Microcontroladores PIC: Organización de la Memoria

ORGANIZACIÓN de la MEMORIA

Dentro del PIC16F877 se distinguen tres bloques de memoria.

- Memoria de programa

En sus 8192 posiciones (8Kx14bits) contiene el programa con las instrucciones que gobiernan la aplicación. Es del tipo no volátil.

- Memoria de datos RAM

Guarda las variables y datos. Son registros de 8 bits. Es volátil.

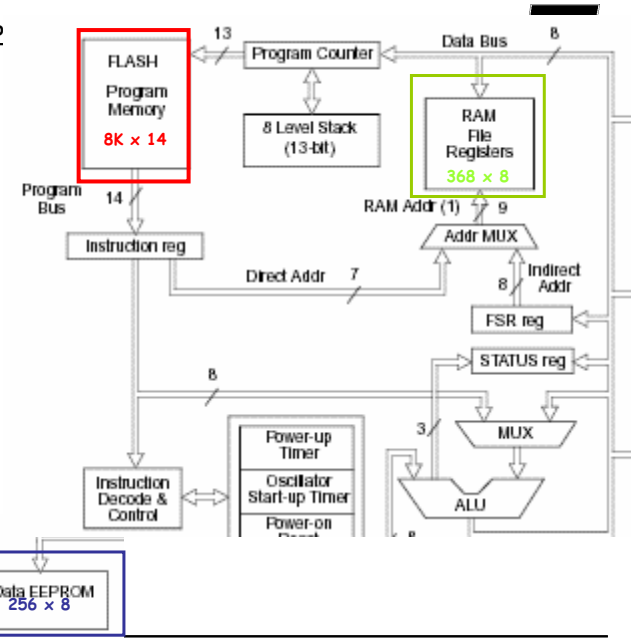
- Memoria EEPROM de datos

Es una pequeña área de memoria de datos de lectura y escritura no volátil que permite garantizar que determinada información estará siempre disponible al reinicializarse el programa. Se gestiona de manera distinta a la memoria de datos RAM.

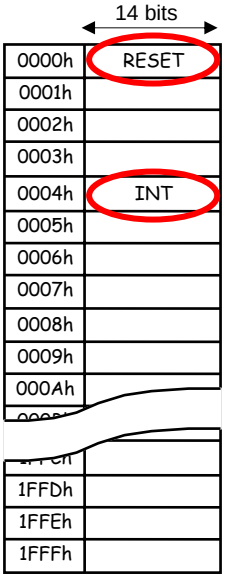
Organización de la Memo

Memoria
de Programa
y de datos

Microcontrolador
PIC16F877
(familia media)



Organización de la Memoria — La memoria de programa



- Almacena todas las instrucciones del programa en ejecución, que estará grabado de forma permanente.
- La información contenida en esta memoria se graba previamente mediante un equipo físico denominado programador o grabador.
- El PIC16F877 tiene una memoria de programa no volátil denominada ROM Flash que admite unas 1000 grabaciones.
- La memoria de programa está organizada en palabras de 14 bits cada una.
- Todas las instrucciones ocupan una posición de memoria de programa
- A la dirección 0 se accederá tras un RESET y a la posición 4 tras una interrupción cualquiera

MEMORIA DE PROGRAMA – CONTADOR DE PROGRAMA (PC)

• A la memoria de programa de los PIC16 (entre ellos los PIC16F87X) se accede mediante un registro de 13 bits que actúa como puntero de la instrucción que se debe ejecutar en el siguiente ciclo de instrucción.

• Este registro de 13 bits se denomina Contador de Programa (Program Counter) y su acrónimo es PC.

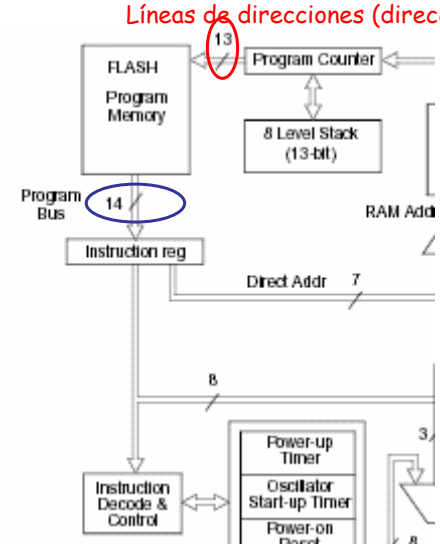
• Con este contador de programa de 13 bits se pueden direccionar 8K posiciones de memoria. Debido a que la codificación de las instrucciones son de 14 bits y para aprovechar las ventajas de la arquitectura Harvard, cada una de esas 8K posiciones corresponde a una instrucción y por tanto, el contador de programa es capaz de direccionar 8K x 14 posiciones ó 8Kinstrucciones.

• Como cada instrucción ocupa una posición de memoria resulta mucho más fácil saber si un dispositivo tiene memoria de programa suficiente para una aplicación.

