1.- Introducción a los Microcontroladores.

1.1.- Introducción.

El microcontrolador nace cuando las técnicas de integración han progresado lo bastante para permitir su fabricación; pero también porque, muy a menudo, tanto en las aplicaciones domésticas como industriales, se tiene la necesidad de sistemas "inteligentes" o, al menos programables. Un ejemplo muy simple es el programador de una lavadora, el cual debe controlar una cierta cantidad de elementos con ciclos y cadencias perfectamente definidas, pero variables en función del programa seleccionado. Otras aplicaciones más técnicas tienen, igualmente, necesidad de sistemas programables. Por ejemplo, una fotocopiadora debe controlar permanentemente un gran número de elementos y de funciones. Gracias a la llegada de los microcontroladores, tarjetas que contenían varias decenas de circuitos lógicos clásicos se han visto reducidas a dos o tres microcontroladores.

Antes de ver qué es un microcontrolador y de analizar sus puntos fuertes y sus defectos, será útil hacer un repaso relativo a la estructura de cualquier sistema programable, que pueda hacer uso de un microcontrolador.

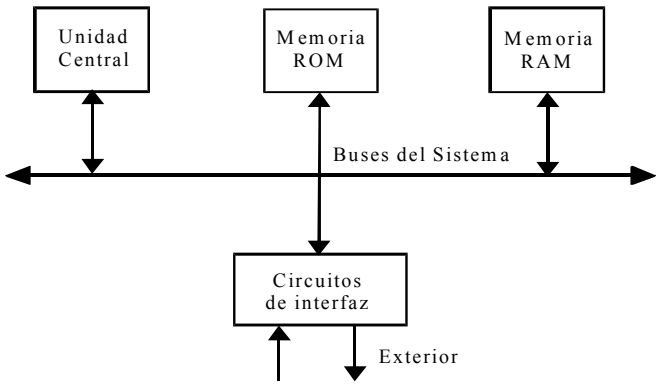

Figura 1.

La figura 1 presenta el esquema tipo de cualquier sistema programable. Veamos que elementos son indispensables para su funcionamiento:

- La unidad central
- La memoria ROM
- La memoria RAM
- Los circuitos de interfaz
- Un bus de interconexión

La presencia de estos elementos básicos es indispensable y aun cuando no siempre se presenten tan claramente como en nuestro esquema siempre existen. Obsérvese, que son los mismos que los de un sistema informático clásico, pero dentro del marco de una aplicación que pueda ser tratada por un microcontrolador.

La unidad central, generalmente constituida por un microprocesador más o menos evolucionado, ejecuta el programa que da vida a la aplicación. Los programas pueden ser muy diversos, puesto que, como es evidente, el que asegura la gestión de un termostato inteligente no tiene nada que ver con el que controla el correcto funcionamiento de una fotocopiadora.

Sin embargo, estos programas tienen en común el hecho de que muy raramente necesitan cálculos complejos y, en cambio, sí suelen incluir numerosas manipulaciones de la información de entrada/salida.

El programa se almacena en un segundo elemento, que es la memoria ROM. Esta memoria puede constituirse de diferentes formas: UVPROM, EEPROM u OTPROM, cualquiera que sea la que se utilice es una memoria no volátil desde la que se ejecutará el programa una vez alimentado el sistema. Para poder trabajar correctamente, nuestro microprocesador necesita, a menudo, almacenar datos temporales en alguna parte, y aquí es donde interviene la memoria RAM, que no necesita ser de grandes dimensiones.

El último elemento y que, generalmente, es el más importante en una aplicación susceptible de utilizar un microcontrolador es todo lo concerniente a los circuitos de interfaz con el mundo exterior, que relacionará al microprocesador con elementos tan dispares como un motor paso a paso, un display de cristal líquido o una botonera hexadecimal.

Después de este pequeño análisis nos podemos preguntar por qué se habla de microcontrolador y, no de un conjunto de elementos que realizan esta función. La respuesta es que el objetivo de los microcontroladores es integrar, tanto como sea posible, en un único encapsulado el conjunto de funciones de la figura 1.

1.2.- Contenido típico de un microcontrolador.

De lo descrito anteriormente, es evidente que un microcontrolador debe contener todos los elementos de la figura 1 en un único encapsulado; aunque no con un diseño tan simple. A la vista de los análisis de los sistemas realizados antes de la aparición de los microcontroladores, los fabricantes de circuitos integrados han perfilado la definición de lo que se debería integrar, para llegar al esquema de la figura 2.

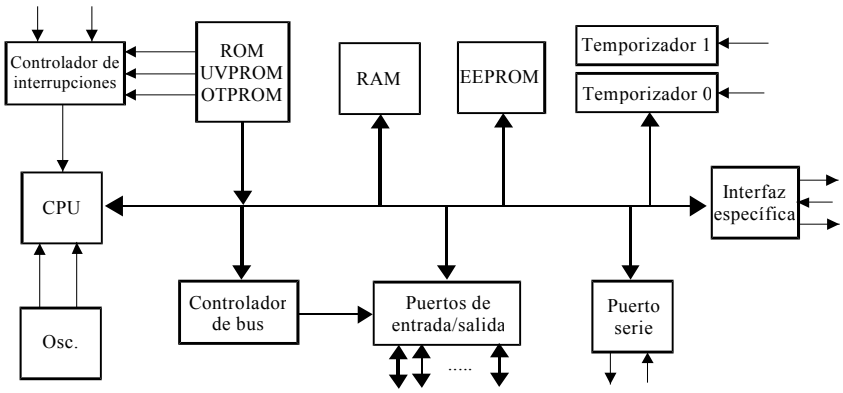

Figura 2

Evidentemente, encontramos en él nuestra unidad central pero, salvo casos particulares, frecuentemente se ha simplificado con respecto a los microprocesadores clásicos. En contrapartida se le han añadido instrucciones de manejo de bits, muy útiles para las entradas/salidas. En ciertos circuitos, esta unidad central está dotada de un gran número de registros internos, que sirven de memoria RAM, por lo que puede parecer que ésta última está ausente de algunos esquemas.

A continuación podemos ver la memoria ROM, aunque ésta no siempre aparece. En determinados encapsulados y hasta hace unos años, esta memoria no podía programarse más que mediante máscara durante la fabricación del circuito. Esto imponía al potencial usuario del microcontrolador comprar un número significativo de piezas idénticas, lo cual era aceptable para una serie grande, pero no para fabricaciones limitadas. Cierto número de microcontroladores estaban, y todavía están, disponibles sin ROM (versiones ROMless en los catálogos).

Posteriormente, los fabricantes han introducido en el chip una memoria programable eléctricamente y borrable mediante rayos ultravioleta (UVPROM) o, más recientemente, borrable eléctricamente (EEPROM). Como los encapsulados que contenían la memoria UVPROM eran relativamente caros (por la ventana de cuarzo), han aparecido otro tipo denominado OTPROM (One Time PROM), la memoria UVPROM existe siempre y se programa como cualquier circuito, pero debido a la ausencia de ventana, no se puede borrar. Es una solución interesante para la producción de series pequeñas y están disponibles en plásticos baratos.

Un último producto para almacenar de forma no volátil son las memorias FLASH, de bajo consumo, que se puede escribir y borrar en circuito al igual que las EEPROM, pero suelen disponer de mayor capacidad que estas últimas. El borrado solo es posible con bloques completos y no se puede realizar sobre posiciones concretas. Son muy recomendables en aplicaciones en las que sea necesario modificar el programa a lo largo de la vida del producto, como consecuencia del desgaste o cambios de piezas, como sucede con los vehículos.

En lo referente a la memoria RAM, suele utilizarse una del tipo SRAM (RAM estática) de pequeño tamaño, por qué generalmente la unidad central posee suficientes registros para realizar operaciones intermedias. En algunos casos, esta memoria se completa con una EEPROM de datos, que memoriza de forma semipermanente datos del usuario que se manejan como constante en la ejecución del programa y que de vez en cuando (pasados meses o años) deben ser modificados.

Algo más delicado es hacer un esquema tipo para los circuitos de interfaz, ya que es un punto donde se distinguen los diferentes microcontroladores, en función de las aplicaciones que pretenden. No obstante se pueden encontrar los siguientes elementos básicos:

- Líneas de entrada/salida paralelo, en cantidad variable, según la finalidad y el tamaño del encapsulado (se plantea un problema de número máximo de pines debido al crecimiento del número de estas líneas).

- Al menos una interfaz de entrada/salida serie asíncrona, más o menos evolucionada según los circuitos.

- Uno o varios temporizadores internos cuyas posibilidades pueden ser muy variables pero que, generalmente, funcionan como contadores ascendentes y descendentes, generadores de impulsos programables, etc.

- Uno o varios convertidores analógicos/digitales, precedidos o no de multiplexores para ofrecer varias vías.

- A veces, pero es raro, un covertidor digital/analógico.

Por último, aunque no sea una verdadera interfaz de entrada/salida en el sentido en que nosotros entendemos, ciertos microcontroladores disponen de un acceso a su bus interno. Esto permite conectar con otros circuitos destinados a cumplir funciones que faltan en el chip, lo que a veces resulta útil. Precisemos, aunque es evidente, que todos los microcontroladores sin memoria ROM interna disponen necesariamente de esta interfaz, puesto que es necesario permitirle acceder a una memoria ROM externa.

1.3.- Las ventajas y defectos de los Microcontroladores.

En primer lugar, un microcontrolador integra en un único encapsulado lo que antes necesitaba una o varias decenas de elementos distintos. Como resultado de estos, se ha obtenido una evidente disminución en el volumen del hardware y del circuito impreso. Esta integración también ha tenido como consecuencia inmediata la simplificación del diseño del circuito impreso, ya que no es necesario llevar los buses de direcciones y de datos de un componente a otro.

