2.- Descripción General del PIC16F877

2.1.- La Familia del PIC16F877

El microcontrolador PIC16F877 de Microchip pertenece a una gran familia de microcontroladores de 8 bits (bus de datos) que tienen las siguientes características generales que los distinguen de otras familias:

- Arquitectura Harvard
- Tecnología RISC
- Tecnología CMOS

Estas características se conjugan para lograr un dispositivo altamente eficiente en el uso de la memoria de datos y programa y por lo tanto en la velocidad de ejecución.

Microchip ha dividido sus microcontroladores en tres grandes subfamilias de acuerdo al número de bits de su bus de instrucciones:

Subfamilia	**Bits del bus de instrucciones**	**nomenclatura**
Base - Line	12	PIC12XXX y PIC14XXX
Mid – Range	14	PIC16XXX
High - End	16	PIC17XXX y PIC18XXX

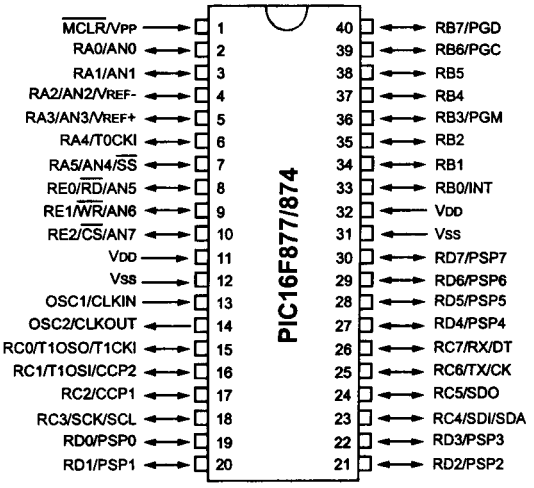
Variantes principales

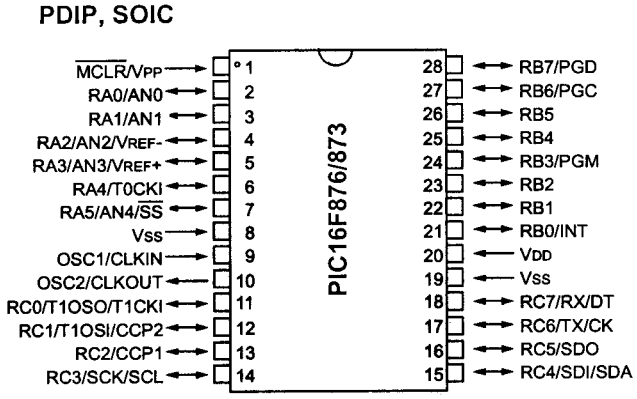
Los microcontroladores que produce Microchip cubren una amplio rango de dispositivos cuyas características pueden variar como sigue:

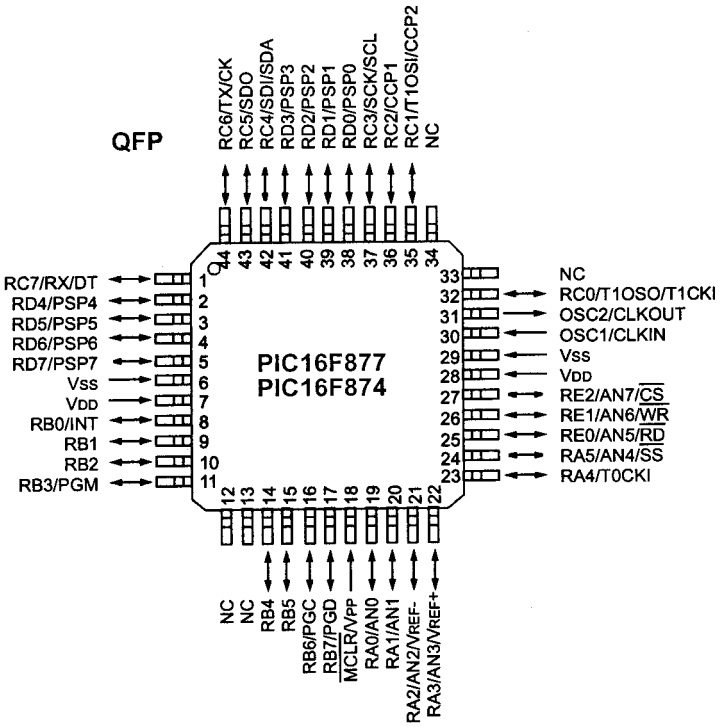
- Empaquetado (desde 8 patitas hasta 68 patitas)
- Tecnología de la memoria incluída (EPROM, ROM, Flash)
- Voltajes de operación (desde 2.5 v. Hasta 6v)
- Frecuencia de operación (Hasta 20 Mhz)

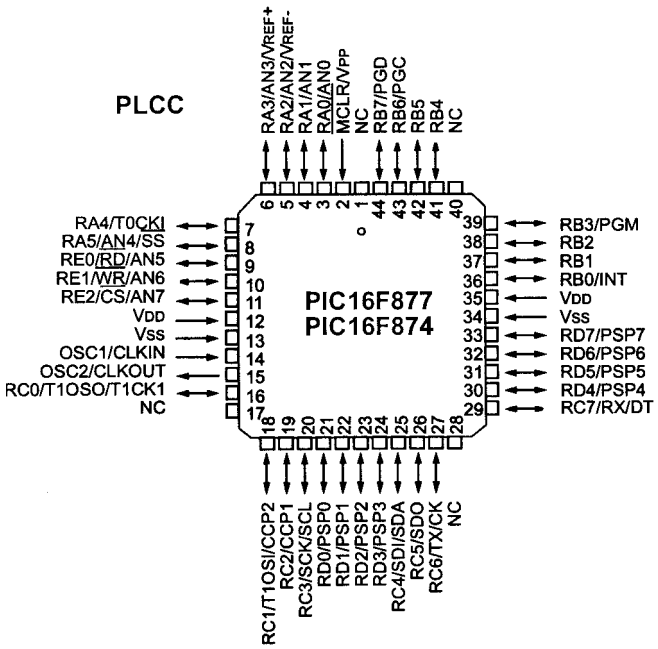
Empaquetados

Aunque cada empaquetado tiene variantes, especilmente en lo relativo a las dimensiones del espesor del paquete, en general se pueden econtrar paquetes tipo PDIP (Plastic Dual In Line Package), PLCC (Plastic Leaded Chip Carrier) y QFP (Quad Flat Package), los cuales se muestran en las figuras siguientes