Acceso a la Memoria de Programa

Líneas de direcciones (direccionamiento)

Líneas de datos de la memoria de programa (contenido)

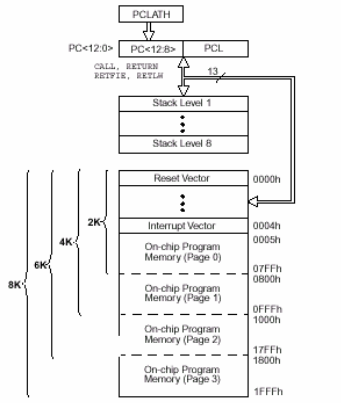


MEMORIA DE PROGRAMA – CONTADOR DE PROGRAMA

• Los 8K de memoria de programa disponible están divididos en 4 páginas de 2K cada una (0h-7FFh, 800h-FFFh, 1000h-17FFh y 1800h-1FFFh). Esto es debido a que las instrucciones de salto y llamada a subprograma permiten cargar sólo 11 bits en el PC (desplazamiento en 2^{11} = 2K)

•Si se están ejecutando instrucciones secuencialmente, el contador de programa pasará de una página a otra sin necesidad de intervención por parte del usuario o programador.

•Para saltar entre páginas de la memoria de programa los 2 bits más altos del PC deben modificarse. Esto se realiza escribiendo en el registro PCLATH (es un registro situado en la memoria de datos).

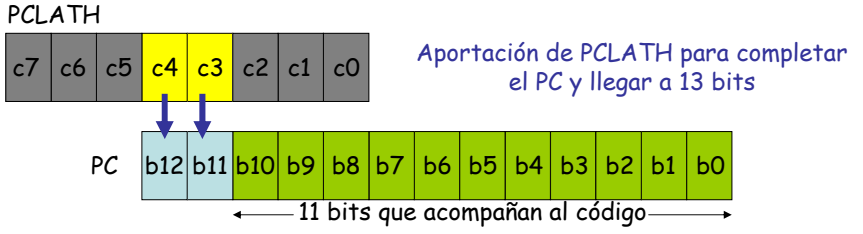


PC y PCLATH

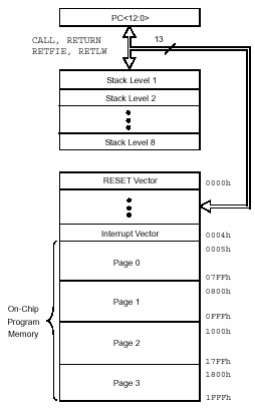
• El PC de 13 bits se divide en dos registros: PCH (parte alta) y PCL (parte baja). El registro PCL es uno más de los registros de RAM, pero PCH no es accesible directamente

• Cada vez que se opere con PCL o se cargue el PC con 11 bits procedentes de un salto o llamada a subprograma, un registro denominado PCLATH, aportará los bits que le falten al PC para llegar a los 13 (aporta 5 bits ó 2 bits).

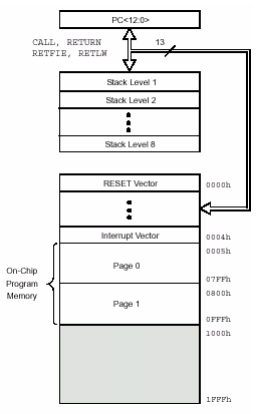
• El PCLATH no es la parte alta del PC, sino que es un complemento al PCL o a la dirección que aporta una instrucción de salto o llamada a subprograma



MEMORIA DE PROGRAMA – CONTADOR DE PROGRAMA



MAPA DE MEMORIA DE PROGRAMA EN UN PIC16F876/877



MAPA DE MEMORIA DE PROGRAMA EN UN PIC16F873/874

- En algunos dispositivos solo una parte del mapa de memoria total posible está implementado.

- En la figura se muestra un dispositivo con 4K de memoria de programa y otro con 8K de memoria de programa.

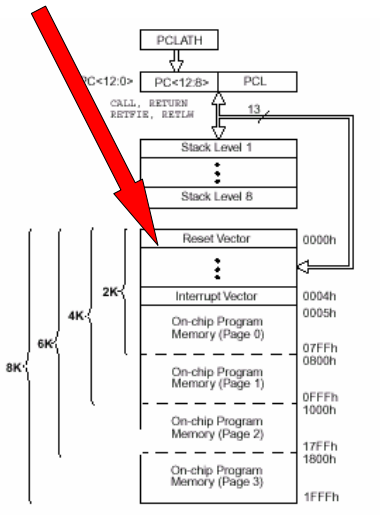
- Los dispositivos con solo 2K de memoria de programa no necesitan paginado.

MEMORIA DE PROGRAMA – VECTOR DE RESET

- El vector de RESET está siempre en la posición 0000h de la memoria de programa.
- Cualquier reset (interno o externo) que se genere en un microcontrolador hará que su contador de programa pase a tener el valor 0000h y que por tanto el microcontrolador pase a ejecutar la instrucción situada en dicha posición.
- El RESET también limpia el contenido del registro PCLATH.