En segundo lugar, de todo lo anterior se deriva un aumento de la fiabilidad del sistema ya que, al disminuir el número de componentes, el número potencial de fuentes de error también disminuye. Además, la cantidad de conexiones entre componentes/zócalos o componentes/circuito impreso también disminuye, lo que aumenta la fiabilidad del sistema. Así mismo, la disminución de componentes reduce el consumo global de todo el sistema, lo que según en que aplicaciones y tipos de alimentación se utilice el microcontrolador puede resultar ventajoso.

Los mayores inconvenientes de los microcontroladores son bastante pocos y, principalmente, se encuentran en el nivel de la programación, pero en dos planos diferentes. El primer inconveniente es el sistema de almacenamiento de los programas que lo hacen funcionar, como ya hemos visto, las opciones de almacenamiento pasan por utilizar una memoria ROM en alguna de su variantes (ROM por máscara, UVPROM, OTPROM, EEPROM, etc...), esto implica que la modificación de los programas realizados va a suponer un esfuerzo de borrado de la memoria completa (o de bloques en le mejor de los casos) o la sustitución del chip de memoria por uno nuevo, lo cual conlleva gastos adicionales en material o en esfuerzo.

El otro inconveniente es el de que en los microcontroladores, como cualquier sistema programado, hay que disponer de una herramienta o medio de desarrollo, es necesario escribir los programas, probarlos y comprobarlos sobre el hardware que rodea al microcontrolador, antes de instalarlos y hacer funcionar el sistema. Este sistema de desarrollo representa, por tanto, una inversión que hay que tener en cuenta en el coste del producto final. Si se prevé la realización de aparatos diversos que utilicen microcontroladores de la misma familia, es bastante fácil de amortizar; en caso contrario puede ser más difícil.

2.- Presentación de un microcontrolador. PIC 16X84.

2.1.- Descripción general.

Microcontrolador diseñado por la empresa Microchip (http://www.microchip.com), que se fabrica en varias versiones de las cuales las más simples, pero interesantes, son la 16C84 y la 16F84. Son idénticas en cuanto a su arquitectura interna a excepción de la memoria ROM y la memoria RAM. En el primer caso contiene una memoria EEPROM de

1Kbytes de 14 bits cada una, en el segundo diseño la memoria tiene la misma capacidad pero es de tipo Flash.

Tal y como se ha comentado, la memoria EEPROM y la Flash son eléctricamente grabables y borrables, lo que permite escribir y borrar el programa bajo prueba manteniendo el microcontrolador en el mismo zócalo y usando el mismo dispositivo para grabar y borrar. Esta característica supone una gran ventaja frente a la mayoría de los microcontroladores, que tienen como memoria de programa reescribible una tipo EPROM. Se graba eléctricamente, pero para borrarlas hay que someterlas durante cierto tiempo a rayos ultravioleta, lo que implica sacar del zócalo el circuito integrado y colocarlo en un borrador de EPROM.

Otra ventaja del PIC16X84 en cuanto a simplificar el proceso de escritura, borrado y reescritura de programas, tan necesario en la fase de diseño, es su sistema de grabación de datos, que se realiza en serie. Para escribir un programa en la memoria se manda la información en serie usando sólo dos patillas, una para la señal de reloj y otra para los datos serie. A continuación exponemos las características más significativas:

MEMORIA DE PROGRAMA: 1 K x 14
MEMORIA DE DATOS: 36 bytes (PIC16C84) y 68 bytes (PIC16F84)
MEMORIA DE DATOS EEPROM: 64 bytes
PILA (Stack): De 8 niveles
INTERRUPCIONES: 4 tipos diferentes
JUEGO DE INSTRUCCIONES: 35
ENCAPSULADO: Plástico DIP de 18 patillas
FRECUENCIA DE TRABAJO: 10 MHz máxima
TEMPORIZADORES: Sólo uno el TMR0. Tambien tiene Perro Guardián (WDT)
LINEAS DE E/S DIGITALES: 13 (5 Puerta A y 8 Puerta B)
VOLTAJE DE ALIMENTACION (V_{DD}): De 2 a 6 V DC
VOLTAJE DE GRABACION (V_{PP}): De 12 a 14 V DC

Existen otras variantes que se comercializan y responden a la nomenclatura genérica PIC16X8X, atendiendo a diversas características como pueden ser la frecuencia máxima de funcionamiento, el tipo de oscilador externo o el margen del voltaje de alimentación.

2.2.- Aspecto externo.

EL PIC16C(F)84 está fabricado con tecnología CMOS de altas prestaciones y encapsulado en plástico con 18 patillas, con la nomenclatura que se muestra en la figura 3. La misión de cada patilla comentada brevemente es:

V_{DD}: Tensión positiva de alimentación.
V_{SS}: Tensión conectada a tierra o negativa de alimentación.
OSC1/CLKIN: Entrada del circuito oscilador externo que proporciona la frecuencia de trabajo del microcontrolador.
OSC2/CLKOUT: Patilla auxiliar del circuito oscilador.
MCLR#: Patilla activa con nivel lógico bajo, lo que se representa con el símbolo #. Su activación origina la reinicialización o Reset del PIC. También se usa durante la grabación de la memoria para introducir por ella la tensión V_{PP}.

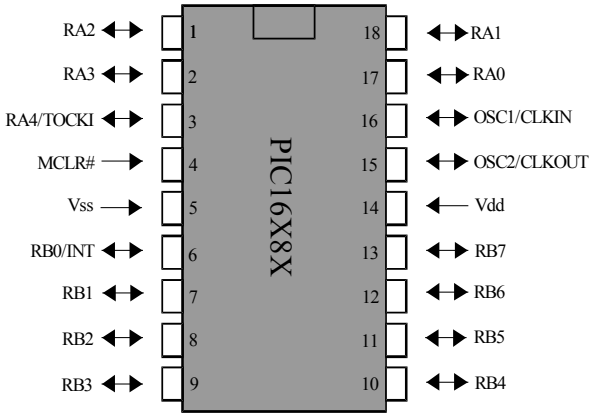

Figura 3

RA0-RA4: Son las 5 líneas de E/S digitales correspondientes a la Puerta A. La línea RA4 multiplexa otra función expresada por TOCKI. En ese caso sirve para recibir una frecuencia externa para alimentar al temporizador TMR0.
RB0-RB7: Son las 8 líneas de E/S digitales de la Puerta B. La línea RB0 multiplexa la función de servir como entrada a una petición externa de una interrupción.

2.3.- La frecuencia de funcionamiento. El reloj.

La frecuencia de trabajo del microcontrolador es un parámetro fundamental a la hora de establecer la velocidad de ejecución de instrucciones y el consumo de energía. Cuando un PIC16X84 funciona a 10 MHz, que es su máxima frecuencia, le corresponde un ciclo de instrucción de 400 ns, puesto que cada instrucción tarda en ejecutarse 4 períodos de reloj, o sea, 4 x 100 ns = 400 ns. Todas las instrucciones del PIC se realizan en un ciclo de instrucción, menos las de salto que tardan el doble.

Los impulsos de reloj entrar por la patilla OSC1/CLKIN y se dividen por 4 internamente, dando lugar a las señales Q1, Q2, Q3 y Q4. Durante un ciclo de instrucción, que comprende las cuatro señales mencionadas, se desarrollan las siguientes operaciones:

Q1: Durante este impulso se incrementa el Contador de Programa.
Q4: Durante este impulso se busca el código de la instrucción en la memoria del programa y se carga en el Registro de Instrucciones.
Q2-Q3: Durante la activación de estas dos señales se produce la decodificación y la ejecución de la instrucción.

Para conseguir ejecutar cada instrucción en un ciclo de instrucción (excepto las de salto), se aplica la técnica de segmentación o *pipe-line*, que consiste en realizar en paralelo las dos fases que comprende cada instrucción.

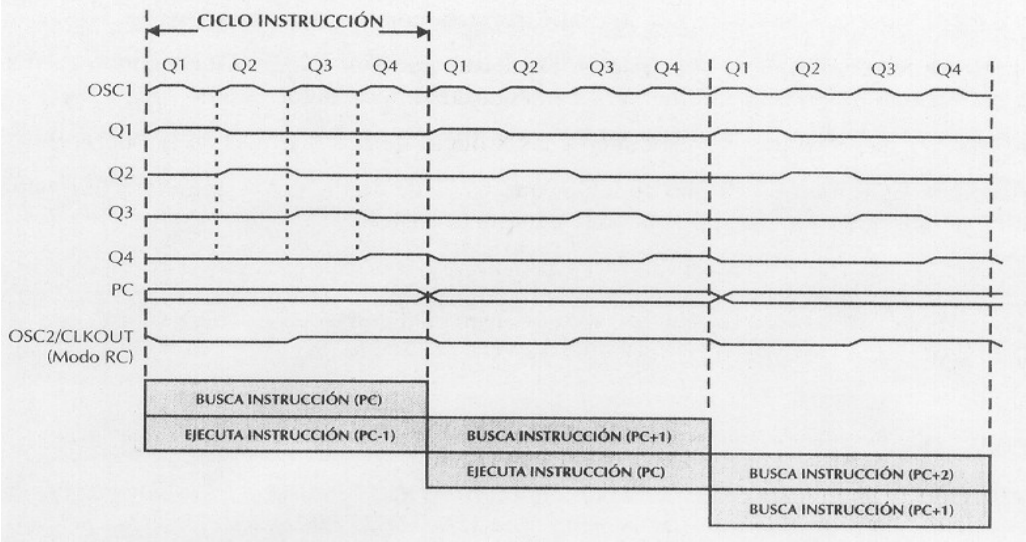

Figura 4.