Nomenclatura

Además de lo mostrado en la tabla anterior, en el nombre específico del microcontrolador pueden aparecer algunas siglas como se muestra en la siguiente tabla:

Tipo de memoria	**Rango de voltaje**	
	Estándar	Extendido
EPROM	PIC16**C**XXX	PIC16**LC**XXX
ROM	PIC16**CR**XXX	PIC16**LCR**XXX
Flash	PIC16**F**XXX	PIC16**LF**XXX

En la siguiente tabla se especifican los rangos de voltaje estándar y extendido manejados por los dispositivos

Rango de voltaje	**EPROM**		**ROM**		**Flash**	
Estándar	**C**	4.5 a 6v	**CR**	4.5 a 6v	**F**	4.5 a 6v
Extendido	**LC**	2.5 a 6v	**LCR**	2.5 a 6v	**LF**	2 a 6v

Oscilador

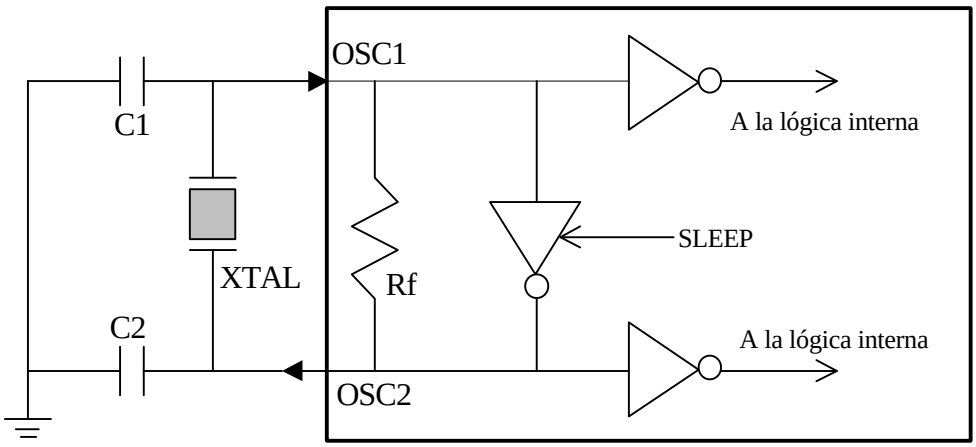
Los PIC de rango medio permiten hasta 8 diferentes modos para el oscilador. El usuario puede seleccionar alguno de estos 8 modos programando 3 bits de configuración del dispositivo denominados: FOSC2, FOSC1 y FOSC0. En algunos de estos modos el usuario puede indicar que se genere o no una salida del oscilador (CLKOUT) a través de una patita de Entrada/Salida. Los modos de operación se muestran en la siguiente lista:

- LP Baja frecuencia (y bajo consumo de potencia)
- XT Cristal / Resonador cerámico externos, (Media frecuencia)
- HS Alta velocidad (y alta potencia) Cristal/resonador
- RC Resistencia / capacitor externos (mismo que EXTRC con CLKOUT)
- EXTRC Resistencia / capacitor externos
- EXTRC Resistencia / Capacitor externos con CLCKOUT
- INTRC Resistencia / Capacitor internos para 4 MHz
- INTRC Resistencia / Capacitor internos para 4 MHz con CLKOUT

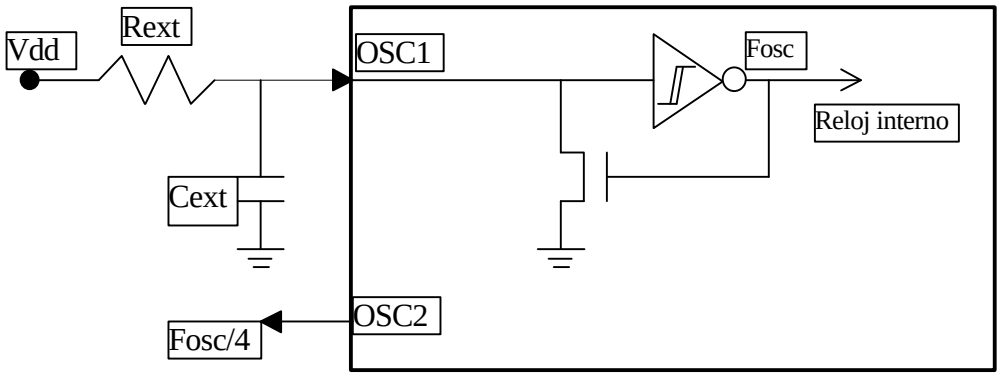
Los tres modos LP, XT y HS usan un cristal o resonador externo, la diferencia sin embargo es la ganancia de los drivers internos, lo cual se ve reflejado en el rango de frecuencia admitido y la potencia consumida. En la siguiente tabla se muestran los rangos de frecuencia así como los capacitores recomendados para un oscilador en base a cristal.

Modo	Frecuencia típica	Capacitores recomendados	
		C1	C2
LP	32 khz 200 khz	68 a 100 pf 15 a 30 pf	68 a 100 pf 15 a 30 pf
XT	100 khz 2 Mhz 4 Mhz	68 a 150 pf 15 a 30 pf 15 a 30 pf	150 a 200 pf 15 a 30 pf 15 a 30 pf
HS	8 Mhz 10 Mhz 20 Mhz	15 a 30 pf 15 a 30 pf 15 a 30 pf	15 a 30 pf 15 a 30 pf 15 a 30 pf

Cristal externo: En los tres modos mostrados en la tabla anterior se puede usar un cristal o resonador cerámico externo. En la siguiente figura se muestra la conexión de un cristal a las patitas OSC1 y OS2 del PIC.



Circuito RC externo: En los modos RC y EXTRC el PIC puede generar su señal oscilatoria basado en un arreglo RC externo conectado a la patita OSC1 como se muestra en la siguiente figura:

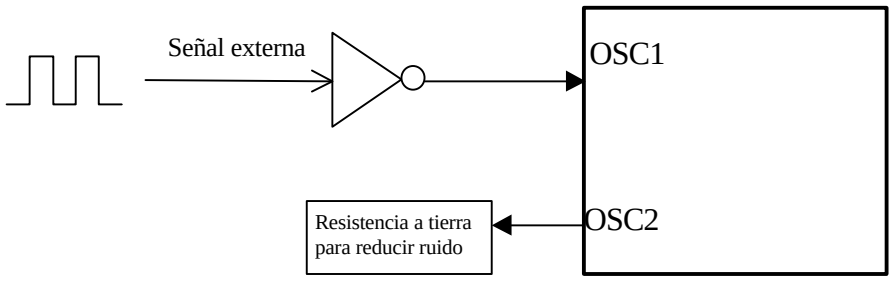


Este modo sólo se recomienda cuando la aplicación no requiera una gran precisión en la medición de tiempos.

