17
Después: W = 0xD9

ADDWF W + F

Sintaxis: [label] ADDWF f,d
Operandos: $d \in [0,1]$, $0 \le f \le 127$
Operación: (W) + (f) ⇒ (dest)
Flags afectados: C, DC, Z
Código OP: 00 0111 dfff ffff

Descripción: Suma el contenido del registro W y el registro f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: ADDWF REG,0

Antes: W = 0x17., REG = 0xC2
Después: W = 0xD9, REG = 0xC2

ANDLW W AND literal

Sintaxis: [label] ANDLW k
Operandos: $0 \le k \le 255$
Operación: : (W) AND (k)⇒ (W)
Flags afectados: Z
Código OP: 11 1001 kkkk kkkk

Descripción: Realiza la operación lógica AND entre el contenido del registro W y k, guardando el resultado en W.

Ejemplo: ADDLW 0xC2

Antes: W = 0x17
Después: W = 0xD9

ANDWF W AND F

Sintaxis: [label] ANDWF f,d
Operandos: $d \in [0,1]$, $0 \le f \le 127$
Operación: (W) AND (f) ⇒ (dest)
Flags afectados: Z
Código OP: 00 0101 dfff ffff

Descripción: Realiza la operación lógica AND entre los registros W y f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: : ANDWF REG,0

Antes: W = 0x17., REG = 0xC2
Después: W = 0x17, REG = 0x02

BCF Borra un bit

Sintaxis: [label] BCF f,b
Operandos: $0 \le f \le 127$, $0 \le b \le 7$
Operación: : 0 ⇒ (f<b>)
Flags afectados: Ninguno
Código OP: 01 00bb bfff ffff

Descripción: Borra el bit b del registro f

Ejemplo: : BCF REG,7

Antes: REG = 0xC7
Después: REG = 0x47

BSF Activa un bit

Sintaxis: [label] BSF f,b
Operandos: $0 \le f \le 127$, , $0 \le b \le 7$
Operación: 1 ⇒ (f<b>)
Flags afectados: Ninguno
Código OP: 01 01bb bfff ffff

Descripción: Activa el bit b del registro f

Ejemplo: : BSF REG,7

Antes: REG = 0x0A
Después: REG = 0x8A

BTFSC Test de bit y salto

Sintaxis: [label] BTFSC f,d
Operandos: $d \in [0,1]$, $0 \le f \le 127$
Operación: Salto si (f<b>) = 0
Flags afectados: Ninguno
Código OP: 01 10bb bfff ffff

Descripción: Si el bit b del registro f es 0, se salta una instrucción y se continúa con la ejecución. En caso de salto, ocupará dos ciclos de reloj.

Ejemplo:
```
BTFSC REG,6
GOTO NO_ES_0
SI_ES_0  Instrucción
NO_ES_0  Instrucción
```

BTFSS Test de bit y salto

Sintaxis: [label] BTFSS f,d
Operandos: $d \in [0,1]$, $0 \le f \le 127$
Operación: Salto si (f<b>) = 1
Flags afectados: Ninguno
Código OP: 01 11bb bfff ffff

Descripción: Si el bit b del registro f es 1, se salta una instrucción y se continúa con la ejecución. En caso de salto, ocupará dos ciclos de reloj.

Ejemplo:
```
BTFSS REG,6
GOTO NO_ES_0
SI_ES_0  Instrucción
NO_ES_0  Instrucción
```

CALL Salto a subrutina

Sintaxis: [label] CALL k
Operandos: $0 \le k \le 2047$
Operación: PC ⇒ Pila; k ⇒ PC
Flags afectados: Ninguno
Código OP: 10 0kkk kkkk kkkk

Descripción: Salto a una subrutina. La parte baja de k se carga en PCL, y la alta en PCLATCH. Ocupa 2 ciclos de reloj.