La estructura segmentada del procesador permite realizar al mismo tiempo la fase de ejecución de una instrucción y la de búsqueda de la siguiente. Cuando la instrucción ejecutada corresponde a un salto no se conoce cuál será la siguiente hasta que se realice, por eso en esta situación se sustituye la fase de búsqueda por un ciclo "vacio", oroginando que las instrucciones de salto tarde en realizarse dos ciclos de instrucción.

Tipos de osciladores

Los PIC admiten cuatro tipos de osciladores externos para aplicarles la frecuencia de funcionamiento, se colocan entre las patillas OSC1 y OSC2. El tipo empleado debe especificarse en dos bits de la "Palabra de Configuración", como se comentará más adelante. Los tipos que se pueden emplear son:

- *Oscilador RC:* Es un oscilador de bajo coste formado por una simple resistencia y un condensador. Proporciona una estabilidad mediocre de la frecuencia, cuyo valor depende de los valores de los dos elementos R-C.
- *Oscilador HS:* Es un oscilador que alcanza una alta velocidad comprendida entre 4 y 10 MHz y está basado en un cristal de cuarzo o un resonador cerámico.
- *Oscilador XT:* Es un oscilador de cristal o resonador para frecuencias estándar comprendidas entre 100 KHz y 4 MHz.
- *Oscilador LP:* Oscilador de bajo consumo con cristal o resonador diseñado para trabajar en un rango de frecuencias de 35 a 200 KHz.

2.4.- Reinicialización o RESET.

Cuando se aplica un nivel lógico bajo a la patilla MCLR# el microcontrolador reinicializa su estado. Dos acciones importantes se producen en la reinicialización o RESET:

1. El Contador de Programa se carga con la dirección 0, apuntando a la primera dirección de la memoria de programa en donde deberá estar situada la primera instrucción del programa de aplicación.
2. la mayoría de los registros de estado y control del procesador toman un estado conocido y determinado.

Se puede provocar el RESET de varias maneras, pero si se desea realizar manualmente, habrá que colocar, conectado a la patilla MCLR#, un circuito con un pulsador, que al ser apretado genere un nivel lógico bajo.

3.- Arquitectura de los microcontroladores PIC16X84.

Para lograr una compactación de código óptima y una velocidad superior a la de sus competidores, los microcontroladores PIC incorporan en su procesador tres de las características más avanzadas en los grandes computadores:

- Procesador tipo RISC.
- Ejecución segmentada.
- Arquitectura HARVARD.

El juego de instrucciones se reduce a 35 y sus modos de direccionamiento se han simplificado al máximo. Con la estructura segmentada se pueden realizar simultáneamente las dos fases en que se descompone cada instrucción. Con la arquitectura HARVARD se puede acceder de forma simultánea e independiente a la memoria de datos y a la de programa. El aislamiento y la diferenciación de los dos tipos de memoria permite que cada uno tenga la longitud y el tamaño más adecuado. De esta forma, en el PIC16X84 la longitud de datos es de un byte, mientras que la de las instrucciones es de 14 bits.

Otra característica relevante de los PIC es el manejo intensivo del Banco de Registros, los cuales participan de manera muy activa en la ejecución de las instrucciones. De igual forma, la memoria RAM complementa los registros internos implementando en sus posiciones registros de propósito específico y de propósito general.

La arquitectura interna del PIC16X84 se presenta en la figura 5 y consta de siete bloques fundamentales.

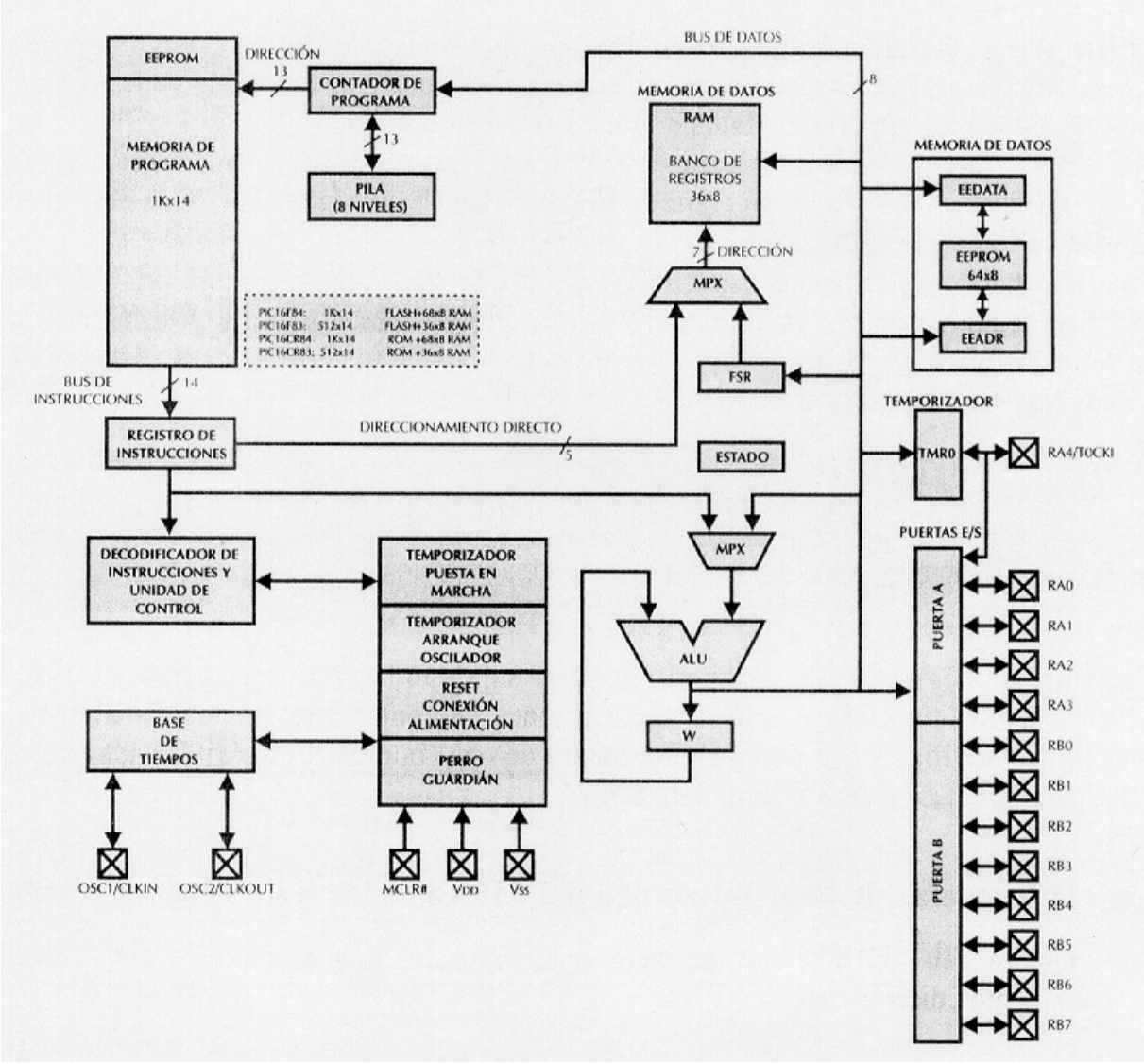

Figura 5.

1. Memoria de programa EEPROM o Flash de 1 k x 14 bits.
2. Memoria de datos formada por dos áreas. Una RAM donde se alojan 22 registros de propósito específico (SFR) y 36 de propósito general (GDR), y otra de tipo EEPROM de 64 bytes.
3. Camino de datos con una ALU de 8 bits y un registro de trabajo W del que normalmente recibe un operando y envía el resultado. EL otro operando puede provenir del bus de datos o del propio código de la instrucción.
4. Diversos recursos conectados al bus de datos, tales como Puertas de Entrada/Salida, Temporizador TMR0, etc.
5. Base de tiempos y circuitos auxiliares.
6. Direccionamiento de la memoria de programa en base al Contador de Programa ligado a una Pila de 8 niveles de profundidad.
7. Direccionamiento directo e indirecto de la memoria RAM.

3.1.- Memoria de Programa.

La arquitectura de los PIC de la gama media admite un mapa de memoria de programa capaz de contener 8192 instrucciones de 14 bits cada una. Este mapa se divide en páginas de 2048 posiciones. para direccionar 8 K posiciones se necesitan 13 bits, que es la longitud que

tiene el Contador de Programa. Sin embargo el PIC16X84 solo tiene implementadas 1 K posiciones, por lo que se ignoran los tres bits de más peso del PC.

Al igual que todos los registros específicos que controlan la actividad del procesador, el Contador de Programa está implementado sobre un par de posiciones de la memoria RAM. Cuando se escribe el Contador de Programa como resultado de una operación de la ALU, los 8 bits de menos peso del PC residen en el registro PCL, que ocupa, repetido, la posición 2 de los dos bancos de la memoria de datos. Los bits de más peso, PC <12:8>, residen en los 5 bits de menos peso del registro PCLATH, que ocupa la posición 0A H de los dos bancos de la memoria RAM. En las instrucciones de salto, los 11 bits de menos peso del PC provienen del código de instrucción y los otros dos de los bits de PCLATH <4:3>, tal y como se muestra en la figura 6.

La Pila es una zona aislada de las memorias de instrucciones y datos. Tiene una estructura LIFO, en la que el último valor guardado es el primero que sale. Tiene 8 niveles de profundidad cada uno con 13 bits. Funciona como un buffer circular, de manera que el valor que se obtiene al realizar el noveno POP es igual al que se obtuvo en el primero.