Rangos.- La frecuencia de oscilación depende no sólo de los valores de Rext y Cext, sino también del voltaje de la fuente Vdd. Los rangos admisibles para resistencia y capacitor son:

Rext: de 3 a 100 Kohms
Cext: mayor de 20 pf

Oscilador externo.- También es posible conectar una señal de reloj generada mediante un oscilador externo a la patita OSC1 del PIC. Para ello el PIC deberá estar en uno de los tres modos que admiten cristal (LP, XT o HS). La conexión se muestra en la siguiente figura:



Oscilador interno de 4Mhz.- En el modo INTRC el PIC usa un arreglo RC interno que genera una frecuencia de 4 Mhz con un rango de error calibrable de ± 1.5%. Para calibrar el error de oscilación se usan los bits CAL3, CAL2 , CAL1 Y CAL0 del registro OSCCAL.

Calibración del oscilador interno.- El fabricante ha colocado un valor de calibración para estos bits en la última dirección de la memoria de programa. Este dato ha sido guardado en la forma de una instrucción RETLW XX. Si no se quiere perder este valor al borrar el PIC (en versiones EPROM con ventana) primero se deberá leer y copiar. Es una buena idea escribirlo en el empaquetado antes de borrar la memoria).

2.2.- Características generales del PIC16F877

La siguiente es una lista de las características que comparte el PIC16F877 con los dispositivos más cercanos de su familia:

PIC16F873	PIC16F874	PIC16F876	PIC16F877

- CPU RISC
- Sólo 35 instrucciones que aprender
- Todas las instrucciones se ejecutan en un ciclo de reloj, excepto los saltos que requieren dos
- Frecuencia de operación de 0 a 20 MHz (DC a 200 nseg de ciclo de instrucción)
- Hasta 8k x 14 bits de memoria Flash de programa
- Hasta 368 bytes de memoria de datos (RAM)
- Hasta 256 bytes de memoria de datos EEPROM
- Hasta 4 fuentes de interrupción
- Stack de hardware de 8 niveles
- Reset de encendido (POR)
- Timer de encendido (PWRT)
- Timer de arranque del oscilador (OST)
- Sistema de vigilancia Watchdog timer.
- Protección programable de código
- Modo SEP de bajo consumo de energía
- Opciones de selección del oscilador
- Programación y depuración serie “In-Circuit” (ICSP) a través de dos patitas
- Lectura/escritura de la CPU a la memoria flash de programa
- Rango de voltaje de operación de 2.0 a 5.5 volts
- Alta disipación de corriente de la fuente: 25mA

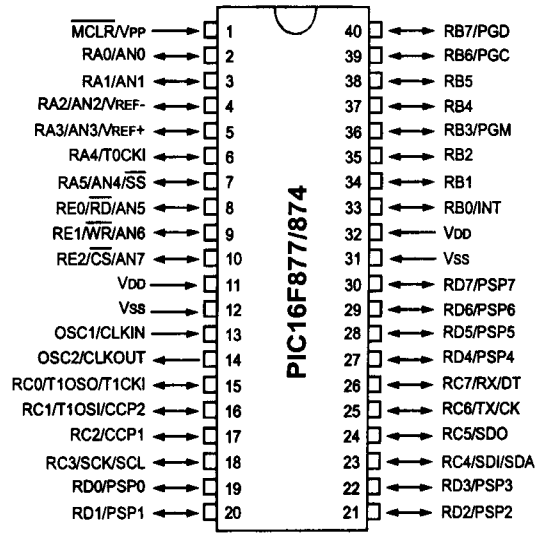
- Rangos de temperatura: Comercial, Industrial y Extendido
- Bajo consumo de potencia:
  - Menos de 0.6mA a 3V, 4 Mhz
  - 20 µA a 3V, 32 Khz
  - menos de 1µA corriente de standby.