Ejemplo:ORIGEN CALL DESTINO

Antes: PC = ORIGEN
Después: PC = DESTINO

CLRF Borra un registro

Sintaxis: [label] CLRF f
Operandos: 0 ≤ f ≤ 127
Operación: : 0x00 ⇒ (f), 1 ⇒ Z
Flags afectados: Z
Código OP: 00 0001 1fff ffff

Descripción: El registro f se carga con 0x00. El flag Z se activa.

Ejemplo: : CLRF REG

Antes: REG = 0x5A
Después: REG = 0x00, Z = 1

CLRW Borra el registro W

Sintaxis: [label] CLRW
Operandos: Ninguno
Operación: : 0x00 ⇒ W, 1 ⇒ Z
Flags afectados: Z
Código OP: 00 0001 0xxx xxxx

Descripción: El registro de trabajo W se carga con 0x00. El flag Z se activa.

Ejemplo: : CLRW

Antes: W = 0x5A
Después: W = 0x00, Z = 1

CLRWDT Borra el WDT

Sintaxis: [label] CLRWDT
Operandos: Ninguno
Operación: 0x00 ⇒ WDT, 1 ⇒ /TO
1 ⇒ /PD
Flags afectados: /TO, /PD
Código OP: 00 0000 0110 0100
Descripción: Esta instrucción borra tanto el WDT como su preescaler. Los bits /TO y /PD del registro de estado se ponen a 1.

Ejemplo: : CLRWDT
Después: Contador WDT = 0,
Preescales WDT = 0,
/TO = 1, /PD = 1

COMF Complemento de f

Sintaxis: [label] COMF f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: : (/ f), 1 ⇒ (dest)
Flags afectados: Z
Código OP: 00 1001 dfff ffff

Descripción: El registro f es complementado. El flag Z se activa si el resultado es 0. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f..

Ejemplo: : COMF REG,0

Antes: REG = 0x13
Después: REG = 0x13, W = 0XEC

DECF Decremento de f

Sintaxis: [label] DECF f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: : (f) – 1 ⇒ (dest)
Flags afectados: Z
Código OP: 00 0011 dfff ffff

Descripción: Decrementa en 1 el contenido de f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: : DECF CONT,1

Antes: CONT = 0x01, Z = 0
Después: CONT = 0x00, Z = 1

DECFSZ Decremento y salto

Sintaxis: [label] DECFSZ f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: (f) -1 ⇒ d; Salto si R=0
Flags afectados: Ninguno
Código OP: 00 1011 dfff ffff

Descripción: Decrementa el contenido del registro f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f. Si la resta es 0 salta la siguiente instrucción, en cuyo caso costaría 2 ciclos.

Ejemplo: DECFSC REG,0
GOTO NO_ES_0
SI_ES_0 Instrucción
NO_ES_0 Salta instrucción anterior

GOTO Salto incondicional

Sintaxis: [label] GOTO k
Operandos: 0 ≤ k ≤ 2047
Operación: k ⇒ PC <8:0>
Flags afectados: Ninguno
Código OP: 10 1kkk kkkk kkkk

Descripción: Se trata de un salto incondicional. La parte baja de k se carga en PCL, y la alta en PCLATCH. Ocupa 2 ciclos de reloj.

Ejemplo: ORIGEN GOTO DESTINO

Antes: PC = ORIGEN
Después: PC = DESTINO

INCF Incremento de f

Sintaxis: [label] INCF f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: : (f) + 1 ⇒ (dest)
Flags afectados: Z
Código OP: 00 1010 dfff ffff

Descripción: Incrementa en 1 el contenido de f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: : INCF CONT,1

Antes: CONT = 0xFF, Z = 0
Después: CONT = 0x00, Z = 1

INCFSZ Incremento y salto

Sintaxis: [label] INCFSZ f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: (f) -1 ⇒ d; Salto si R=0
Flags afectados: Ninguno
Código OP: 00 1111 dfff ffff

Descripción: Incrementa el contenido del registro f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f. Si la resta es 0 salta la siguiente instrucción, en cuyo caso costaría 2 ciclos.