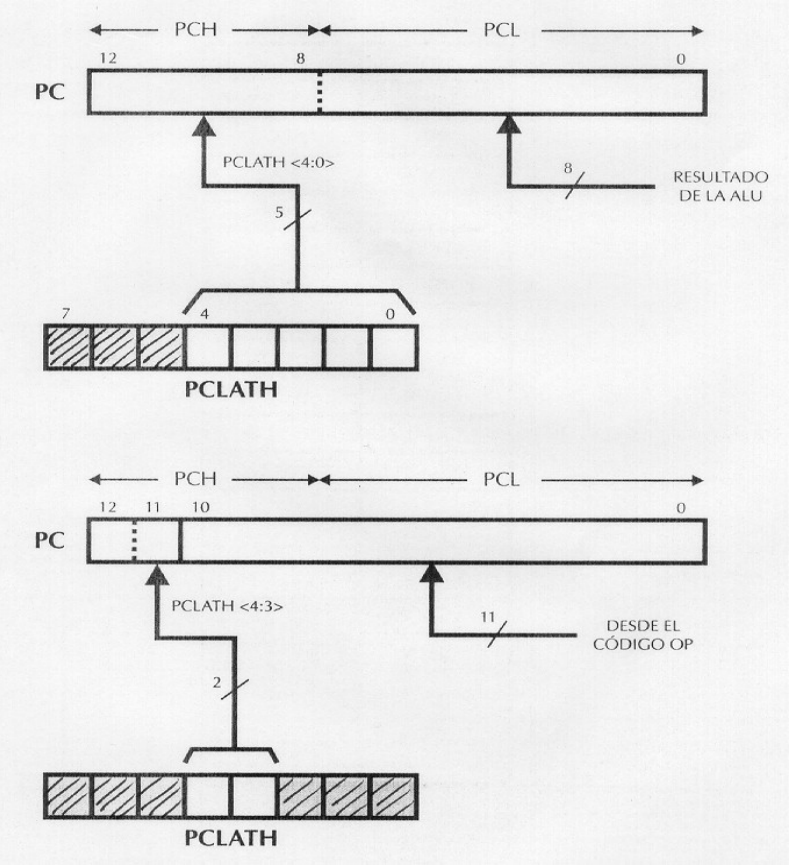

Figura 6.

3.2.- Memoria de Datos RAM.

La memoria de datos del PIC16X84 dispone de dos zonas diferentes:

1. **Area de RAM estática o SRAM**, donde reside el Banco de Registros Específicos (SFR) y el Banco de Registros de Propósito General (GPR). El primer banco tien 24 posiciones de tamaño byte, aunque dos de ellas no son operativas, y el segundo 68.

2. **Area EEPROM**, de 64 bytes donde, opcionalmente, se pueden almacenar datos que no se pierden al deconectar la alimentación.

La zona de memoria RAM se halla dividida en dos bancos (banco 0 y banco 1) de 128 bytes cada uno. En el PIC16X84 sólo se hallan implementadas físicamente las 48 primeras posiciones de cada banco, de las cuales las 12 primeras están reservadas a los Registros de Propósito Específico (SFR), que son los encargados del control del procesador y sus recursos. Algunos de dichos registros se hallan repetidos en la misma dirección de los dos bancos, para simplificar su acceso (INDF, ESTADO, FSR, PCLATH e INTCON). Las posiciones apuntadas por la dirección 7 H y la apuntada por la 87 H no son operativas. Los 36 registros restantes de cada banco se destinan a Registros de Propósito General y en realidad sólo son operativos los 68 del banco 0 porque los del banco 1 se mapean sobre el banco 0, es decir, cuando se apunta a un registro general del banco 1, se accede al mismo del banco 0, véase la figura 7.

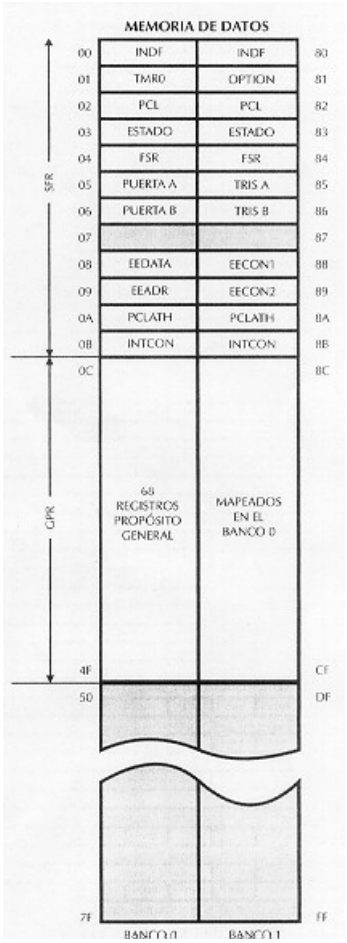

Figura 7

Para seleccionar el banco a acceder hay que manipular el bit 5 (RP0) del registro ESTADO. Si RP0=1 se accede ala banco 1 y si RP0=0 se accede al banco 0. Tras un Reset se accede automáticamente al banco 0. Para seleccionar un registro de propósito general no hay que tener en cuenta el estado del bit RP0, porque al estar mapeado el banco 1 sobre el banco 0, cualquier direccionamiento de un registro del banco 1 corresponde a su homólogo del banco 0. En el direccionamiento directo a los registro GPR se ignora el bit de más peso, que identifica el banco y sus direcciones están comprendidas entre el valor 0x0C y 0x4F en hexadecimal.

Direccionamiento de la memoria de datos.

En los PIC de gama media la memoria de datos está organizada para alojar un máximo de 4 bancos de 128 bytes cada uno. Los PIC16C84 sólo tienen implementados los 48 primeros bytes de los bancos 0 y 1. En el resto de los PIC de esta familia se destinan dos bits del registro ESTADO (RP0 y RP1) para determinar el banco y otros siete para elegir una de las 128 posiciones del banco seleccionado, tal y como muestra la figura 8.

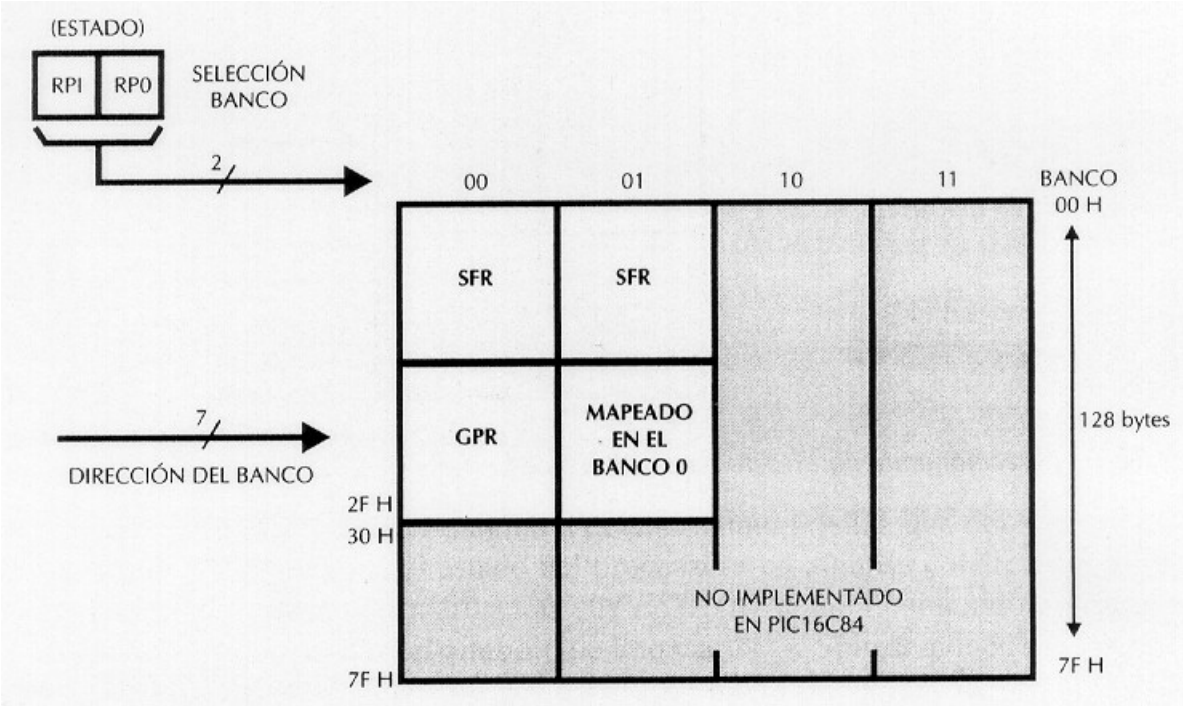

Figura 8

- Direccionamiento Directo: El operando que utiliza la instrucción en curso se referencia mediante su dirección, que viene incluida en el código OP de la misma, concretamente en los 7 bits de menos peso. El banco a acceder lo determinan los bits RP0 y RP1 del registro ESTADO. En el caso del PIC16C84 sólo se usa el bit RP0 al tener implementados únicamente dos bancos.

- Direccionamiento Indirecto: Este modo de direccionamiento se usa cuando en una instrucción se utiliza como operando el registro INDF, que ocupa la dirección =de ambos bancos. En realidad el registro INDF no está implementado físicamente y cuando se le hace referencia, se accede a la dirección de un banco especificada con los bits de menos peso del registro FSR. El bit de más peso de FSR junto al bit IRP del registro ESTADO se encargan de seleccionar el banco a acceder, mientras que los 7 bits de menos peso apuntan a la posición. Como sólo hay dos bancos en el PIC16C84 en este modo de direccionamiento, el bit IRP es 0 siempre.

3.3.- El Registro de Estado.