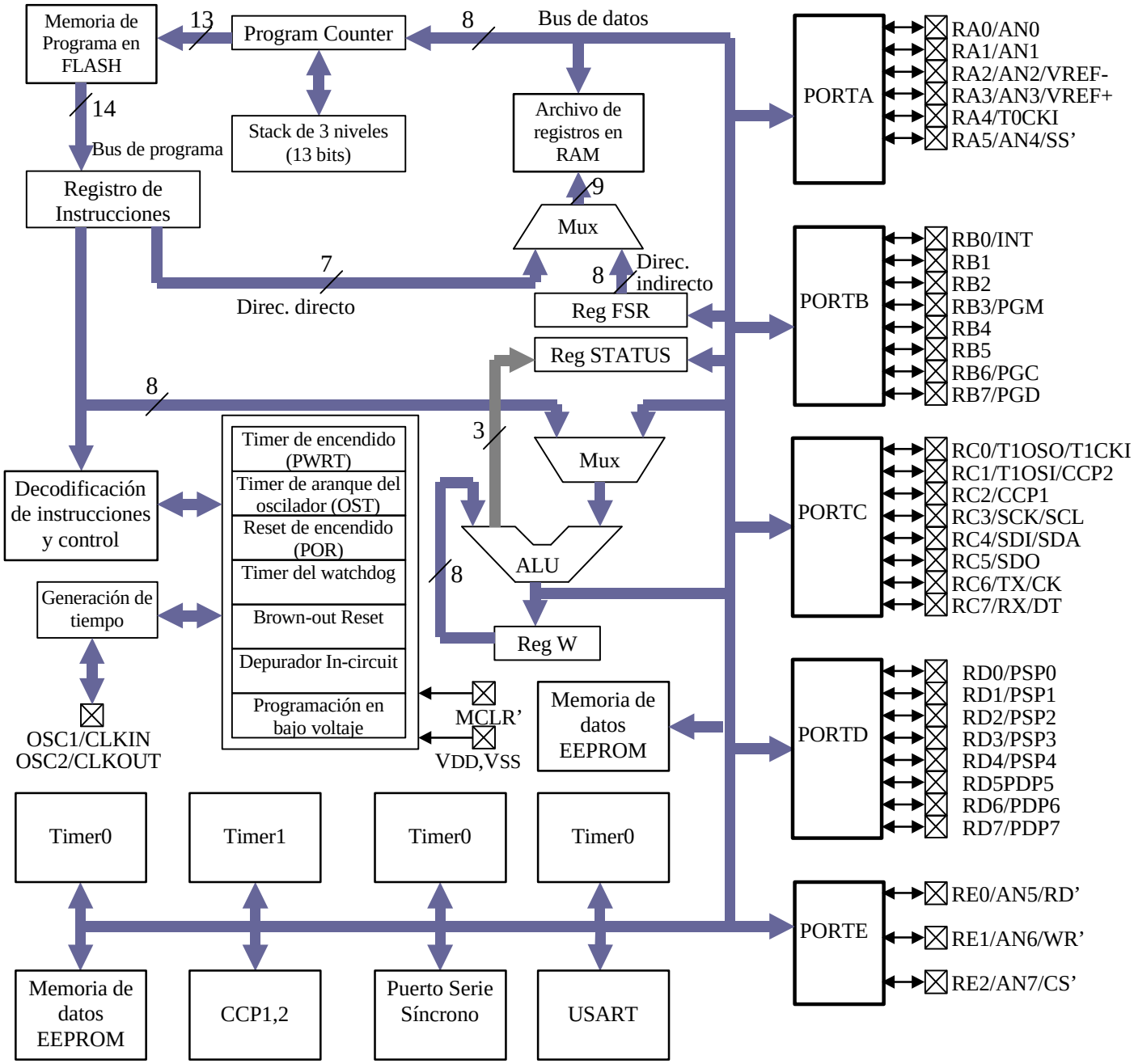
Periféricos

- **Timer0**: Contador/Temporizador de 8 bits con pre-escalador de 8 bits
- **Timer1**: Contador/Temporizador de 16 bits con pre-escalador
- **Timer0**: Contador/Temporizador de 8 bits con pre-escalador y post-escalador de 8 bits y registro de periodo.
- **Dos módulos de Captura, Comparación y PWM**
- **Convertidor Analógico/Digita**l: de 10 bits, hasta 8 canales
- Puerto Serie Síncrono (SSP)
- Puerto Serie Universal (USART/SCI).
- Puerto Paralelo Esclavo (PSP): de 8 bits con líneas de protocolo

2.3.- Diagrama de Bloques del PIC16F877

En la siguiente figura se muestra a manera de bloques la organización interna del PIC16F877, Se muestra también junto a este diagrama su diagrama de patitas, para tener una visión conjunta del interior y exterior del Chip.



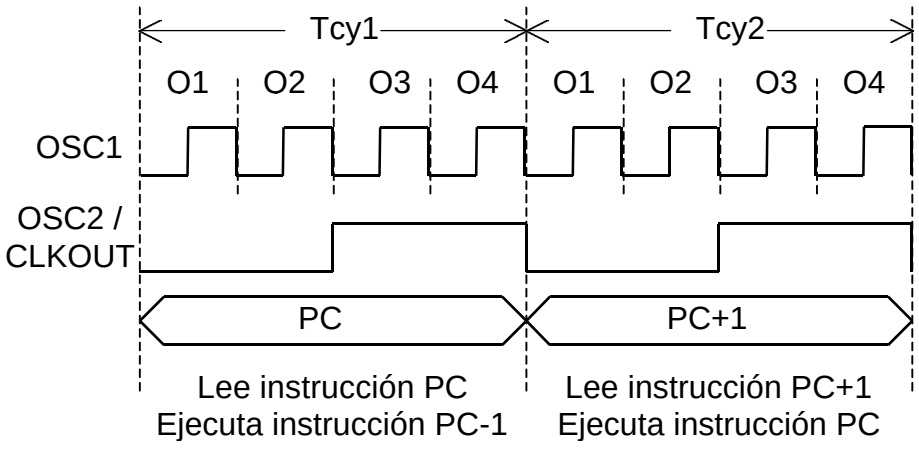

Memoria de Programa en FLASH
13
Program Counter
8
Bus de datos
14
Bus de programa
Stack de 3 niveles (13 bits)
Archivo de registros en RAM
Registro de Instrucciones
9
Mux
7
Direc. directo
8
Direc. indirecto
Reg FSR
Reg STATUS
8
3
Mux
Timer de encendido (PWRT)
Timer de aranque del oscilador (OST)
Reset de encendido (POR)
Timer del watchdog
Brown-out Reset
Depurador In-circuit
Programación en bajo voltaje
Decodificación de instrucciones y control
Generación de tiempo
ALU
8
Reg W
MCLR'
VDD,VSS
OSC1/CLKIN
OSC2/CLKOUT
Memoria de datos EEPROM
PORTA
RA0/AN0
RA1/AN1
RA2/AN2/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/AN4/SS'
PORTB
RB0/INT
RB1
RB2
RB3/PGM
RB4
RB5
RB6/PGC
RB7/PGD
PORTC
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2
RC2/CCP1
RC3/SCK/SCL
RC4/SDI/SDA
RC5/SDO
RC6/TX/CK
RC7/RX/DT
PORTD
RD0/PSP0
RD1/PSP1
RD2/PSP2
RD3/PSP3
RD4/PSP4
RD5PDP5
RD6/PDP6
RD7/PDP7
PORTE
RE0/AN5/RD'
RE1/AN6/WR'
RE2/AN7/CS'
Timer0
Timer1
Timer0
Timer0
Memoria de datos EEPROM
CCP1,2
Puerto Serie Síncrono
USART

2.4.- Descripción de la CPU

La CPU es la responsable de la interpretación y ejecución de la información (instrucciones) guardada en la memoria de programa. Muchas de estas instrucciones operan sobre la memoria de datos. Para operar sobre la memoria de datos además, si se van a realizar operaciones lógicas o aritméticas, requieren usar la Unidad de Lógica y Aritmética (ALU). La ALU controla los bits de estado (Registro STATUS), los bits de este registro se alteran dependiendo del resultado de algunas instrucciones.