Ejemplo: INCFSC REG,0
GOTO NO_ES_0
SI_ES_0 Instrucción
NO_ES_0 Salta instrucción anterior

IORLW W OR literal

Sintaxis: [label] IORLW k
Operandos: $0 \le k \le 255$
Operación: : (W) OR (k) $\Rightarrow$ (W)
Flags afectados: Z
Código OP: 11 1000 kkkk kkkk

Descripción: Se realiza la operación lógica OR entre el contenido del registro W y k, guardando el resultado en W.

Ejemplo: : IORLW 0x35

Antes: W = 0x9A
Después: W = 0xBF

IORWF W AND F

Sintaxis: [label] IORWF f,d
Operandos: $d \in [0,1]$, $0 \le f \le 127$
Operación: (W) OR (f) $\Rightarrow$ (dest)
Flags afectados: Z
Código OP: 00 0100 dfff ffff

Descripción: Realiza la operación lógica OR entre los registros W y f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: : IORWF REG,0

Antes: W = 0x91, REG = 0x13
Después: W = 0x93, REG = 0x13

MOVLW Cargar literal en W

Sintaxis: [label] MOVLW f
Operandos: $0 \le f \le 255$
Operación: (k) $\Rightarrow$ (W)
Flags afectados: Ninguno
Código OP: 11 00xx kkkk kkkk

Descripción: El literal k pasa al registro W.

Ejemplo: MOVLW 0x5A

Después: REG = 0x4F, W = 0x5A

MOVF Mover a f

Sintaxis: [label] MOVF f,d
Operandos: $d \in [0,1]$, $0 \le f \le 127$
Operación: (f) $\Rightarrow$ (dest)
Flags afectados: Z
Código OP: 00 1000 dfff ffff
Descripción: El contenido del registro f se mueve al destino d. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f. Permite verificar el registro, puesto que afecta a Z.

Ejemplo: MOVF REG,0

Después: W = REG

MOVWF Mover a f

Sintaxis: [label] MOVWF f
Operandos: $0 \le f \le 127$
Operación: W $\Rightarrow$ (f)
Flags afectados: Ninguno
Código OP: 00 0000 1fff ffff

Descripción: El contenido del registro W pasa el registro f.

Ejemplo: MOVWF REG,0

Antes: REG = 0xFF, W = 0x4F
Después: REG = 0x4F, W = 0x4F

NOP No operar

Sintaxis: [label] NOP
Operandos: Ninguno
Operación: No operar
Flags afectados: Ninguno
Código OP: 00 0000 0xx0 0000

Descripción: No realiza operación alguna. En realidad consume un ciclo de instrucción sin hacer nada.

Ejemplo: : CLRWDT

Después: Contador WDT = 0,
Preescales WDT = 0,
/TO = 1, /PD = 1

RETFIE Retorno de interrup.

Sintaxis: [label] RETFIE
Operandos: Ninguno
Operación: : 1 $\Rightarrow$ GIE; TOS$\Rightarrow$PC
Flags afectados: Ninguno
Código OP: 00 0000 0000 1001

Descripción: El PC se carga con el contenido de la cima de la pila (TOS): dirección de retorno. Consume 2 ciclos. Las interrupciones vuelven a ser habilitadas.

Ejemplo: : RETFIE

Después: PC = dirección de retorno
GIE = 1

RETLW Retorno, carga W

Sintaxis: [label] RETLW k
Operandos: $0 \le k \le 255$
Operación: : (k)$\Rightarrow$ (W); TOS$\Rightarrow$PC
Flags afectados: Ninguno
Código OP: 11 01xx kkkk kkkk

Descripción: El registro W se carga con la constante k. El PC se carga con el contenido de la cima de la pila (TOS): dirección de retorno. Consume 2 ciclos.