Hasta ahora ESTADO es el registro más usado y llega el momento de describirlo en su totalidad. Ocupa la dirección 3 tanto del banco 0 como del 1 de la memoria de datos RAM. Sus bits tienen tres misiones distintas:

1. Se encargan de avisar de la incidencias del resultado de la ALU (C, DC y Z).
2. Indican el estado de Reset (TO# y PD#).
3. Seleccionan el banco a acceder en la memoria de datos (IRP, RP0 y RP1).

En la figura 9 se muestra el diagrama de distribución de los bits del registro ESTADO, su misión es la siguiente:

C: Acarreo en el bit de más peso

1: Acarreo en el bit de más peso.
0: No acarreo en el bit de más peso.

DC: Acarreo en el 4º bit

1: Acarreo en el 4º bit.
0: No acarreo en el 4º bit.

Z: Cero

1: El resultado de una instrucción lógico-aritmética ha sido cero.
0: El resultado de una instrucción lógico-aritmética no ha sido cero.

PD#: Power Down

1: Se pone a este valor después de la conexión a la alimentación o al ejecutar *clrwdt*.
0: Se pone a este valor al ejecutar *sleep*.

TO#: Time Out

1: Se pone a este valor después de la conexión a la alimentación o al ejecutar *clrwdt* y *sleep*.
0: Se pone a este valor al desbordarse el Perro Guardián (*Watchdog*).

RP1-RP0: Selección de banco en direccionamiento directo

Como el PIC16X84 sólo tiene dos bancos únicamente emplea el bit RP0, de forma que cuando vale 1 se accede al banco 1 y cuando vale 0 se accede al banco 0. Después de un Reset RP0=0.

IRP: Selección del banco en direccionamiento indirecto

Este bit junto con el de más peso del registro FSR sirven para determinar el banco de la memoria de datos seleccionado. En el PIC16X84 al disponer de dos bancos no se usa este bit.

Figura 9

3.4.- Temporizador/Contador TMR0.

Una de las labores más habituales en los programas de control de dispositivos suele ser determinar intervalos concretos de tiempo, y recibe le nombre de "temporizador" (*timer*) el elemento encargado de realizar esta función. También suele ser frecuente contar los impulsos que se producen en el exterior del sistema, y el elemento destinado a este fin se denomina "contador". Si las labores del temporizador o contador las asignamos al programa principal robarían mucho tiempo al procesador en detrimento de actividades más importantes. Por este motivo se diseñan recursos específicamente orientados a estas misiones.

Los PIC16X84 poseen un temporizador/contador de 8 bits, llamado TMR0, que actúa de dos maneras diferentes:

1.ª Como contador de sucesos, que están representados por los impulsos que se aplican a la patilla RA4/T0CKI. Al llegar al valor FF H se desborda el contador y, con el siguiente impulso, pasa a 00 H, advirtiendo esta circunstancia con la activación de un señalizador y/o provocando una interrupción.
2.ª Como temporizador, cuando se carga en el registro que implementa el recurso un valor inicial se incrementa con cada ciclo de instrucción (Fosc/4) hasta que se desborda, o sea, pasa de FF H a 00 H y avisa poniendo a el bit señalizador y/o provocando una interrupción.

Para que el TMR0 funcione como un cantador de impulsos aplicados a la patilla T0CKI hay que poner a 1 el bit T0CS, que es el que ocupa la posición 5 del registro OPTION. En esta situación, el registro TMR0, que es el ubicado en la dirección 1 del banco 0 de la memoria de datos, se incrementa con cada flanco activo aplicado a la patilla T0CKI. El tipo de flanco activo se elige programando el bit T0SE, que es el que ocupa la posición 4 del registro OPTION. Si T0SE 0 1, el flanco activo es el de bajada, y si T0SE = 0, es el de subida. Cuando se desea que TMR0 funcione como temporizador el bit T0CS = 0.

En realidad, los PIC16X84 disponen de dos temporizadores, el TMR0 y el Perro Guardián (*Watchdog*). El primero actúa como principal y sobre él recae el control de tiempos y la cuenta de impulsos. El otro vigila que el programa no se cuelgue, y para ello cada cierto tiempo comprueba si el programa se está ejecutando normalmente. En caso contrario, si el control está detenido en un bucle infinito a la espera de algún acontecimiento que no se produce, el Perro Guardián "ladra", lo que se traduce en un Reset que inicializa todo el sistema.

A menudo el TMR0 y el Perro Guardián precisan controlar largos intervalos de tiempo y necesitan aumentar la duración de los impulsos de reloj que les incrementa. Para cubrir este requisito se dispone de un circuito programable denominado Divisor de frecuencia, que divide la frecuencia utilizada por diversos rangos. Para programar el comportamiento del TMR0, el Perro Guardián y el Divisor de frecuencia se utilizan algunos bits del registro OPTION y de la Palabra de Configuración, que se verán más adelante. En la figura 10 se proporciona un esquema simplificado de la arquitectura del circuito de control de tiempos usado en los PIC16X84.

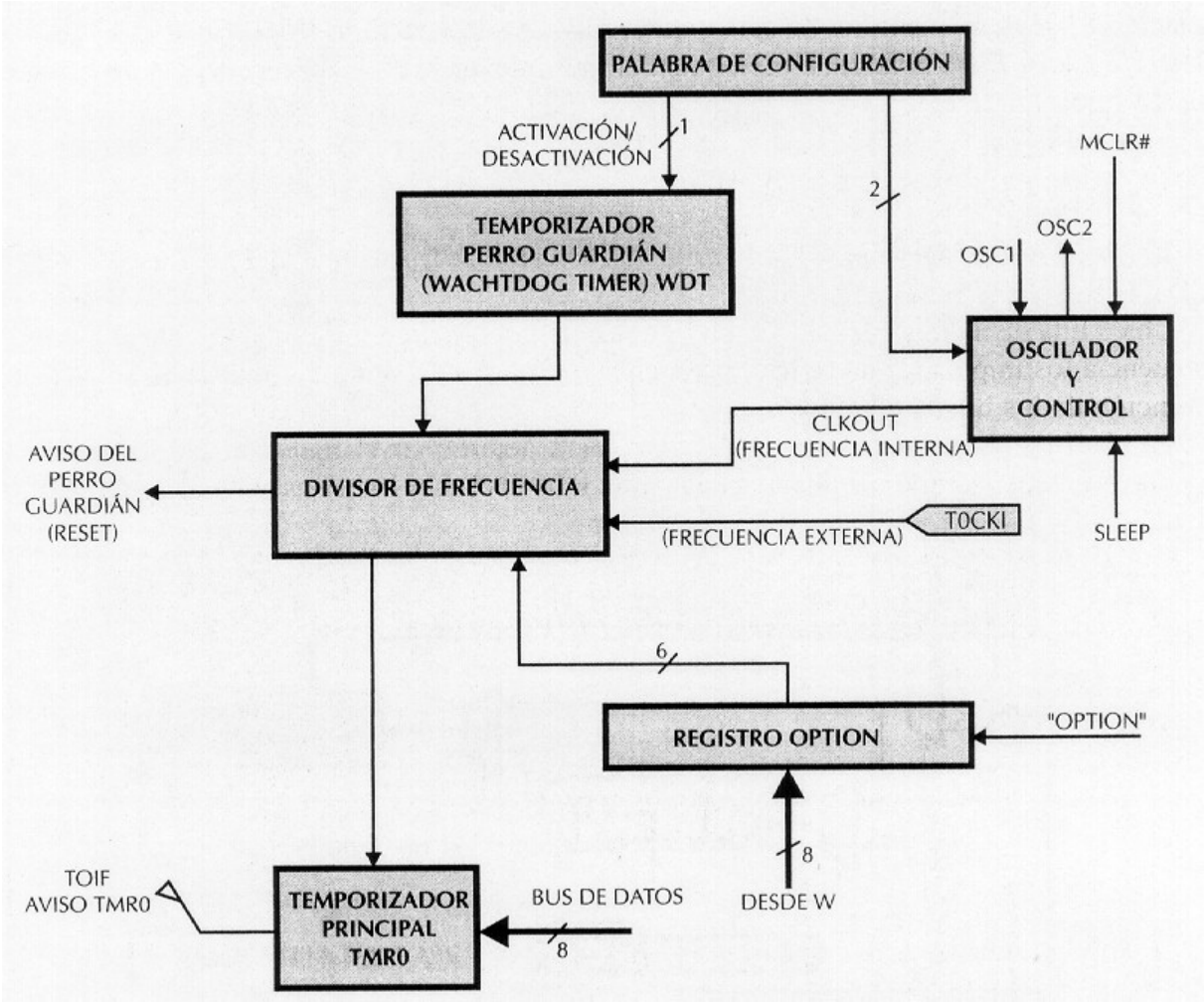

Figura 10

El Divisor de frecuencias puede usarse con el TMR0 o con el WDT. Con el TMR0 actúa como *Pre-divisor*, es decir, los impulsos pasan primero por el Divisor y luego se aplican al TMR0, una vez aumentada su duración. Con el WDT actúa después, realizando la función de *Post-divisor*. Los impulsos, que divide por un rango el Divisor de frecuencia, pueden provenir de la señal de reloj interna (Fosc/4) o de los que se aplican a la patilla T0CKI. El TMR0 se comporta como un registro de propósito específico (SFR) ubicado en la dirección 1 del banco 0 de la memoria de datos. EN igual dirección, pero en el banco 1, se halla el registro OPTION.

TMR0 puede ser leído y escrito en cualquier momento al estar conectado al bus de datos. Funciona como un contador ascendente de 8 bits. Cuando funciona como temporizador conviene cargarle con el valor de los impulsos que se quiere temporizar, pero expresados en complemento a 2. De esta manera, al llegar al número de impulsos deseados se desborda y al pasar por 00 H se activa el señalizador TOIF y/o se produce una interrupción.