Ejemplo de programa:

```
            ORG 0X00
            GOTO INICIO

            ORG 0X05
INICIO      BSF STATUS,RP0
            MOVLW 0X00
            MOVWF TRISB
            ....
```

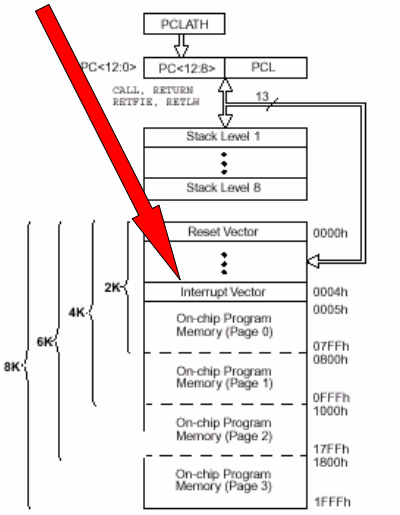


MEMORIA DE PROGRAMA – VECTOR DE INTERRUPCION

• El vector de INTERRUPCION está siempre en la posición 0004h de la memoria de programa.
• Cualquier interrupción que se fuerce a un microcontrolador hará que su contador de programa pase a tener el valor 0004h y que por tanto el microcontrolador pase a ejecutar la instrucción situada en dicha posición.
• El salto a la rutina de interrupción no modifica el contenido del PCLATH, por lo que cualquier modificación que se realice del PC en la rutina de interrupción debe realizarse con cuidado de lo que se tiene cargado en el PCLATH

Ejemplo de programa:

```
            ORG 0X04
             GOTO RSI

             ORG 0XF0
RSI          BTFSC INTCON,RBIF
            GOTO RSI_PORTB
            ....
```



MEMORIA DE PROGRAMA – Información de calibración

• En algunos dispositivos, especialmente en aquellos que tienen la opción de utilizar como oscilador una red RC interna, viene grabada en la memoria de programa una información de calibración.

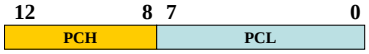
•Esta información es programada por MICROCHIP cuando el dispositivo está en la fase final de test.

•Dicha información de calibración está programada habitualmente al final de la memoria de programa y está programada como una instrucción RETLW, donde el valor literal es la información de calibración que ha de cargarse en un registro llamado OSCCAL.

•La utilización de este valor permite obtener una frecuencia de reloj mas aproximada a la nominal.

CONTADOR DE PROGRAMA

El contador de programa (PC) es un registro de 13 bits que se descompone en 2 registros:



El byte bajo PC<0:7> se denomina PCL y está disponible en la memoria de datos. Es un registro que se puede leer y escribir directamente desde programa.

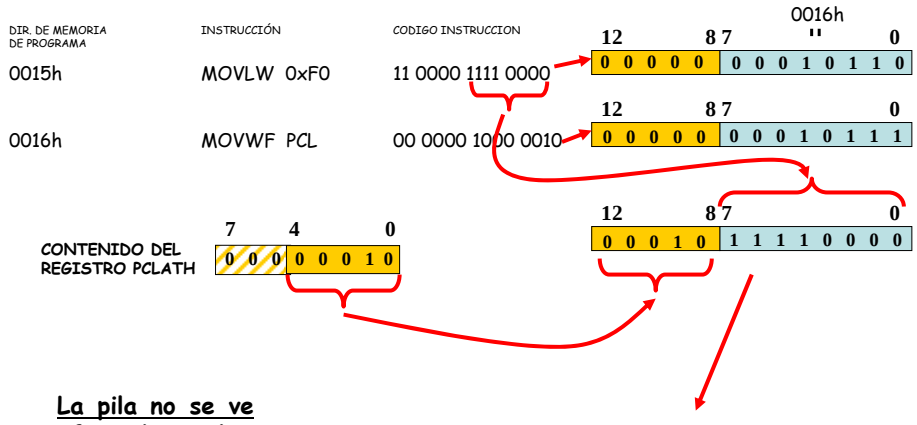
El "cuasibyte" alto PC<8:12> se denomina PCH y no está disponible en la memoria de datos. Este registro no se puede leer ni escribir directamente. La escritura debe realizarse usando como registro intermedio el registro PCLATH que sí es un registro de 8 bits accesible en la memoria de datos del microcontrolador. El contenido del registro PCLATH se transfiere a la parte alta del PC en el momento en que se escribe en el registro PCL. También aporta dos bits al PC en los saltos.

Las instrucciones que modifican el PCL y que por tanto pueden modificar el PCH son las siguiente:

- Instrucciones que tengan el PCL como destino. Ej. MOVWF PCL
- Instrucciones GOTO
- Instrucciones CALL