Ciclo de instrucción

El registro Program Counter (PC) es gobernado por el ciclo de instrucción como se muestra en la siguiente figura. Cada ciclo de instrucción la CPU lee (ciclo Fetch) la instrucción guardada en la memoria de programa apuntada por PC y al mismo tiempo ejecuta la instrucción anterior, esto debido a una ***cola de instrucciones*** que le permite ejecutar una instrucción mientras lee la próxima:



Como puede verse, cada ciclo de instrucción (Tcy) se compone a su vez de cuatro ciclos del oscilador (Tosc). Cada ciclo Q provee la sincronización para los siguientes eventos:

Q1: Decodificación de la instrucción
Q2: Lectura del dato (si lo hay)
Q3: Procesa el dato
Q4: Escribe el dato

Debido a esto cada ciclo de instrucción consume 4 ciclos de reloj, de manera que si la frecuencia de oscilación es Fosc, Tcy será 4/Fosc.

Registros de la CPU.

Registro PC.- Registro de 13 bits que siempre apunta a la siguiente instrucción a ejecutarse. En la siguiente sección se dan mayores detalles en el manejo de este registro.

Registro de Instrucción.- Registro de 14 bits. Todas las instrucciones se colocan en el para ser decodificadas por la CPU antes de ejecutarlas.

Registro W.- Registro de 8 bits que guarda resultados temporales de las operaciones realizadas por la ALU

Registro STATUS.- Registro de 8 bits, cada uno de sus bits (denominados ***Banderas***) es un indicador de estado de la CPU o del resultado de la última operación como se indica en la siguiente figura:

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	6	5	4	3	2	1	Bit 0

- **Notación**: En adelante se usará lo siguiente:

R= Bit leíble W= Bit Escribible U= No implementado (se lee como 0)
-n= Valor después del Reset de encendido

Z.- Este bit se ***pone*** (=1) para indicar que el resultado de la última operación fue cero, de lo contrario se ***limpia*** (=0)

C.- Bit de acarreo/préstamo’ de la última operación aritmética (en el caso de resta, se guarda el préstamo invertido

CD.- Acarreo/Préstamo’ proveniente del cuarto bit menos significativo. Funciona igual que el bit C, pero para operaciones de 4 bits.

2.5.- Conjunto de Instrucciones de Rango Medio

En la siguiente tabla se resumen las 35 instrucciones que reconoce la CPU de los PIC de medio rango, incluyendo su mnemónico, tiempo de ejecución, código de máquina y afectación de banderas:

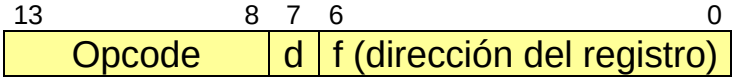
Mnemónico	Descripción	Ciclos	Código de Máquina	Banderas afectadas
Operaciones con el archivo de registros orientadas a bytes				
ADDWF f,d	Suma f + W	1	**00 0111 dfff ffff**	C,DC,Z
ANDWF f,d	W AND f	1	**00 0101 dfff ffff**	Z
CLRF f	Limpia f	1	**00 0001 1fff ffff**	Z
CLRW	Limpia W	1	**00 0001 0xxx xxxx**	Z
COMF f,d	Complementa los bits de f	1	**00 1001 dfff ffff**	Z
DECF f,d	Decrementa f en 1	1	**00 0011 dfff ffff**	Z
DECFSZ f,d	Decrementa f, escapa si 0	1(2)	**00 1011 dfff ffff**	
INCF f,d	Incrementa f en 1	1	**00 1010 dfff ffff**	Z
INCFSZ f,d	Incrementa f, escapa si 0	1(2)	**00 1111 dfff ffff**	
IORWF f,d	W OR f	1	**00 0100 dfff ffff**	Z
MOVF f,d	Copia el contenido de f	1	**00 1000 dfff ffff**	Z
MOVWF f	Copia contenido de W en f	1	**00 0000 1fff ffff**	
NOP	No operación	1	**00 0000 0xx0 0000**	
RLF f,d	Rota f a la izquierda	1	**00 1101 dfff ffff**	C
RRF f,d	Rota f a la derecha	1	**00 1100 dfff ffff**	C
SUBWF f,d	Resta f – W	1	**00 0010 dfff ffff**	C,DC,Z
SWAPF f,d	Intercambia nibbles de f	1	**00 1110 dfff ffff**	
XORWF f,d	W EXOR f	1	**00 0110 dfff ffff**	Z
Operaciones con el archivo de registros orientadas a bits				
BCF f,b	Limpia bit b en f	1	**01 00bb bfff ffff**	
BSF f,b	Pone bit b en f	1	**01 01bb bfff ffff**	
BTFSC f,b	Prueba bit b en f, escapa si 0	1(2)	**01 10bb bfff ffff**	
BTFSS f,b	Prueba bit b en f, escapa si 1	1(2)	**01 11bb bfff ffff**	
Operaciones con literales y de control				
ADDLW k	Suma literal k + W	1	**11 111x kkkk kkkk**	C,DC,Z
ANDLW k	k AND W	1	**11 1001 kkkk kkkk**	Z
CALL k	Llamado a subrutina	2	**10 0kkk kkkk kkkk**	
CLRWDT	Limpia timer del watchdog	1	**00 0000 0110 0100**	TO',PD'
GOTO k	Salto a la dirección k	2	**10 1kkk kkkk kkkk**	
IORLW k	k OR W	1	**11 0000 kkkk kkkk**	Z
MOVLW k	Copia literal a W	1	**11 00xx kkkk kkkk**	
RETFIE	Retorna de interrupción	2	**00 0000 0000 1001**	
RETLW k	Retorna con literal k en W	2	**11 01xx kkkk kkkk**	
RETURN	Retorna de subrutina	2	**00 0000 0000 1000**	
SLEEP	Activa Modo standby	1	**00 0000 0110 0011**	TO'PD'
SUBLW k	Resta k - W	1	**11 110x kkkk kkkk**	C,CD,Z
XORLW k	k EXOR W	1	**11 1010 kkkk kkkk**	Z

Formato General de las Instrucciones.

Cada instrucción en lenguaje de máquina (binario) del PIC contiene un código de operación (***opcode***) el cual puede ser de 3 a 4 o 6 bits, dependiendo del tipo de instrucción.

A continuación se describe el formato para cada tipo de instrucción de los PIC de rango medio:

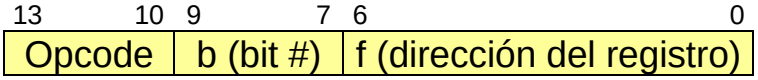
Operaciones con el archivo de registros orientadas a bytes



El bit **d** especifica el destino del resultado de la operación:

d = 0: destino W
d = 1: destino f
f = dirección de 7 bits del archivo de registros.