Para calcular los tiempos a controlar con TMR0 se utilizan las siguientes fórmulas prácticas.

Temporización = 4 • Tosc • (Valor cargado en TMR0) • (Rango del Divisor)

Valor a cargar en TMR= = Temporización/4 • Tosc • Rango del Divisor

En cualquier momento se puede leer el valor que contiene TMR0, sin detener su cuenta. En la figura 11 se ofrece el esquema de funcionamiento de TMR0. Obsérvese que hay un bloque que retrasa 2 ciclos la cuenta para sincronizar el momento del incremento producido por la señal aplicada a T0CKI con el que se producen los impulsos internos de reloj. Cuando se escribe TMR0 se retrasan 2 ciclos su reincremento y se pone a 0 el Divisor de frecuencia.

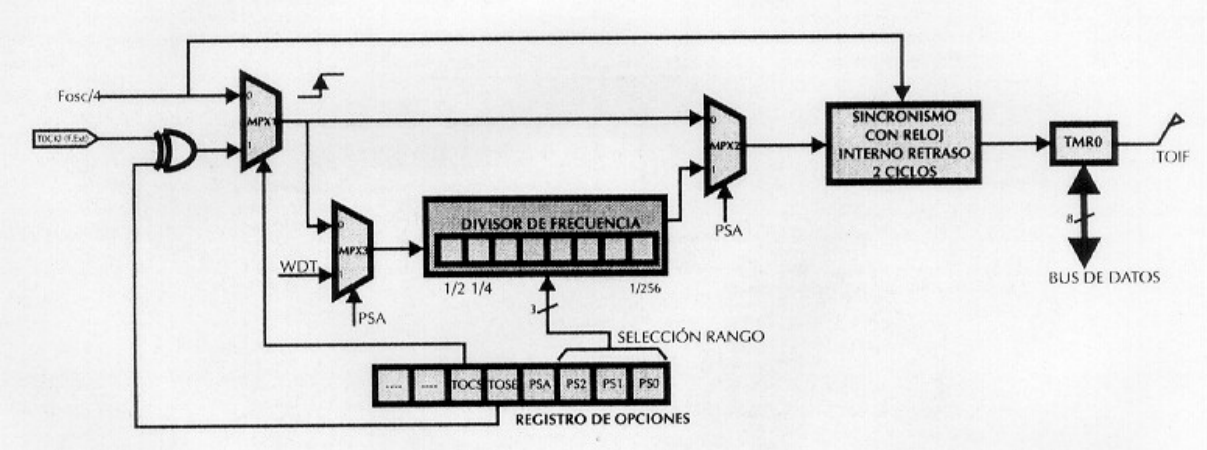

Figura 11

3.5.- El registro OPTION.

La misión principal de este registro es controlar TMR0 y el Divisor de frecuencia. Ocupa la posición 81 H de la memoria de datos, que equivale a la dirección 1 del banco 1. EL bit T0CS (*Timer 0 Clock edge Select*) selecciona en el multiplexor MPX1 la procedencia de los impulsos de reloj, que pueden ser del oscilador interno (Fosc/4) o los que se aplican desde el exterior por la patilla T0CKI. El bit T0SE (Timer 0 clock Source sElect) elige el tipo de flanco activo en los impulsos externos. El bit PSA del registro OPTION asigna el Divisor de frecuencia al TMR0 (PSA= 0) o al WDT (PSA = 1).

Los 3 bits de menos peso de OPTION seleccionan el rango por el que divide el Divisor de frecuencia los impulsos que se le aplican en su entrada. El bit 6 INTEDG (INTerrupt EDGe) sirve para determinar el flanco activo que provocará una interrupción externa al aplicarse a la patilla RB0/INT. Un 1 si es de subida y un 0 si es de bajada. El bit 7 RBPU# (RB Pull-Up) activa, si vale 0, o desactiva, cuando vale 1, las resistencias Pull-Up que pueden conectarse en las líneas de la Puerta B. .La figura 12 muestra la distribución y función de los bits de OPTION.

RBPO#	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0

PS2: PS0 Valor con el que actúa el Divisor de frecuencia

PS2	PS1	PS0	División del TMR0	División del WDT
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

PSA: Asignación del Divisor de frecuencias

1 = El Divisor de frecuencia se le asigna al WDT
0 = El Divisor de frecuencia se le asigna al TMR0

TOSE: Tipo de flanco en T0CKI

1 = Incremento de TMR0 cada flanco descendente
0 = Incremento de TMR0 cada flanco ascendente

TOCS: Tipo de reloj para el TMR0

1 = Pulsos introducidos a través de T0CKI (contador)
0 = Pulsos de reloj interno Fosc/4 (temporizador)

INTEDG: Flanco activo interrupción externa

1 = Flanco ascendente
0 = Flanco descendente

RBPO#: Resistencias Pull-up Puerta B

1 = Desactivadas
0 = Activadas

Figura 12

3.6.- El Perro Gardián (WDT).

Se trata de un contador interno de 8 bits que origina un *Reset* cuando se desborda. Su control de tiempos es independiente del TMR0 y está basado en un simple circuito R-C. Su actuación es opcional y puede bloquearse para que no funcione programando el bit WDTE de la palabra de Configuración. La temporización nominal con la que se halla programado el Perro Guardián es de 18 ms, pero utilizando el Divisor de frecuencia puede aumentarse hasta alcanzar los 2,3 segundos.

Para evitar que se desborde el Perro Guardián hay que refrescarle previamente. En realidad este refresco consiste en ponerle a cero mediante las instrucciones *clrwdt* y *sleep*. El programador debe analizar las instrucciones de la tarea y situar alguna de esas dos en sitios estratégicos por los que pase el flujo de control antes que transcurra el tiempo asignado al WDT. De esta manera si el programa se cuelga no se refresca el Perro Guardián y se produce la reinicialización del sistema.

La instrucción *clrwdt* borra al WDT y reinicia su cuenta. Sin embargo, la instrucción *sleep*, además de borrar WDT detiene al sistema y lo mete en un estado de reposo o de bajo consumo. Si no se desactiva el Perro Guardián al entrar en modo de reposo, al completar su cuenta provocará un *Reset* y sacará al microcontrolador del modo de bajo consumo. Para desactivar el Perro Guardián hay que escribir un 0 en el bit 2 (WDTE) de la Palabra de Configuración.

3.7.- Las Puertas de E/S.

Los PIC16X84 sólo disponen de dos puertas de E/S. La Puerta A posee 5 líneas, RA0-RA4, y una de ellas soporta dos funciones multiplexadas. Se trata de RA4/TOCKI, que puede actuar como línea de E/S o como la patilla por la que se reciben los impulsos que debe contar TMR0. La Puerta B tiene 8 líneas, RB0-RB7, y también tiene una con funciones multiplexadas, la RB0/INT, que, además de línea típica de E/S, también sirve como patilla por la que reciben los impulsos externos que provocan una interrupción.

Cada línea de E/S puede configurarse independientemente como entrada o como salida, según se ponga a 1 o a 0, respectivamente, el bit asociado del registro de configuración de cada puerta (TRISA y TRISB). Se llaman PUERTAA y PUERTAB los registros que guardan la información que entra o sale por la puerta y ocupan las direcciones 5 y 6 del banco 0 de la memoria de datos. Los registros de configuración TRISA y TRISB ocupan las mismas direcciones pero en el banco 1. Al reiniciarse el PIC todos los bits de los registros TRIS quedan a 1, con lo que las líneas de las puertas quedan configuradas como entradas. Cada línea de salida puede suministrar una corriente máxima de 20 mA y si es de entrada puede absorber hasta 25 mA. Al existir una limitación en la disipación máxima de la potencia del chip se restringe la corriente máxima de absorción de la Puerta A a 80 mA y la de suministro a 50 mA. La Puerta B puede absorber un máximo de 150 mA y suministrar un total de 100 mA.

La Puerta A.

Las líneas RA3-RA0 admiten niveles de entrada TTL y de salida CMOS. La línea RA4/TOCKI dispone de un circuito Trigger Schmitt que proporciona una buena inmunidad al ruido y la salida tiene drenador abierto. RA4 multiplexa su función de E/S con la entrada de impulsos externos para el TMR0. En el circuito de la figura 13 se muestra la adaptación de las patillas RA3-RA0 a las señales internas del procesador.