CONTADOR DE PROGRAMA

Instrucciones que tengan el PCL como destino



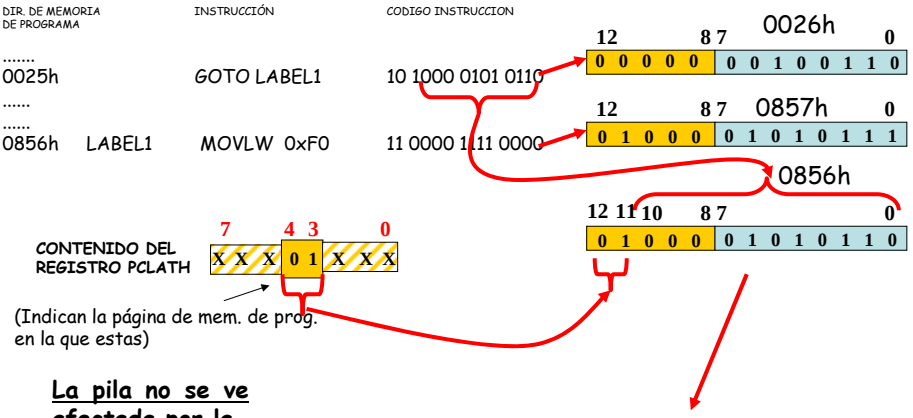
La pila no se ve afectada por la instrucción

Próxima instrucción a ejecutar 02F0h

CONTADOR DE PROGRAMA

Instrucción GOTO

DIR. DE MEMORIA DE PROGRAMA		INSTRUCCIÓN	CODIGO INSTRUCCION
.....			
0025h		GOTO LABEL1	10 1000 0101 0110
.....			
.....			
0856h	LABEL1	MOVLW 0xF0	11 0000 1111 0000



(Indican la página de mem. de prog. en la que estas)

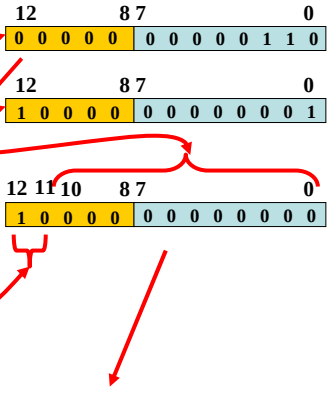
La pila no se ve afectada por la instrucción

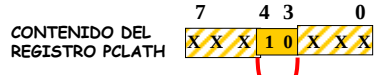
Próxima instrucción a ejecutar 0856h

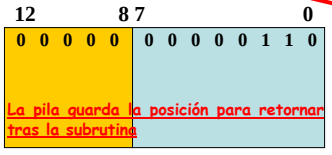
CONTADOR DE PROGRAMA

Instrucción CALL

DIR. DE MEMORIA DE PROGRAMA		INSTRUCCIÓN	CODIGO INSTRUCCION
0004h		.....	
0005h		CALL SUBRUT1	10 0000 0000 0000
.....			
.....			
1000h	SUBRUT1	MOVLW 0xF0	11 0000 1111 0000
.....			
1100h		RETURN	





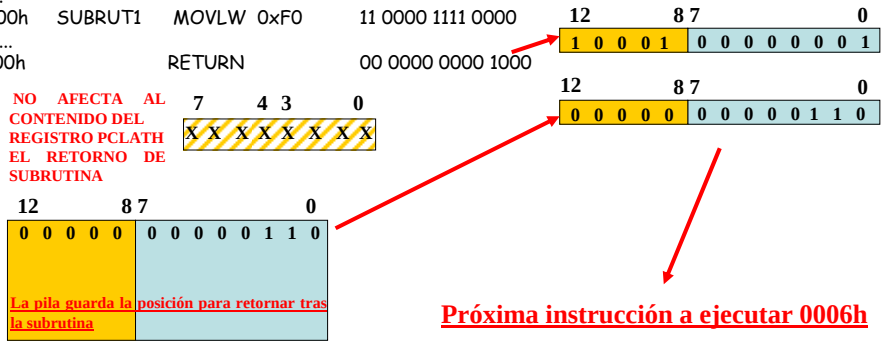


Próxima instrucción a ejecutar 1000h

CONTADOR DE PROGRAMA

Instrucción RETURN, RETFIE ó RETLW

DIR. DE MEMORIA DE PROGRAMA		INSTRUCCIÓN	CODIGO INSTRUCCION
0004h		.....	
0005h		CALL SUBRUT1	10 0000 0000 0000
.....			
.....			
1000h	SUBRUT1	MOVLW 0xF0	11 0000 1111 0000
.....			
1100h		RETURN	00 0000 0000 1000



NO AFECTA AL CONTENIDO DEL REGISTRO PCLATH EL RETORNO DE SUBRUTINA

Próxima instrucción a ejecutar 0006h

CONTADOR DE PROGRAMA

Debe tenerse cuidado al implementar tablas de conversión (como por ejemplo en las utilizadas en teclados matriciales) que la tabla no quede a caballo entre 2 páginas de memoria.