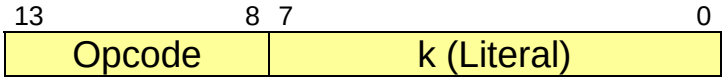
Operaciones con el archivo de registros orientadas a bits



b : Especificación en tres bits del bit sobre el que se va a operar
f = dirección de 7 bits del archivo de registros.

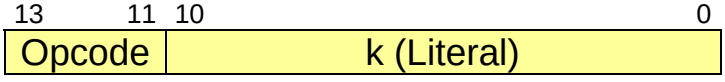
Operaciones con literales y de control

Formato general:



k : Literal = Valor de un operando de 8 bits

Formato para CALL y GOTO:



k : Literal = Valor de un operando de 8 bits

2.6.- Organización de la memoria del PIC

Los PIC tienen dos tipos de memoria: Memoria de Datos y Memoria de programa, cada bloque con su propio bus: Bus de datos y Bus de programa; por lo cual cada bloque puede ser accesado durante un mismo ciclo de oscilación.

La Memoria de datos a su vez se divide en

- Memoria RAM de propósito general
- Archivo de Registros (Special Function Registers (SFR))

2.6.1.- La Memoria de Programa

Los PIC de rango medio poseen un registro Contador del Programa (PC) de 13 bits, capaz de direccionar un espacio de 8K x 14, como todas la instrucciones son de 14 bits, esto significa un bloque de 8k instrucciones. El bloque total de 8K x 14 de memoria de programa está subdividido en 4 páginas de 2K x 14. En la siguiente figura se muestra esta organización.

Dirección	
0000h	Vector de Reset
. . .	. . .
0004h	Vector de interrupción
0005h . . . 07FFh	Página 0
0800h . . . 0FFFh	Página 1
1000h . . . 17FFh	Página 2
1800h . . . 1FFFh	Página 3

Observación1: No todos los PIC tienen implementado todo el espacio de 8K de memoria de programa (Consultar las hojas de datos del PIC específico).

Observación2: El fabricante puede grabar datos de calibración en localidades de memoria de programa por lo que se deberán anotar en papel antes de borrar los dispositivos con ventana transparente.

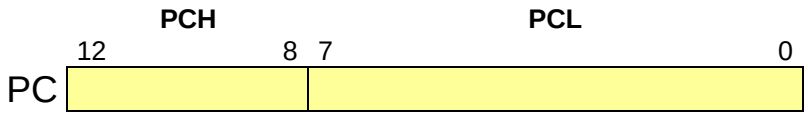
Vector de Reset.- Cuando ocurre un reset el contenido del PC es forzado a cero, ésta es la dirección donde la ejecución del programa continuará después del reset, por ello se le llama "**dirección del vector de reset**".

Vector de interrupción.- Cuando la CPU acepta una solicitud de interrupción ejecuta un salto a la dirección 0004h, por lo cual a esta se le conoce como "**dirección del vector de interrupción**". El registro PCLATH no es modificado en esta circunstancia, por lo cual habrá que tener cuidado al manipular el PC dentro de la Rutina de Atención a la Interrupción (Interrupt Service Routine (ISR)).

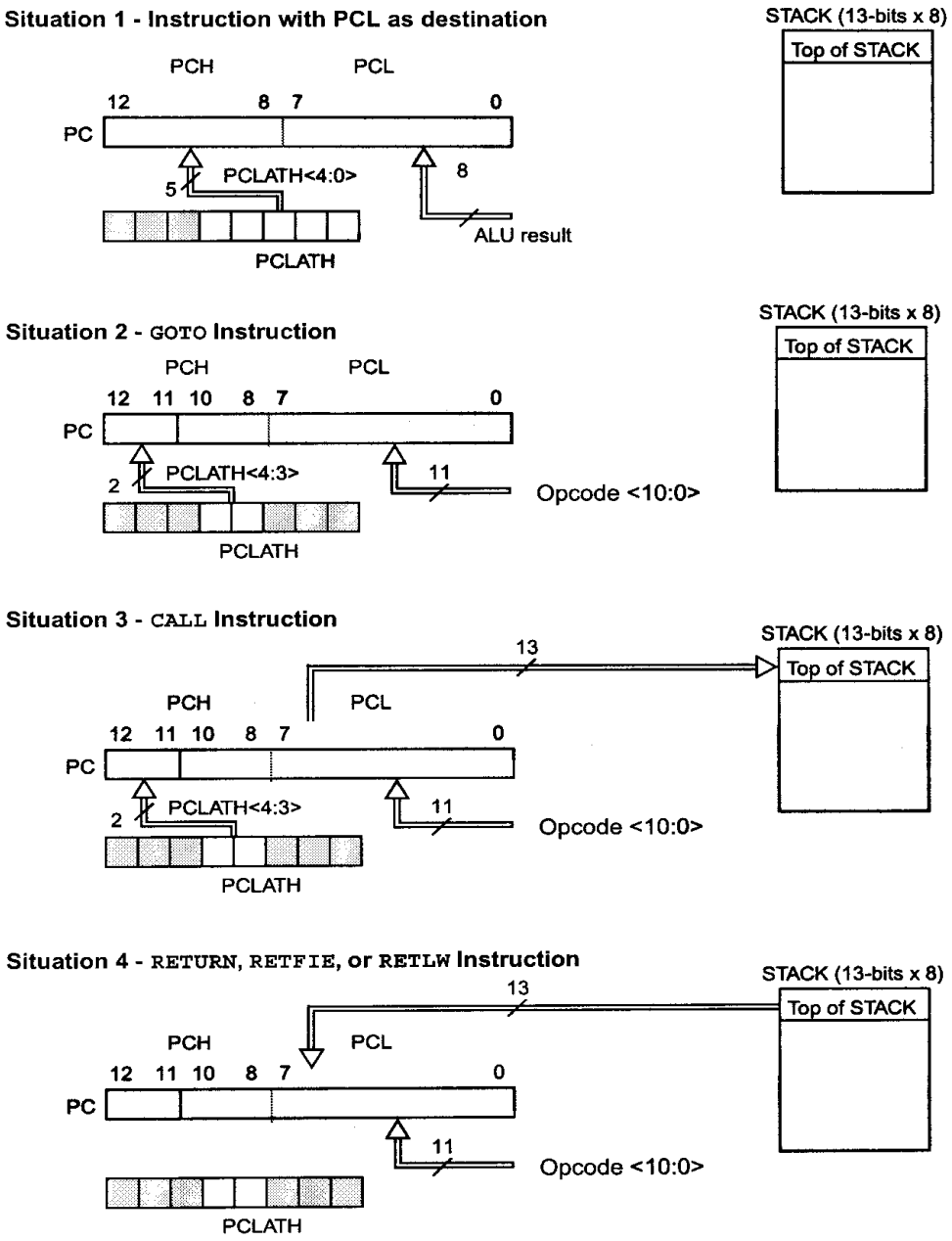
Manejo del Contador del Programa (PC)

El registro contador del programa (PC) especifica la dirección de la instrucción que la CPU buscará (fetch) para ejecutarla.