Ejemplo: : RETLW 0x37

Después: PC = dirección de retorno
W = 0x37

RETURN Retorno de rutina

Sintaxis: [label] RETURN
Operandos: Ninguno
Operación: : TOS $\Rightarrow$ PC
Flags afectados: Ninguno
Código OP: 00 0000 0000 1000

Descripción: El PC se carga con el contenido de la cima de la pila (TOS): dirección de retorno. Consume 2 ciclos.

Ejemplo: : RETURN

Después: PC = dirección de retorno

RLF Rota f a la izquierda

Sintaxis: [label] RLF f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: Rotación a la izquierda
Flags afectados: C
Código OP: 00 1101 dfff ffff

Descripción: El contenido de f se rota a la izquierda. El bit de menos peso de f pasa al carry (C), y el carry se coloca en el de mayor peso. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: RRF REG,0

Antes: REG = 1110 0110, C = 0
Después: REG = 1110 0110,
W = 1100 1100, C = 1

RRF Rota f a la derecha

Sintaxis: [label] RRF f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: Rotación a la derecha
Flags afectados: C
Código OP: 00 1100 dfff ffff

Descripción: El contenido de f se rota a la derecha. El bit de menos peso de f pasa al carry (C), y el carry se coloca en el de mayor peso. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: RRF REG,0

Antes: REG = 1110 0110, C = 1
Después: REG = 1110 0110,
W = 01110 0011, C = 0

SLEEP Modo bajo consumo

Sintaxis: [label] SLEEP
Operandos: Ninguno
Operación: 0x00⇒WDT, 1 ⇒ / TO
0 ⇒ WDT Preescaler, 0 ⇒ / PD
Flags afectados: / PD, / TO
Código OP: 00 0000 0110 0011

Descripción: El bit de energia se pone a 0, y a 1 el de descanso. El WDT y su preescaler se borran. El micro para el oscilador, llendo al modo "durmiente".

Ejemplo: : SLEEP

Preescales WDT = 0,
/TO = 1, /PD = 1

SUBLW Resta Literal - W

Sintaxis: [label] SUBLW k
Operandos: 0 ≤ k ≤ 255
Operación: (k) - (W) ⇒ (W)
Flags afectados: Z, C, DC
Código OP: 11 110x kkkk kkkk
Descripción: Mediante el método del complemento a dos el contenido de W es restado al literal. El resultado se almacena en W.

Ejemplos: SUBLW 0x02

Antes:W=1,C=?. Después: W=1, C=1
Antes:W=2,C=?. Después: W=0, C=1
Antes:W=3,C=?.Después:W=FF,C=0
(El resultado es negativo)

SUBWF Resta f – W

Sintaxis: [label] SUBWF f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: (f) – (W)⇒ (dest)
Flags afectados: C, DC, Z
Código OP: 00 0010 dfff ffff
Descripción: Mediante el método del complemento a dos el contenido de W es restado al de f. . Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplos: SUBWF REG,1
Antes: REG = 0x03, W = 0x02, C = ?
Después:REG=0x01, W = 0x4F, C=1
Antes: REG = 0x02, W = 0x02, C = ?
Después:REG=0x00, W =0x02, C= 1
Antes: REG= 0x01, W= 0x02, C= ?
Después:REG=0xFF, W=0x02, C= 0
(Resultado negativo)

SWAPF Intercambio de f

Sintaxis: [label] SWAPF f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: : (f<3: 0>)⇔(f<7:4>)
Flags afectados: Ninguno
Código OP: 00 1110 dfff ffff

Descripción: Los 4 bits de más peso y los 4 de menos son intercambiados. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: : SWAPF REG,0

Antes: REG = 0xA5
Después: REG = 0xA5, W = 0x5A

XORLW W OR literal

Sintaxis: [label] XORLW k
Operandos: 0 ≤ k ≤ 255
Operación: : (W) XOR (k)⇒ (W)
Flags afectados: Z
Código OP: 11 1010 kkkk kkkk

Descripción: Se realiza la operación lógica XOR entre el contenido del registro W y k, guardando el resultado en W.