En la porción de programa para un PIC16F877 que se adjunta, se puede apreciar como en un determinado momento se pretende cargar el contenido del PORTB (que puede variar entre 0x00 y 0x1F), sumarle 0x20 y sacar el valor obtenido por el PORTC (suponer los registros asociados a los puertos programados oportunamente). Se observa, no obstante, que para determinados valores del PORTB la operación no se ejecuta correctamente. ¿Por qué?. Propóngase una posible solución para resolver el problema.

```
        include "p16F877.inc"
        org     00h             ;Reset
Vector
        goto    Start
        org     0F0h
        nop
        org     100h
Start
        bsf     STATUS,RP0
        clrf    TRISC
        bcf     STATUS,RP0
Bucle
        movlw   0x07
        movwf   PCLATH
        movf    PORTB,w
        call    TABLA
        movwf   PORTC
        goto    Bucle
org     7F0h
TABLA
        addwf   PCL,f
        retlw   0x20
        retlw   0x21
        retlw   0x22
        retlw   0x23
        retlw   0x24
        retlw   0x25
        retlw   0x26
        retlw   0x27
        retlw   0x28
        retlw   0x29
        retlw   0x2A
        retlw   0x2B
        retlw   0x2C
        retlw   0x2D
        retlw   0x2E
        retlw   0x2F
        retlw   0x30
        retlw   0x31
        retlw   0x32
        retlw   0x33
        retlw   0x34
        retlw   0x35
        retlw   0x36
        retlw   0x37
        retlw   0x38
        retlw   0x39
        retlw   0x3A
        retlw   0x3B
        retlw   0x3C
        retlw   0x3D
        retlw   0x3E
        retlw   0x3F
        end
```

LA PILA PARA ALMACENAR EL PC (STACK)

• La pila permite almacenar las direcciones (PCs) a donde debe retornar el programa cuando se finaliza una llamada a una subrutina o cuando se finaliza la ejecución de una rutina de interrupción.

• Los microcontroladores de la familia PIC16 tienen una pila de 8 niveles x 13 bits, que por tanto permite concatenar como máximo 8 saltos a subrutinas (CALLs) o ejecuciones de rutinas de interrupción (salto a 0004h).

• El espacio de memoria para la pila no forma parte de la memoria de programa ni de la memoria de datos que tiene el microcontrolador. Es un espacio de memoria totalmente independiente. El puntero de pila no se puede leer ni escribir. La pila es gestionada por el hardware.

• Al contrario que en otros micros, no se dispone de una pila en RAM que se pueda gestionar por software (no hay inst."PUSH" ni "POP")

¿Cuál o cuales de los siguientes programas para un PIC16C67 realizará correctamente la puesta a cero de TMR0? Razónese la respuesta

```
            INCLUDE P16C67.INC

            ORG 0X00
            BSF     PCLATH,3
            CALL    SUBRUT1
ESPERA      GOTO    ESPERA

            ORG     0X800
            NOP

            ORG     0X1800
SUBRUT1     CLRF    TMR0
            BCF     PCLATH,3
            RETURN

            END
```

NO

```
            INCLUDE P16C67.INC

            ORG 0X00
            MOVLW   0XFF
            MOVWF   PCLATH
            CALL    SUBRUT1
ESPERA      GOTO    ESPERA

            ORG     0X800
            NOP

            ORG     0X1800
SUBRUT1     CLRF    TMR0
            CLRF    PCLATH
            RETURN

            END
```

SI

```
            INCLUDE P16C67.INC

            ORG 0X00
            BSF     PCLATH,3
            BSF     PCLATH,4
            CALL    SUBRUT1
ESPERA      GOTO    ESPERA

            ORG     0X800
            NOP

            ORG     0X1800
SUBRUT1     CLRF    TMR0
;           CLRF    PCLATH
            RETURN

            END
```

SI

¿Cuál o cuales de los siguientes programas para PIC16C67 realizará correctamente la ejecución desde la posición 0x100 hasta la etiquetada como WAIT?. Suponer que TECLA es una variable donde se tiene el valor para el desplazamiento dentro de TABLA. Razónese la respuesta.

a)

```
  INCLUDE
P16C67.INC

   ORG  0X100
   MOVLW    0x10
   MOVWF    PCLATH
   MOVF     TECLA,W
   CALL     TABLA
   CLRF     PCLATH
WAIT  GOTO   WAIT

   ORG  0X1000
TABLA
   ADDWF    PCL,F
   RETLW    0x01
   RETLW    0x02
.................
   RETLW....0x0F
END
```

SI

b)

```
INCLUDE P16C67.INC

   ORG  0X100
   BSF    PCLATH,4
   MOVF    TECLA,W
   CALL    TABLA
WAIT  GOTO   WAIT

   ORG  0X1000
TABLA
   ADDWF    PCL,F
   RETLW    0x01
   RETLW    0x02
.................
   RETLW....0x0F
END
```

NO

c)

```
INCLUDE P16C67.INC

   ORG  0X100
   MOVLW    0x1F
   MOVWF    PCLATH
   MOVF     TECLA,W
   CALL     TABLA
   CLRF     PCLATH
WAIT  GOTO   WAIT

   ORG  0X1000
TABLA
   ADDWF    PCL,F
   RETLW    0x01
   RETLW    0x02
.................
   RETLW....0x0F
END
```

NO

d)

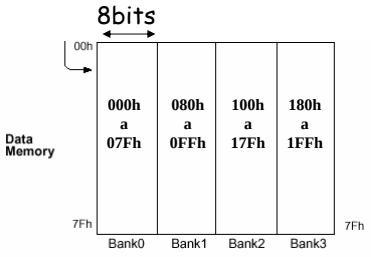
```
INCLUDE P16C67.INC

   ORG  0X100
   BSF    PCLATH,3
   MOVF    TECLA,W
   CALL    TABLA
   CLRF    PCLATH
WAIT  GOTO   WAIT

   ORG  0X1000
TABLA
   ADDWF    PCL,F
   RETLW    0x01
   RETLW    0x02
.................
   RETLW....0x0F
END
```