El PC consta de 13 bits, separados en dos partes: como se muestra en la figura siguiente



El byte de orden bajo es llamado el registro PCL, mientras que el byte de orden alto es llamado registro PCH. Este último contiene los bits PC<12:8> y ***no se puede leer o escribir directamente*** Todas las actualizaciones al registro PCH deben ser hechas a través del registro PCLATH. En la siguiente figura se ilustran las cuatro situaciones y las maneras correspondientes en que el PC puede ser actualizado.



- **Nota**: PCLATH nunca es actualizado con el contenido de PCH

Paginación

Para saltar entre una página y otra, los bits más significativos del PC deberán ser modificados. Debido a que las instrucciones GOTO y CALL sólo pueden direccionar un bloque de 2K (pues usan una dirección de 11 bits) deben existir otros dos bits que acompleten los 13 bits del PC para moverse sobre los 8K de memoria de programa.

Estos dos bits extra se encuentran en un SFR denominado PCLATH (Program Counter Latch High) en sus bits PCLATH<4:3>. Por esto antes de un GOTO o un CALL el usuario deberá asegurarse que estos bits apunten a la página deseada.

Si las instrucciones se ejecutan secuencialmente el PC cruza libremente los límites de página sin necesidad de que el usuario escriba en el PCLATH

Memoria de Stack

La memoria de stack es una area de memoria completamente separada de la memoria de datos y la memoria de programa. El stack consta de 8 niveles de 13 bits cada uno. Esta memoria es usada por la CPU para almacenar las direcciones de retorno de subrutinas. El apuntador de stack no es ni legible ni escribible.

Cuando se ejecuta una instrucción CALL o es reconocida una interrupción el PC es guardado en el stack y el apuntador de stack es incrementado en 1 para apuntar a la siguiente posición vacía. A la inversa, cuando se ejecuta una instrucción RETURN, RETLW o RETFIE el contenido de la posición actual del stack es colocado en el PC.

- **Nota 1**: PCLATH no se modifica en ninguna de estas operaciones

- **Nota 2**: Cuando el apuntador de stack ya está en la posición 8 y se ejecuta otro CALL se reinicia a la posición 1 sobrescribiendo en dicha posición. No existe ningún indicador que avise de esta situación. Así que el usuario deberá llevar el control para que esto no ocurra.

2.6.2.- La Memoria de Datos

La memoria de datos consta de dos áreas mezcladas y destinadas a funciones distintas:

- Registros de Propósito Especial (SFR)

- Registro de Propósito General (GPR)

Los SFR son localidades asociadas específicamente a los diferentes periféricos y funciones de configuración del PIC y tienen un nombre específico asociado con su función. Mientras que los GPR son memoria RAM de uso general.

Bancos de memoria

Toda la memoria de datos está organizada en 4 ***bancos*** numerados 0, 1, 2 y 3. Para seleccionar un banco se debe hacer uso de los bits del registro STATUS<7:5> denominados IRP, RP1 y RP0.

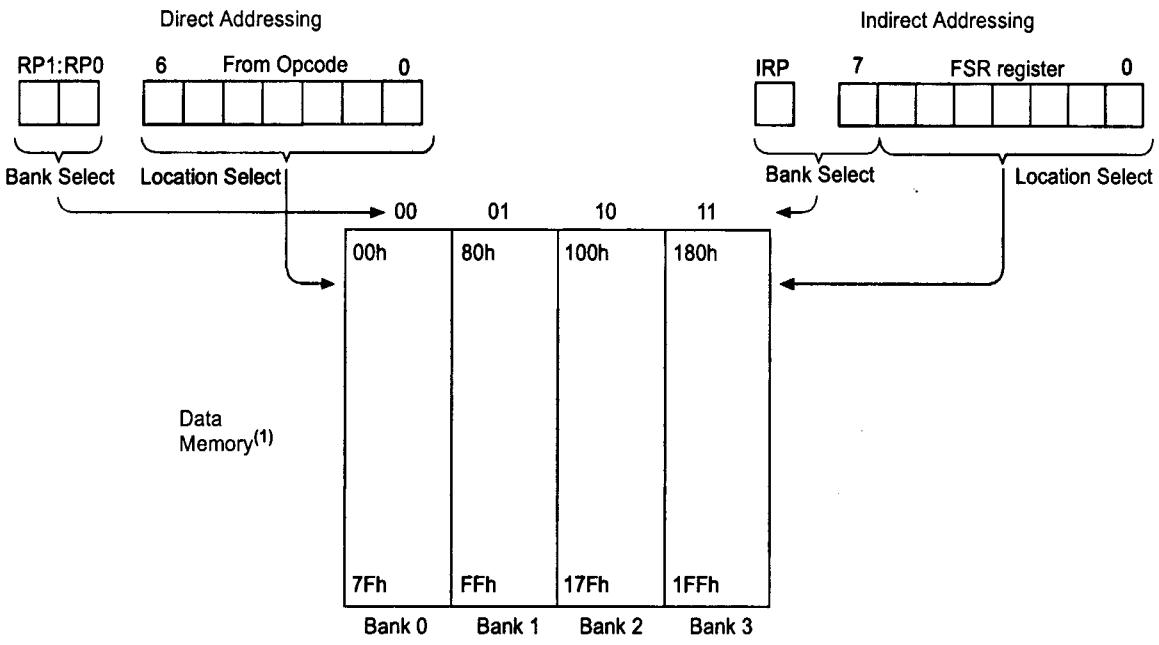
Hay dos maneras de acceder a la memoria de datos: ***Direccionamiento directo e indirecto***. La selección de bancos se basa en la siguiente tabla

Direccionamiento Indirecto (IRP)	RP1:RP0	Banco
0	0 0	0
	0 1	1
1	1 0	2
	1 1	3

Cada banco consta de 128 bytes (de 00h a 7Fh). En las posiciones más bajas de cada banco se encuentran los SFR, y arriba de éstos se encuentran los GPR. Toda la memoria de datos está implementada en Ram estática.