Ejemplo: : XORLW 0xAF

Antes: W = 0xB5
Después: W = 0x1A

XORWF W AND F

Sintaxis: [label] XORWF f,d
Operandos: d ∈ [0,1], 0 ≤ f ≤ 127
Operación: (W) XOR (f) ⇒ (dest)
Flags afectados: Z
Código OP: 00 0110 dfff ffff

Descripción: Realiza la operación lógica XOR entre los registros W y f. Si d es 0, el resultado se almacena en W, si d es 1 se almacena en f.

Ejemplo: : XORWF REG,0

Antes: W = 0xB5, REG = 0xAF
Después: W = 0xB5, REG = 0x1A

Pin Diagrams

PDIP, SOIC

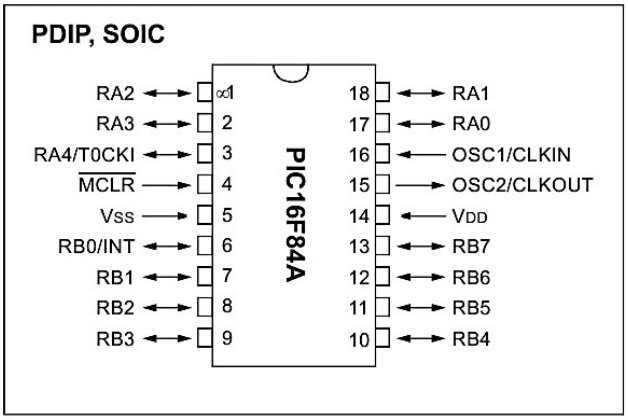


TABLE 1-1: PIC16F84A PINOUT DESCRIPTION

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
MCLR	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST(1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt-on-change pin.
RB5	11	11	12	I/O	TTL	Interrupt-on-change pin.
RB6	12	12	13	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming data.
VSS	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	P	—	Positive supply for logic and I/O pins.

Legend: I= input O = Output — = Not used I/O = Input/Output TTL = TTL input P = Power ST = Schmitt Trigger input