NO

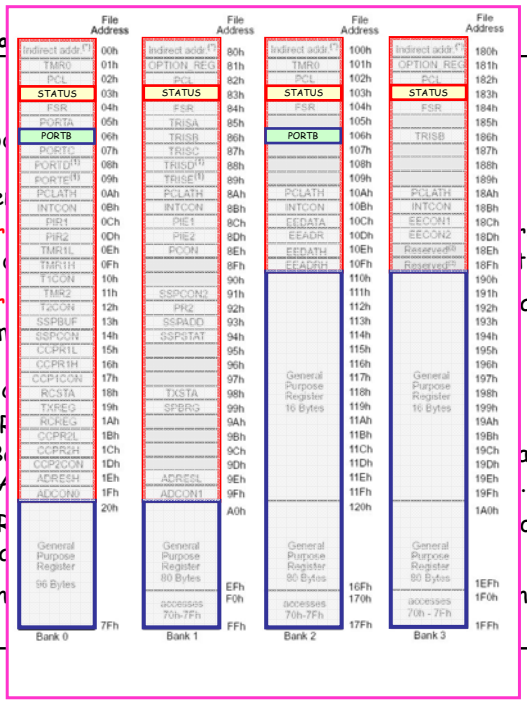
MEMORIA RAM DE DATOS

La memoria de datos está distribuida en 4 posibles bancos de 128 bytes cada uno. Por tanto, la máxima cantidad de memoria disponible en los microcontroladores PIC16 es de 512 bytes. No obstante, no están implementadas todas las posiciones de memoria en todos los bancos. Por ejemplo, un PIC16F876 solo tiene implementadas 368 posiciones de memoria de datos.

La memoria de datos se organiza en bancos de 128 bytes porque cuando se indica una dirección de operando fuente, sólo se pueden incluir 7 bits en la codificación. Nos movemos pues en 2^7=128 bytes ¿y los otros 2 bits?



A cada posición de memoria se le denomina registro.



- Almacena to
- Se distingue
  - ➢ Registr
    Cada uno
  - ➢ Registr
    datos tem
- Cuenta con
  - ➢ Los SFR
    9Fh del B
    Banco 3. A
  - ➢ Los GPR
    Bancos 1 a
  - ➢ Existen
    lectura.

ros registros.
trol del PIC.
ara guardar
0, de 80h a
a 18Fh del
osiciones de los
n '0' en caso de

MODOS DE DIRECCIONAMIENTO DE LA MEMORIA DE DATOS

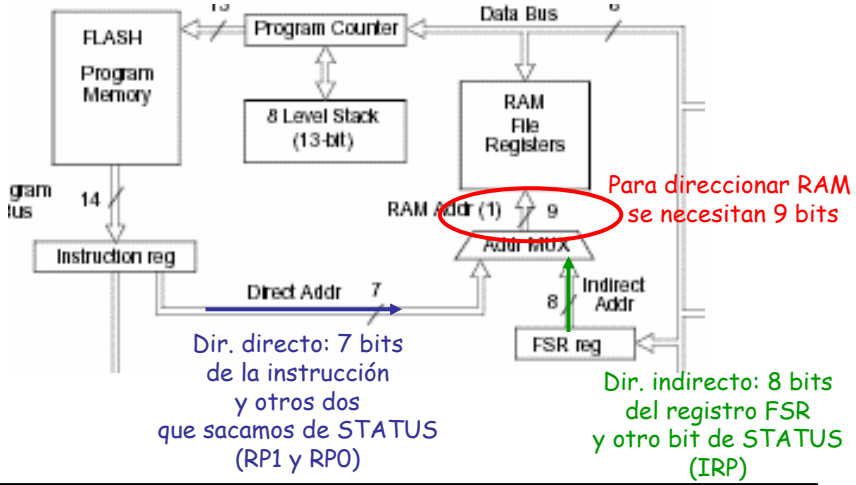
- La memoria de datos está distribuida en 4 posibles bancos de 128 bytes cada uno.

- Existen 2 modos de direccionamiento para acceder a cualquiera de las posiciones de la memoria de datos:
  Direccionamiento directo
  Direccionamiento indirecto

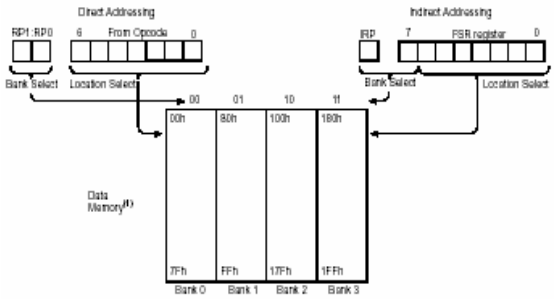
- Direccionamiento directo: La posición de memoria con la que se trabaja viene directamente definida en el código de la instrucción.