Direccionamiento Directo

Para acceder una posición de memoria mediante direccionamiento directo, la CPU simplemente usa la dirección indicada en los 7 bits menos significativos del código de operación y la selección de banco de los bits RP1:RP0 como se ilustra en la siguiente figura.



Direccionamiento indirecto

Este modo de direccionamiento permite acceder una localidad de memoria de datos usando una dirección de memoria *variable* a diferencia del direccionamiento directo, en que la dirección es fija. Esto puede ser útil para el manejo de tablas de datos.

El registro INDF.- En la figura anterior se muestra la manera en que esto se realiza. Para hacer posible el direccionamiento indirecto se debe usar el registro INDF. Cualquier instrucción que haga un acceso al registro INDF en realidad accesa a la dirección apuntada por el registro FSR (File Select Register).

La selección de banco en el caso de direccionamiento indirecto se realiza mediante los bits IRP (STATUS<7>) y el bit 7 del registro FSR, como se muestra en la figura.

El registro INDF mismo al leerse de manera indirecta (con FSR=0) producirá un cero. Y al escribirse de manera indirecta no es afectado.

A continuación se muestra un ejemplo del uso de este direccionamiento para limpiar las localidades RAM 20h a 2Fh.

Ejemplo 1.- Blanqueo de un bloque de memoria de datos desde la localidad 20h a la localidad 2Fh

```
        MOVLW 0X20   ;carga valor de apuntador a RAM
        MOVWF FSR    ;inicializa apuntador
Sigue   CLRF INDF    ;limpia localidad apuntada por FSR
        INCF FSR,F   ;incrementa apuntador
        BTFSS FSR,4  ;si ya terminó escapa a continuar
        GOTO sigue   ;si no repite
Continuar    ...
```

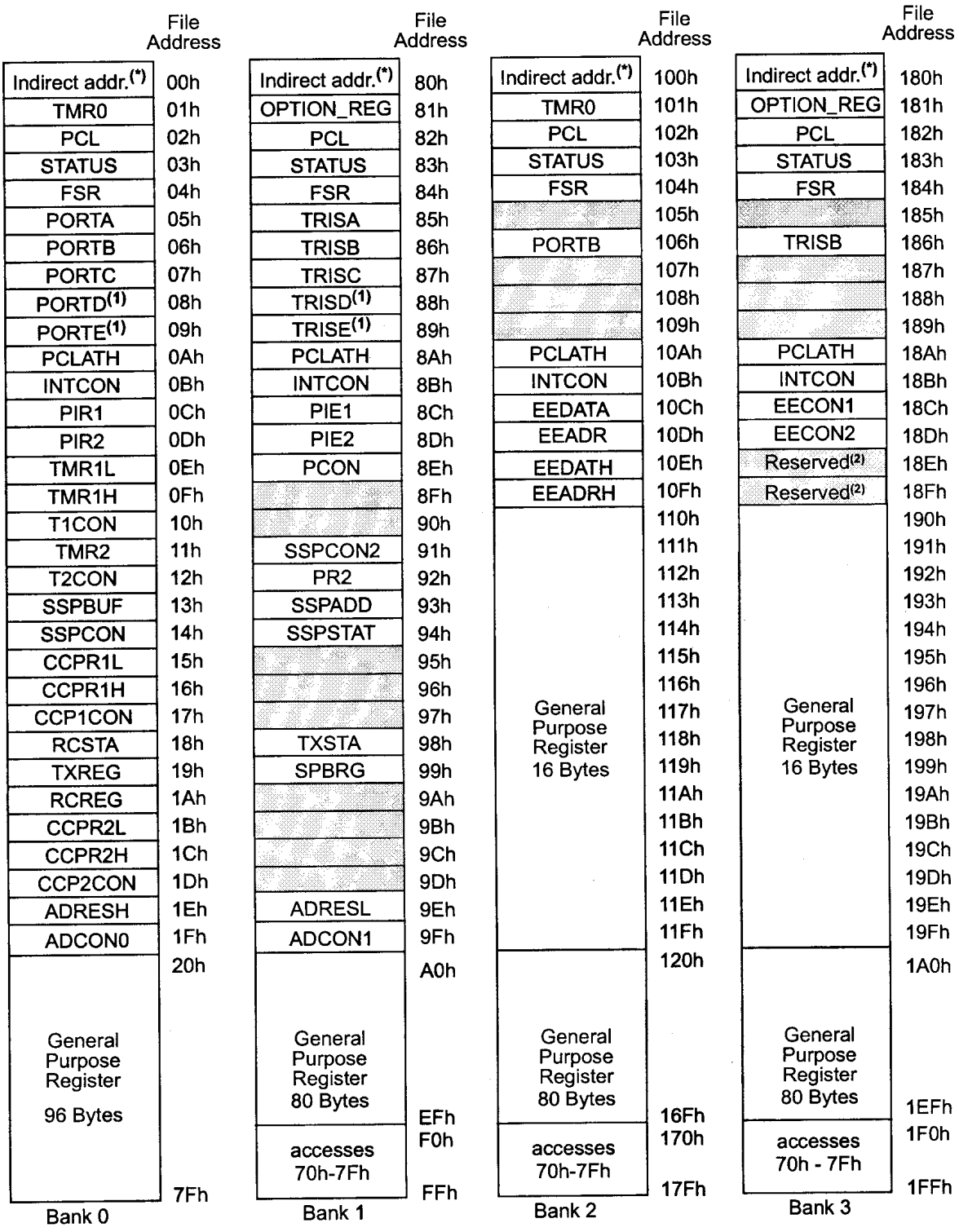
En el siguiente ejemplo se muestra la manera como se switchea mediante instrucciones dentro del programa de un banco a otro

Ejemplo 2.- Switcheo entre bancos de memoria RAM

```
        CLRF STATUS      ;Limpia registro STATUS
                         ;(Banco 0)
        ...
        BSF STATUS,RP0   ;Banco 1
        …
        BSF STATUS RP1   ;Banco 3
        …
        BCF STATUS RP0   ;Banco 2
        …
```

El Archivo de Registros

Aunque el archivo de registros en RAM puede variar de un PIC a otro, la familia del PIC16F87x coincide casi en su totalidad. En la siguiente figura se muestra a detalle el mapa de este archivo de registros y su organización en los cuatro bancos que ya se describieron.



Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
2: These registers are reserved, maintain these registers clear.