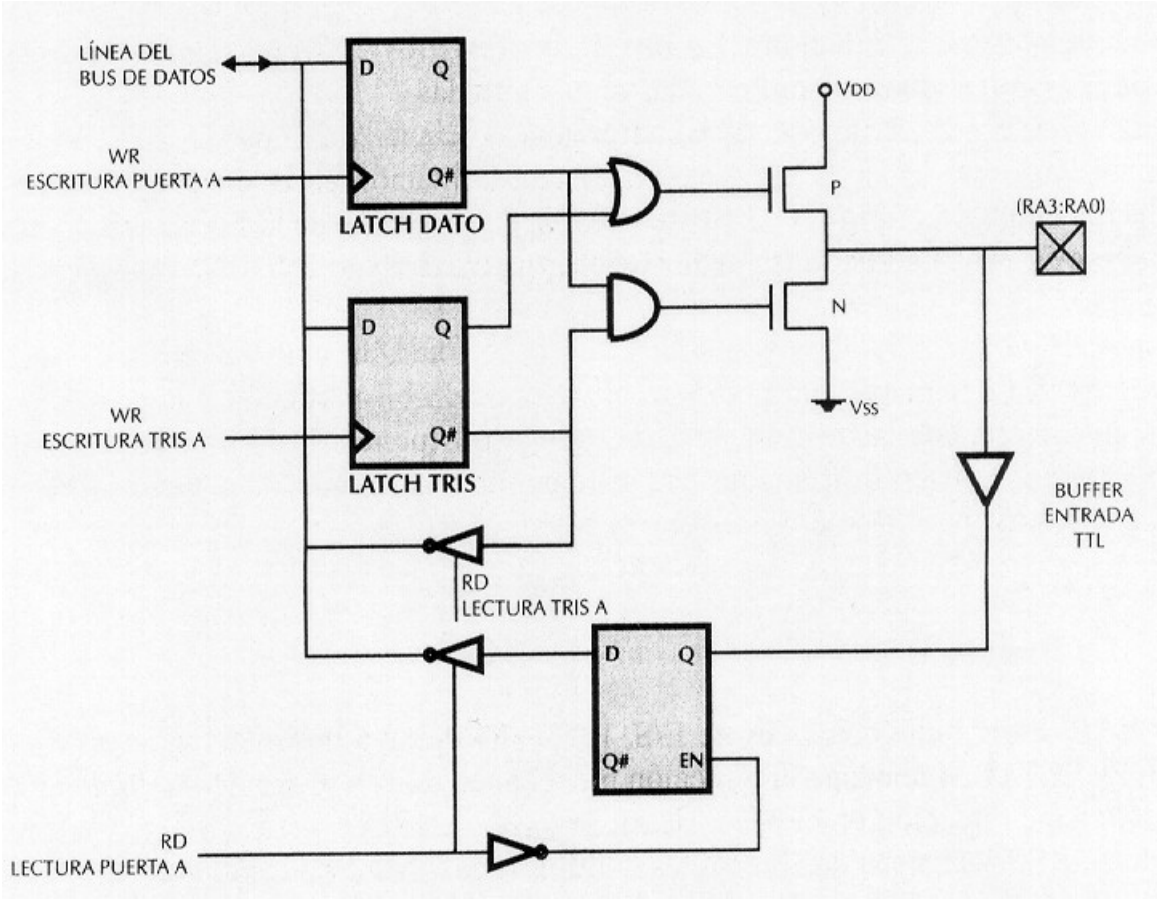

Figura 13

Cuando se lee una línea de la Puerta A se recoge el nivel lógico que tiene en ese momento. Las líneas cuando actúan como salidas están conectadas a unos biestable, lo que significa que sus patillas sacan el nivel lógico que se haya cargado por última vez en el registro PUERTAA. La escritura de una puerta, tal y como se ve en la figura 13, implica que primero se deposita el nivel lógico en la línea correspondiente del bus interno de datos y se activa la señal WRITE, lo que origina el almacenamiento de dicho nivel en el biestable. En esta situación, el biestable de configuración (TRIS) debería tener valor 0 para que actuase como salida. Con estos valores la puerta OR tendría un 0 a su salida y la AND también. Estos valores producen la conducción de los transistores MOS, en función del valor escrito, dando como salida final V_{DD} o V_{SS} (1 ó 0 respectivamente).

Si una línea actúa como entrada, el nivel lógico depositado en ella desde el exterior pasa a la línea correspondiente del bus interno de datos cuando se activa la señal READ y se hace conductor el dispositivo triestado que les une. Al programarse como entrada, los dos transistores MOS de salida quedan bloqueados y la línea de alta impedancia. Téngase en cuenta que cuando se lee una línea de entrada se obtiene el estado actual que tiene su patilla correspondiente (almacenado en el tercer biestable) y no el que haya almacenado en el biestable de datos. La información presente en una línea de entrada se muestrea al iniciarse el ciclo de instrucción y debe matenerse estable durante su desarrollo.

La Puerta B.

Consta de 8 líneas bidireccionales de E/S, RB7-RB0, cuya información se almacena en el registro PUERTAB, que ocupa la dirección 6 del banco 0. El registro de configuración

TRISB ocupa la misma dirección pero del banco 1. La línea RB0/INT tiene dos funciones multiplexadas; además de patilla de E/S actúa como la patilla para la petición de una interrupción externa, cuando se autoriza esta función mediante la adecuada programación del registro INTCON.

A todas las líneas de esta puerta se las permite conectar una resistencia *pull-up* de elevado valor con el positivo de la alimentación. Para este fin hay que programar en el registro OPTION el bit RBPU# = 0, afectando la conexión de la resistencia a todas las líneas. Con el *Reset* todas las líneas quedan configuradas como entradas y se desactivan las resistencias de *pull-up*.

Las cuatro líneas de más peso, RB7-RB4, pueden programarse para soportar una misión especial. Cuando las 4 líneas actúan como entradas se las puede programar para generar una interrupción si alguna de ellas cambia su estado lógico. Esta posibilidad es muy práctica en el control de teclados. En la figura 14 se muestra el esquema de conexionado entre las patillas RB7-RB4 y las líneas correspondientes del bus interno.

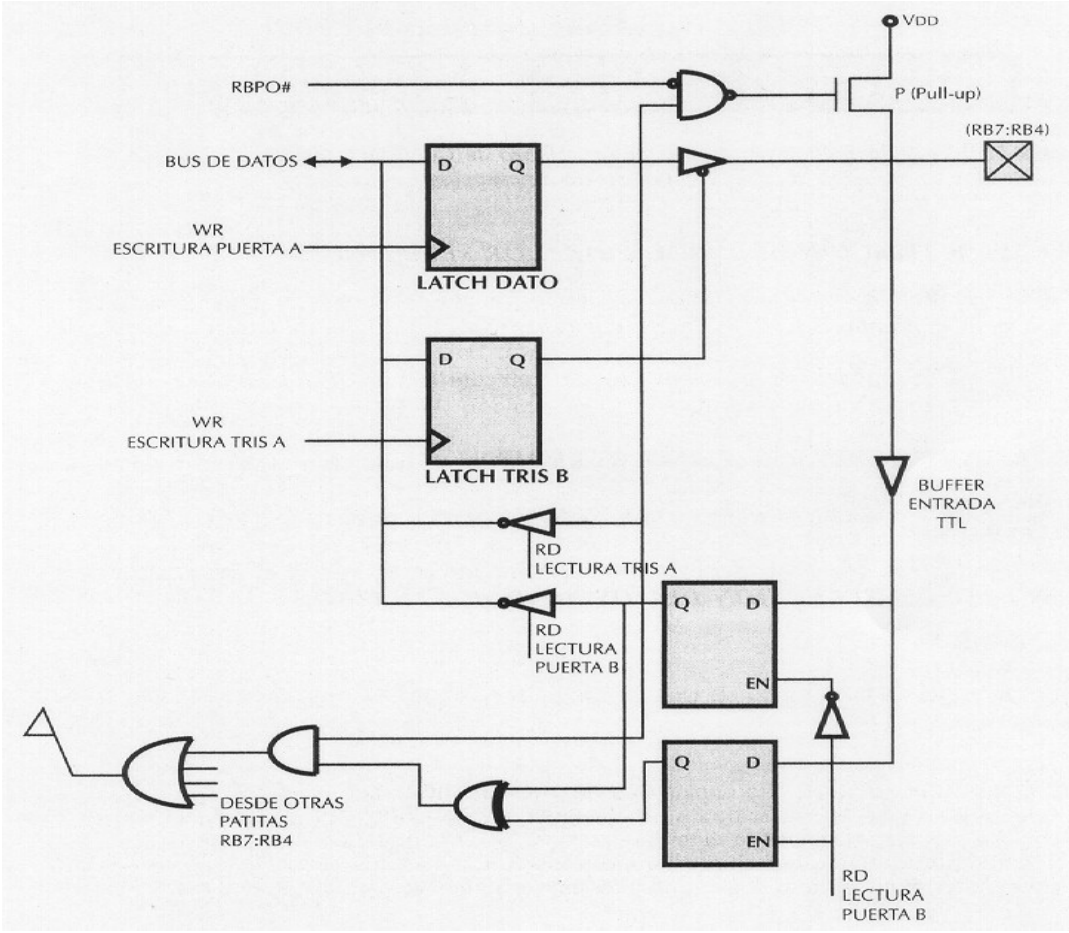

Figura 14

El estado de las patillas RB7-RB4 en modo de entrada se compara con el valor antiguo que tenían y que se había almacenado en un biestable durante la última lectura de la Puerta B. El cambio de estado en alguna de esas líneas origina una interrupción y la activación del señalizador RBIF. La línea RB6 también se utiliza para la grabación serie de la memoria de programas y sirve para soportar la señal de reloj. La línea RB7 constituye la entrada de los datos en serie.

3.8.- La Palabra de CONFIGURACION.

Se trata de una posición reservada de la memoria de programa situada en la dirección 2007 H y accesible únicamente durante el proceso de grabación. Al escribirse el programa de la aplicación es necesario grabar el contenido de esta posición de acuerdo con las características del sistema. En la figura 15 se muestra la distribución y asignación de los 14 bits de la Palabra de Configuración de los PIC16F84, que tienen las siguientes funciones:

CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	PWRTE#	WDTE	FOSC1	FOSC0

Figura 15

CP: BITS DE PROTECCIÓN DE LA MEMORIA DE CÓDIGO

1: No protegida

0: Protegida. El programa no se puede leer, evitando copias. Tampoco se puede sobreescribir. Además evita que pueda ser accedida la memoria EEPROM de datos y, finalmente, si se modifica el bit CP de 0 a 1, se borra completamente la EEPROM.

PWRTE: ACTIVACIÓN DEL TEMPORIZADOR "POWER-UP"

El temporizador "power-up" retrasa 72 ms la puesta en marcha o Reset que se produce al conectar la alimentación al PIC, para garantizar la estabilidad de la tensión aplicada.

1: Desactivado

0: Activado

WDT: ACTIVACIÓN DEL PERRO GUARDIÁN

1: Activado el WDT

0: Desactivado

FOSC1-FOSC0: SELECCIÓN DEL OSCILADOR UTILIZADO

1-1: Oscilador RC

1-0: Oscilador HS

0-1: Oscilador XT

0-0: Oscilador LP

3.9.- La memoria EEPROM de Datos.

Los PIC16X84 tienen 64 bytes de memoria EEPROM de datos, donde se pueden almacenar datos y variables que interesa que no se pierdan cuando se desconecta la alimentación al sistema. Soporta 1.000.000 de ciclos de escritura/borrado y es capaz de guardar la información sin alterarla más de 40 años. La memoria EEPROM no está mapeada en la zona de la memoria de datos donde se ubican los registros SFR y GPR. Para poder leerla y escribirla durante el funcionamiento normal del microcontrolador hay que utilizar 4 registros del banco SFR: *EEDATA*, *EEADR* y *EECON1*.

En el registro EEADR, ubicado en la dirección 9 del banco 0, se carga la dirección a acceder de la EEPROM de datos. las 64 posiciones de un byte ocupan las direcciones de un mapa que comienza en 00 H y termina en 3F H, por eso los 2 bits de más peso de este registro siempre valen 0. En el registro EEDATA, ubicado en la dirección 8 del banco 0, se depositan los datos que se leen o escriben. El registro EECON1, que ocupa la dirección 88 H de la memoria de datos, o la dirección 8 del banco 1, tiene misiones de control de las operaciones en la EEPROM y la distribución de sus bits se presenta en la figura 16, sirviendo cada uno de ellos para lo siguiente:

RD: Lectura
1: Se pone a 1 cuando se va a realizar un ciclo de lectura. Luego pasa a 0 automáticamente.

WR: Escritura
1: Se pone a 1 cuando se va a realizar un ciclo de ESCRITURA. Luego pasa a 0 automáticamente.

WREN: Permiso de escritura
1: Permite la escritura en la EEPROM.
0: Prohibe la escritura.

WRERR: Señalizador de error en escritura
1: Se pone a 1 cuando una operación de escritura ha terminado prematuramente.
0: La operación de escritura se ha completado correctamente.

EEIF: Señalizador de final de operación de escritura
1: Cuando este señalizador se pone a 1 indica que la operación se ha completado con éxito. Se pone a 0 por programa.
0: La operación de escritura no se ha completado.

------	------	------	EEIF	WRERR	WREN	WR	RD

Figura 16

4. Interrupciones.

Las llamadas a las subrutinas mediante instrucciones del tipo CALL son desviaciones del flujo de control del programa originadas por instrucciones, por lo que se consideran síncronas. Se producen cada vez que se ejecuta dicha instrucción. La interrupciones son desviaciones del flujo de control del programa originadas asíncronamente por diversos sucesos que no se hallan bajo la supervisión de las instrucciones. Dichos sucesos pueden ser externos al sistema, como la generación de un flanco o nivel activo en una patilla del microcontrolador, o bien, internos, como el desbordamiento de un contador.

El comportamiento del microcontrolador ante la interrupción es similar al de la instrucción tipo CALL de llamada a subrutina. En ambos casos se detiene la ejecución del programa en curso, se salva la dirección actual del PC en la Pila y se carga el PC con una dirección, que en el caso de CALL viene acompañando a la misma instrucción, y en el caso de una interrupción es una dirección "reservada" de la memoria de código, llamada *Vector de Interrupción* que da paso a un trozo de código denominado Rutina de Servicio de la Interrupción (RSI).

La RSI suele comenzar guardando en la memoria de datos algunos registros específicos del procesador. Concretamente aquellos que la RSI va a emplear y va a alterar su contenido. Antes del retorno al programa principal se recuperan los valores guardados y se restaura completamente el estado del procesador. Algunos procesadores salvan estos registros en la Pila, pero los PIC no disponen de instrucciones para meter (*push*) y sacar (*pop*) información de la Pila, utilizando para este fin registros de propósito general de la memoria de datos.

Los PIC16X84 pueden ser interrumpidos por 4 causas diferentes, pero todas ellas desvían el flujo de control a la dirección 0004 H, por lo que otra de las operaciones iniciales

de la RSI es averiguar cuál de las posibles causas ha sido la responsable de la interrupción en curso, para ello se exploran los señalizadores de las fuentes de interrupción. Otro detalle importante en la RSI de los PIC16X84 es que estos microcontroladores poseen un bit GIE (*Global Interrupt Enable*) que cuando vale 0 prohibe todas las interrupciones. Pues bien, al comenzar la RSI dicho bit GIE se pone automáticamente a 0, con objeto de no atender nuevas interrupciones hasta que se termine la que ha comenzado. En el retorno final de la interrupción, GIE pasa a valer automáticamente 1 para volver a tener en cuenta las interrupciones.

Antes del retorno conviene borrar el señalizador de la causa de interrupción que se ha atendido, porque si bien los señalizadores se ponen a 1 automáticamente en cuanto se produce la causa que indican, la puesta a 0 se hace por programa. En la figura 17 se muestra un organigrama de las fases más importantes que se desarrollan durante el proceso de ejecución de una interrupción.

4.1.- Causas de Interrupción.

Los PIC16X84 tienen 4 causas o fuentes posibles de interrupción:

1ª Activación de la patilla RB0/INT
2ª Desbordamiento del temporizador TMR0
3ª Cambio de estado en una de las 4 patillas de más peso (RB7-RB4) de la Puerta B
4ª Finalización de la escritura en la EEPROM de datos

Cuando ocurre cualquiera de los 4 sucesos indicados se origina una petición de interrupción, que si se acepta y se atiende comienza depositando el valor del PC actual en la Pila, poniendo el bit GIE = 0 y cargando en el PC el valor 0004 H, que es el Vector de Interrupción donde se desvía el flujo de control. Cada fuente de interrupción dispone de un señalizador o "flag", que es un bit que se pone automáticamente a 1 cuando se produce. Además cada fuente de interrupción tiene otro bit de permiso, que según su valor permite o prohibe la realización de dicha interrupción.

4.2.- El Registro de Control de Interrupciones INTCON.

La mayor parte de los señalizadores y bits de permiso de las fuentes de interrupción en los PIC16X84 están implementados sobre los bits del registro INTCON, que ocupa la dirección 0B H del banco 0, hallándose duplicado en el banco 1. El significado de cada bit, que se muestra en la figura 17, es el siguiente:

GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Figura 17

GIE: Permiso Global de Interrupciones
0: Prohíbe todas las interrupciones.
1: Permite la ejecución de todas las interrupciones, cuyos bits de permiso individuales también las permitan.

EEIE: Permiso de Interrupción por fin de escritura en la EEPROM
0: Prohíbe que se produzca esta interrupción.

1: Permite que se origine esta interrupción cuando termina la escritura en la EEPROM de datos.

T0IE: Permiso de Interrupción por sobrepasamiento del TMR0
0: Prohíbe esta interrupción.
1: Permite una interrupción al desbordarse el TMR0.

INTE: Permiso de Interrupción por activación de la patilla RB0/INT
0: Prohíbe esta interrupción.
1: Permite la interrupción al activarse RB0/INT.

RBIE: Permiso de Interrupción por cambio de estado en RB7-RB4
0: Prohíbe esta interrupción
1: Permite esta interrupción.

T0IF: Señalizador de sobrepasamiento del TMR0
0: Indica que el TMR0 no se ha desbordado.
1: Toma este valor cuando ha ocurrido el desbordamiento.

INTF: Señalizador de activación de la patilla RB0/INT
0: Indica que RB0/INT aún no se ha activado.
1: Se pone a 1 al activarse RB0/INT.

RBIF: Señalizador de cambio de estado en las patillas RB7-RB4
0: No ha cambiado el estado de RB7-RB4.
1: Pasa a 1 cuando cambia el estado de alguna de esas líneas.

Siempre que se produzca una interrupción por cualquier causa, GIE=1 y el PC se carga con el valor 0004 H, que es el Vector de Interrupción. Para conocer que causa ha provocado la interrupción se exploran los señalizadores, tres de los cuales se ubican en el registro INTCON y el cuarto, EEIF, que se pone a 1 cuando finaliza la escritura en le EEPROM, se halla en el bit 4 del registro EECON1. Los señalizadores deben ponerse a 0 por programa antes del retorno de la interrupción y son operativos aunque la interrupción esté prohíbida con su bit de permiso correspondiente.

4.3. Interrupción Externa INT.

Esta fuente de interrupciones es sumamente importante para atender los acontecimientos externos en tiempo real. Cuando ocurre alguno de ellos activa la patilla RB0/INT y se hace una petición de interrupción. Entonces, de forma automática, el bit INTF=1 y, si el bit de permiso INTE=1 se autoriza el desarrollo de la interrupción.

Mediante el bit 6, llamado INTDEG, del registro OPTION se puede seleccionar cual sereá el flanco activo en RB0/INT. Si se desea que sea ascendente se escribe un 1 en dicho bit, y si se desea descendente se escribe un 0. El procesador explora el señalizador INTF al final del primer ciclo de reloj de cada ciclo de instrucción. Recuérdese que cada ciclo de instrucción constaba de 4 ciclos de reloj: Q1, Q2, Q3 y Q4. Al terminar Q1 se exploran los señalizadores produciéndose un período de latencia de 3 ó 4 ciclos de instrucción desde el momento que hay un señalizador activado hasta que se inicializa la interrupción.

4.4.- Interrupción por desbordamiento del TMR0.

Cuando el TMR0 se desborda y pasa del valor FF H al 00 H, el señalizador T0IF se pone automáticamente a 1. Sí además, el bit de permiso de interrupción del TMR0 T0IE=1 y el bit de Permiso Global de Interrupciones GIE=1, se produce una interrupción.