- Direccionamiento indirecto: La posición de memoria con la que se trabaja viene definida por el contenido de el registro FSR (Posición 04h, 84h, 104h ó 184h), es decir, el registro FSR actúa como puntero de la posición de memoria con la que se pretende operar.

MODOS DE DIRECCIONAMIENTO DE LA MEMORIA DE DATOS



MODOS DE DIRECCIONAMIENTO DE LA MEMORIA DE DATOS



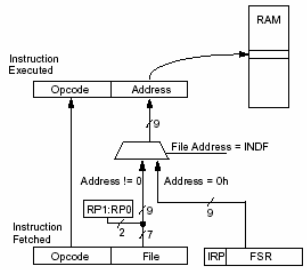
BANCO ACCEDIDO	DIRECTO (RP1:RP0)	INDIRECTO (IRP)
0	0 0	0
1	0 1	
2	1 0	1
3	1 1	

Tanto para el direccionamiento directo como el indirecto la dirección completa (9 bits) no se puede obtener del código de la operación (7 bits) o del registro FSR (8 bits).

Para direccionamiento directo, la dirección a operar se obtiene completando la dirección incluida en el código de la instrucción con los bits RP1:RP0 del registro STATUS. (Ver tabla)

Para direccionamiento indirecto, la dirección se obtiene completando el contenido del registro FSR con el bit IRP del registro STATUS. (Ver tabla).

DIRECCIONAMIENTO INDIRECTO



1) SE LEE EL REGISTRO CONTENIDO EN EL CODIGO DE LA INSTRUCCIÓN.
2) SI EL REGISTRO ES DISTINTO DEL CORRESPONDIENTE A INDF (00h,80h,100h ó 180h) LA INSTRUCCIÓN SE EJECUTA SOBRE EL REGISTRO INDICADO EN LA INSTRUCCIÓN COMPLEMENTADO CON LOS BITS RP1:RP0 DEL REGISTRO STATUS (DIRECCIONAMIENTO DIRECTO)
3) SI EL REGISTRO CORRESPONDE A LA POSICION DE INDF (00h,80h,100h ó 180h), LA INSTRUCCIÓN SE EJECUTA SOBRE EL REGISTRO INDICADO EN EL REGISTRO FSR COMPLEMENTADO CON EL BIT IRP DEL REGISTRO STATUS

Ejemplo de direccionamiento indirecto donde se limpian las posiciones de memoria de datos comprendidas entre la posición 20h y la 2Fh (ambas incluidas)

```
          BCF STATUS, IRP     ; dir. Indirecto, bancos 0/1
          MOVLW 0x20          ; Inicializa puntero a RAM
          MOVWF FSR           ;
NEXT      CLRF INDF           ; limpia registro INDF
          INCF FSR,F          ; Inc puntero
          BTFSS FSR,4         ; Todo limpio?
          GOTO NEXT           ; NO, limpia siguiente
CONTINUE  :                   ; SI, continua
```


Ejemplo de direccionamiento indirecto donde se limpian todas las posiciones de memoria de datos correspondientes a registros de propósito general (GPR).

```
    CLRF STATUS     ; Limpia STATUS (Banco 0)
    MOVLW 0x20      ; 1ª dir de los GPRs en el banco
    MOVWF FSR       ; Moverlo al reg, de dir. ind.
Bank0_LP
    CLRF INDF0      ; Limpia el GPR apuntado por FSR
    INCF FSR        ; Inc.puntero
    BTFSS FSR, 7    ; Fin de banco? (FSR = 80h, C = 0)
    GOTO Bank0_LP   ; NO, limpia siguiente
;
; Banco 1
; (**Solo si el micro tiene banco 1 **)
;
    MOVLW 0xA0      ; 1ª dir de los GPRs en el banco
    MOVWF FSR       ; Moverlo al reg, de dir. ind.
Bank1_LP
    CLRF INDF0      ; Limpia el GPR apuntado por FSR
    INCF FSR        ; Inc.puntero
    BTFSS STATUS,C  ; Fin de banco? (FSR = 00h, C = 1)
    GOTO Bank1_LP   ; NO, limpia siguiente
;
; Banco 2
; (**Solo si el micro tiene banco 2 **)
;
;
    BSF STATUS, IRP ; Selecciona Banco 2 y 3
    MOVLW 0x20      ; 1ª dir de los GPRs en el banco
    MOVWF FSR       ; Moverlo al reg, de dir. ind.
Bank2_LP
    CLRF INDF0      ; Limpia el GPR apuntado por FSR
    INCF FSR        ; Inc.puntero
    BTFSS FSR, 7    ; Fin de banco? (FSR = 80h, C = 0)
    GOTO Bank2_LP   ; NO, limpia siguiente
;
; Banco 3
; (**Solo si el micro tiene banco 3 **)
;
    MOVLW 0xA0      ; 1ª dir de los GPRs en el banco
    MOVWF FSR       ; Moverlo al reg, de dir. ind.
Bank3_LP
    CLRF INDF0      ; Limpia el GPR apuntado por FSR
    INCF FSR        ; Inc.puntero
    BTFSS STATUS,C  ; Fin de banco? (FSR = 00h, C = 1)
    GOTO Bank3_LP   ; NO, limpia siguiente
```

REGISTROS DE FUNCIONES ESPECIALES ASOCIADOS AL NÚCLEO DEL MICROCONTROLADOR- REGISTRO STATUS

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

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	TO	PD	Z	DC	C
bit 7							bit 0

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

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
1 = Bank 2, 3 (100h - 1FFh)
0 = Bank 0, 1 (00h - FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
11 = Bank 3 (180h - 1FFh)
10 = Bank 2 (100h - 17Fh)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)
Each bank is 128 bytes

bit 4 **TO**: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 **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/borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)
(for 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/borrow bit (ADDWF, ADDLW ,SUBLW ,SUBWF instructions)
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: For borrow, the polarity is reversed. 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.

REGISTROS DE FUNCIONES ESPECIALES ASOCIADOS AL NUCLEO DEL MICROCONTROLADOR- REGISTRO OPTION

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

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 **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

REGISTROS DE FUNCIONES ESPECIALES ASOCIADOS AL NUCLEO DEL MICROCONTROLADOR- REGISTRO INTCON

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

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	PEIE	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 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

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; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

REGISTROS DE FUNCIONES ESPECIALES ASOCIADOS AL NUCLEO DEL MICROCONTROLADOR- REGISTRO PIE1

REGISTER 2-4: PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE(1):** Parallel Slave Port Read/Write Interrupt Enable bit
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D converter interrupt
0 = Disables the A/D converter interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Synchronous Serial Port Interrupt Enable bit
1 = Enables the SSP interrupt
0 = Disables the SSP interrupt

bit 2 **CCP1IE**: CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Note 1: PSPIE is reserved on PIC16F873/876 devices; always maintain this bit clear.

REGISTROS DE FUNCIONES ESPECIALES ASOCIADOS AL NUCLEO DEL MICROCONTROLADOR- REGISTRO PIR1

REGISTER 2-5: PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF(1):** Parallel Slave Port Read/Write Interrupt Flag bit
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer is full
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer is empty
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Synchronous Serial Port (SSP) Interrupt Flag
1 = The SSP interrupt condition has occurred, and must be cleared in software before returning from the Interrupt Service Routine. The conditions that will set this bit are:
• SPI - A transmission/reception has taken place.
• I2C Slave - A transmission/reception has taken place.
• I2C Master
- A transmission/reception has taken place.
- The initiated START condition was completed by the SSP module.
- The initiated STOP condition was completed by the SSP module.
- The initiated Restart condition was completed by the SSP module.
- The initiated Acknowledge condition was completed by the SSP module.
- A START condition occurred while the SSP module was idle (Multi-Master system).
- A STOP condition occurred while the SSP module was idle (Multi-Master system).
0 = No SSP interrupt condition has occurred.
bit 2 **CCP1IF**: CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode
bit 1 **TMR2IF**: TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred
bit 0 **TMR1IF**: TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow
Note 1: PSPIF is reserved on PIC16F873/876 devices; always maintain this bit clear.

REGISTROS DE FUNCIONES ESPECIALES ASOCIADOS AL NUCLEO DEL MICROCONTROLADOR- REGISTRO PIE2

REGISTER 2-6: PIE2 REGISTER (ADDRESS 8Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **CMIE**: Comparator Interrupt Enable bit
1 = Enables the Comparator interrupt
0 = Disable the Comparator interrupt
bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIE**: EEPROM Write Operation Interrupt Enable
1 = Enable EE Write Interrupt
0 = Disable EE Write Interrupt

bit 3 **BCLIE**: Bus Collision Interrupt Enable
1 = Enable Bus Collision Interrupt
0 = Disable Bus Collision Interrupt

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

REGISTROS DE FUNCIONES ESPECIALES ASOCIADOS AL NUCLEO DEL MICROCONTROLADOR- REGISTRO PIR2

REGISTER 2-7: PIR2 REGISTER (ADDRESS 0Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6 **CMIF**: Comparator Interrupt Flag bit
1 = The Comparator input has changed (must be cleared in software)
0 = The Comparator input has not changed

bit 5 **Unimplemented:** Read as '0'

bit 4 **EEIF**: EEPROM Write Operation Interrupt Flag bit
1 = The write operation completed (must be cleared in software)
0 = The write operation is not complete or has not been started

bit 3 **BCLIF**: Bus Collision Interrupt Flag bit
1 = A bus collision has occurred in the SSP, when configured for I2C Master mode
0 = No bus collision has occurred

bit 2-1 **Unimplemented:** Read as '0'

bit 0 **CCP2IF**: CCP2 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused

REGISTROS DE FUNCIONES ESPECIALES ASOCIADOS AL NUCLEO DEL MICROCONTROLADOR- REGISTRO PCON

REGISTER 2-8: PCON REGISTER (ADDRESS 8Eh)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-1
—	—	—	—	—	—	POR	BOR
bit 7							bit 0

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **POR**: Power-on Reset Status bit
1 = No Power-on Reset occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **BOR**: Brown-out Reset Status bit
1 = No Brown-out Reset occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)