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A

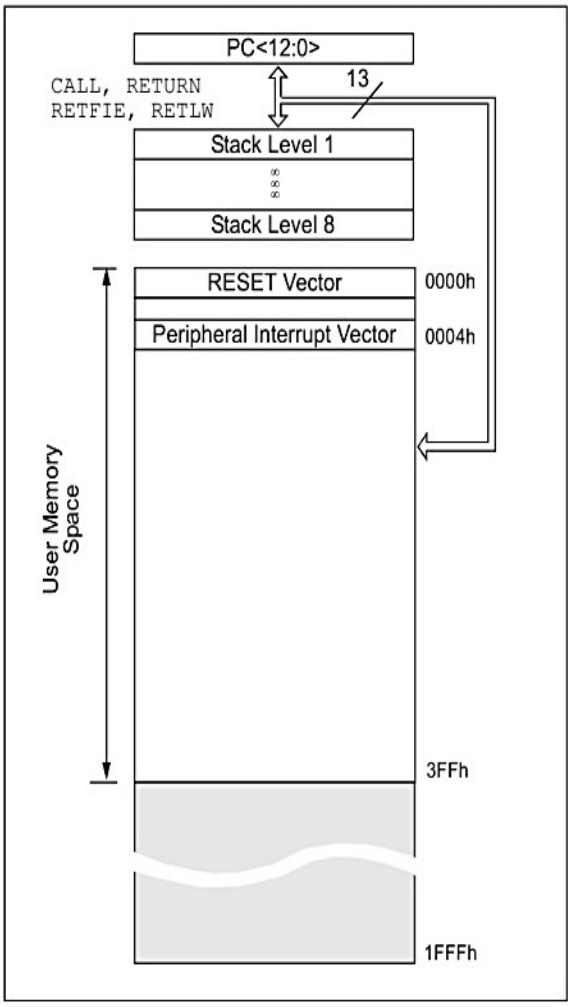
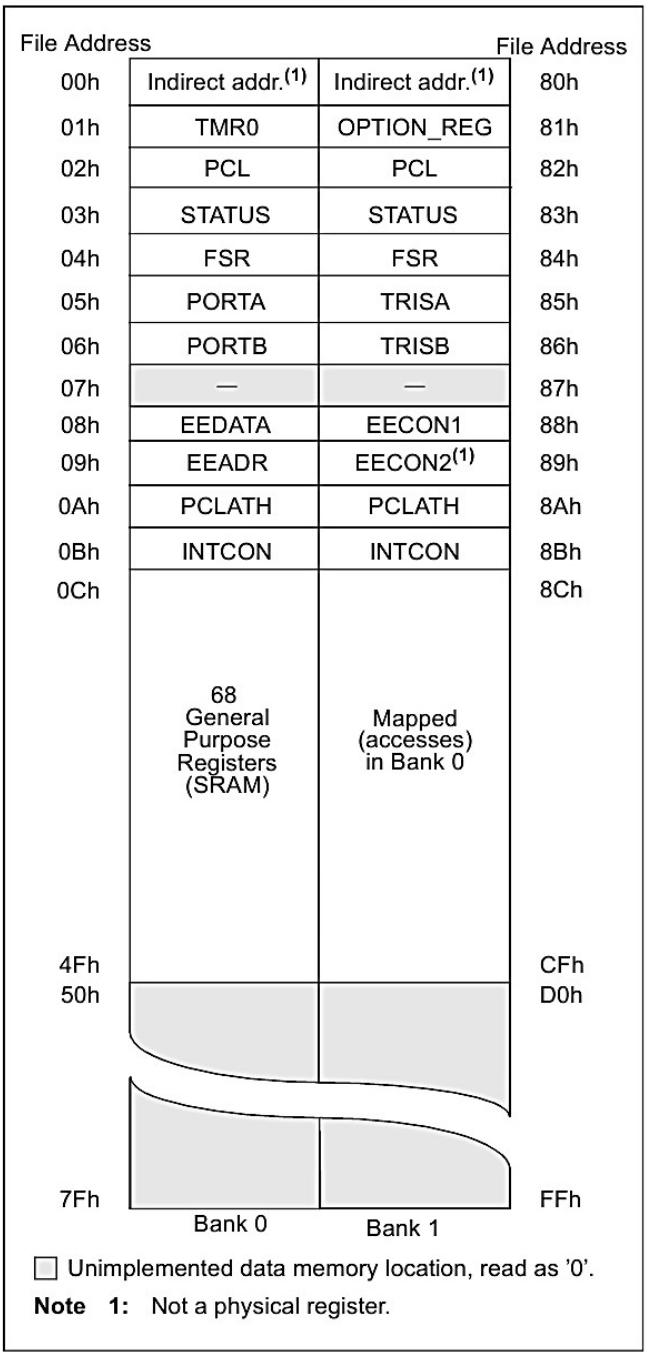


FIGURE 2-2: REGISTER FILE MAP - PIC16F84A



REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7-6 **Unimplemented:** Maintain as '0'

bit 5 **RP0**: Register Bank Select bits (used for direct addressing)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

bit 4 $\overline{\textbf{TO}}$: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 $\overline{\textbf{PD}}$: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{\text{borrow}}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{\text{borrow}}$, the polarity is reversed)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 2-2: OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 $\overline{\textbf{RBPU}}$: PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG**: Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS**: TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE**: TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA**: Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0**: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **EEIE**: EE Write Complete Interrupt Enable bit
1 = Enables the EE Write Complete interrupts
0 = Disables the EE Write Complete interrupt

bit 5 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown