TERCER CURSO. INFORMÁTICA INDUSTRIAL II

Escuela Politécnica Superior
Universidad de Huelva

Interrupciones y Temporizadores

Manuel Sánchez Raya
Versión 1.0
30 de Noviembre de 2003

ÍNDICE

BIBLIOGRAFÍA:

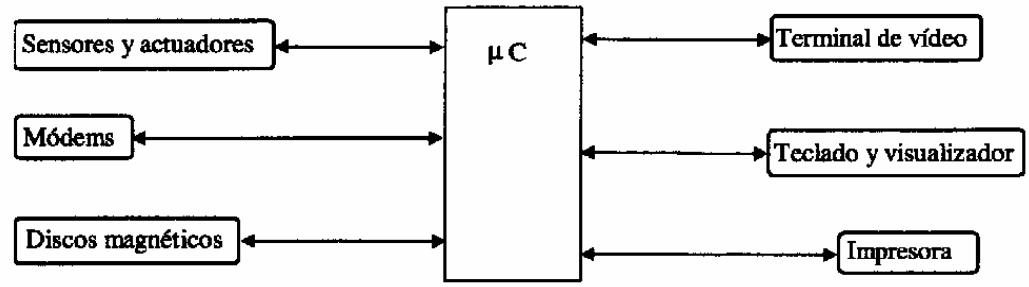
Apuntes de Ingeniería Técnica Industrial. Universidad de Málaga.

Microcontroladores MCS-51 y MCS-251. José Matas Alcalá, Rafael Ramón Ramos Lara

1.- Introducción.

Las interrupciones juegan un papel de suma importancia dentro de cualquier sistema basado en microprocesador o microcontrolador, pues estos deben habitualmente gestionar y controlar distintos periféricos asociados que, de forma continua, requieren la dedicación de la CPU para llevar a buen término las tareas que tienen asignadas.

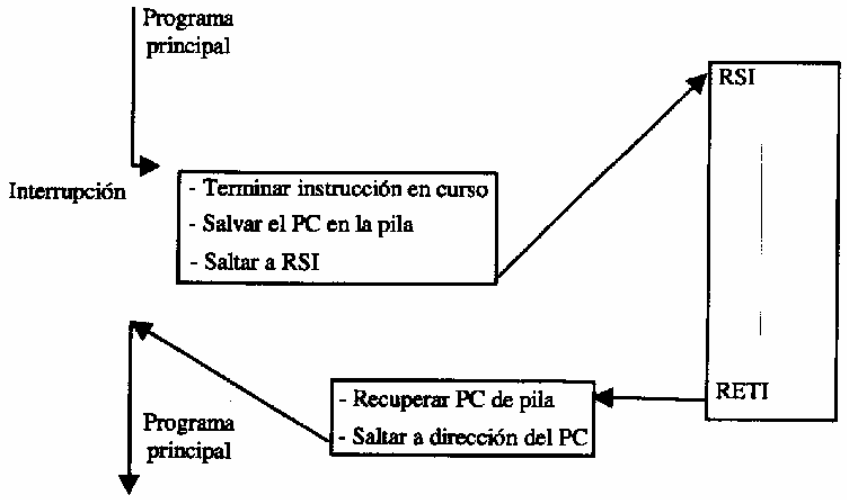
Una interrupción la realiza de forma ASÍNCRONA un periférico o un dispositivo conectado físicamente al microcontrolador, cuando requiere a la CPU el desvío del flujo de ejecución del programa para gestionar y controlar los diversos sucesos que no se encuentran bajo su supervisión directa. De esta manera se mejora la eficiencia de la CPU, ya que ésta no tiene que estar continuamente pendiente de si acontece o no un suceso en un instante de tiempo determinado, y puede realizar otras tareas de mayor interés, atendiendo a los sucesos tan sólo cuando éstos se producen. Los sucesos acontecidos pueden ser externos al sistema, como la activación de un nivel lógico o un flanco en un terminal del microcontrolador, por parte de un periférico, o bien internos, como el desbordamiento de un temporizador interno del microcontrolador al llegar éste a su máxima capacidad de cuenta.



Cuando se produce una interrupción el microcontrolador ejecuta un proceso de atención a la interrupción. En este proceso la CPU deja de ejecutar la secuencia de instrucciones en la que se encuentra y pasa a ejecutar la **rutina de servicio a la interrupción** (RSI), que se encarga de efectuar la gestión del periférico. Una vez terminada esta rutina, la CPU regresa a la secuencia donde se produjo la interrupción, y sigue con el rumbo que tenía.

En el proceso de atención a la interrupción se realizan los siguientes pasos:

1. **Termina** de ejecutar la instrucción en curso.
2. Salva el valor del contador de programa, **PC, en la pila**, de manera que la CPU, al terminar el proceso, pueda seguir ejecutando el programa a partir de la instrucción siguiente a la última ejecutada.
3. La **CPU salta a la dirección** donde está almacenada la rutina de servicio de interrupción (RSI) y ejecuta esta rutina, que tiene como finalidad atender al periférico o dispositivo que ha generado la interrupción.
4. La rutina RSI debe terminar con una instrucción de **retorno de interrupción**, RETI. La CPU al ejecutar esta instrucción lee la dirección almacenada en la pila y la asigna al contador de programa, de manera que la CPU reanuda la ejecución del programa a partir de la instrucción siguiente a la instrucción donde se produjo la interrupción.



A la dirección de salto a partir de la cual se almacena la rutina de RSI se la denomina **vector de interrupción**. Según el tipo de microcontrolador o microprocesador, las direcciones del vector de interrupción pueden ser fijas, es decir, especificadas por el fabricante, o bien pueden ser definidas por el programador. Los vectores de interrupción de la familia MCS-51 son fijos y su valor viene predeterminado por el fabricante. El microcontrolador PIC solo dispone de un vector de interrupción desde el que controlar todos los periféricos que pueden provocar una interrupción, así que dentro de la RSI se tendrá que comprobar qué periférico generó la interrupción.

Otro factor importante que se debe considerar en el proceso de interrupciones consiste en la habilitación de máscaras y en el establecimiento de prioridades. Una máscara no es más que un indicador de tipo bit que gobierna el estado de una puerta de transmisión conectada entre la entrada en interrupción y la CPU. Al bit que realiza la función de máscara se le denomina **bit de habilitación de interrupción**. La activación de un bit de habilitación de interrupción permite que la interrupción pase de la línea de entrada hacia la CPU, mientras que la desactivación de este bit hace que la interrupción no pueda pasar hacia la CPU, e inhiba la interrupción; es decir, la interrupción no es atendida a menos que su bit de habilitación correspondiente esté activado. Los bits de habilitación de interrupciones permiten, pues, que el programador pueda activar o desactivar ciertas interrupciones, en las circunstancias que sean de su consideración.

Generalmente todas las entradas de interrupción suelen tener un bit de habilitación de interrupción asociado; no obstante, puede haber alguna entrada de interrupción que adolezca de este bit, lo que se denomina **interrupción no enmascarable**, en cuyo caso el programador no tiene control sobre la interrupción y la CPU está obligada siempre a atenderla. Las interrupciones no enmascarables se reservan para aquellos periféricos o sucesos de suma importancia, como, por ejemplo, la caída de tensión de la red eléctrica, en que la CPU dispone de los pocos milisegundos en los que los condensadores de la fuente de alimentación son capaces de sostener la tensión de alimentación para salvar los registros que se consideren imprescindibles en una memoria estable, como una memoria E2PROM o una memoria RAM con una batería externa.

El **establecimiento de prioridades** dentro del proceso de interrupción se considera cuando existe la posibilidad de que varias interrupciones se produzcan de manera simultánea, por lo que es necesario estipular un orden de atención a las interrupciones producidas. Para ello, las entradas de interrupción suelen tener uno o varios bits asociados con los que el programador puede establecer un orden de prioridad en la atención de las interrupciones. La familia MCS-51 dispone de prioridades. En el PIC las prioridades se asignan mediante el orden de comprobación de periféricos dentro de la única RSI de que dispone.

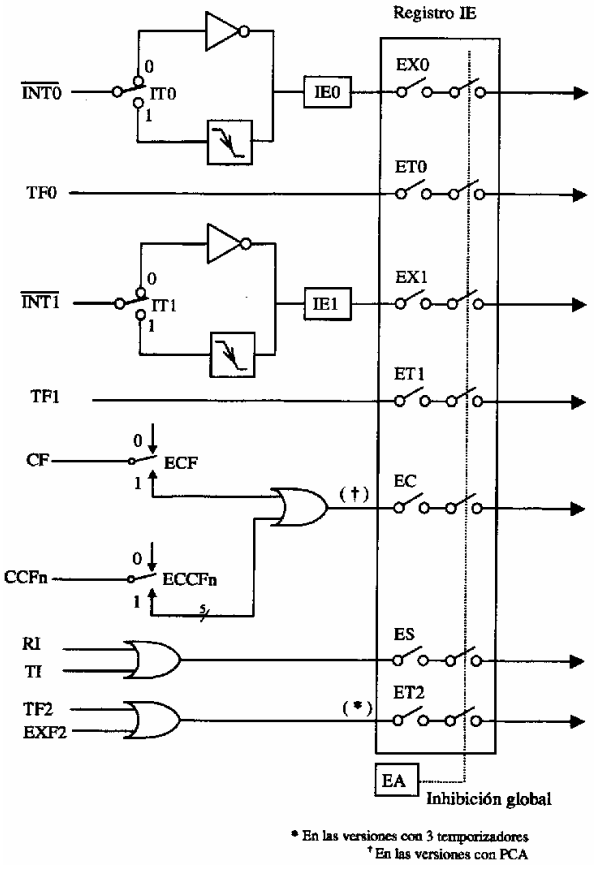
2.- Las interrupciones en la familia MCS-51.

La familia MCS-51 puede llegar a tener hasta siete fuentes de interrupción distintas, todas ellas con un bit de habilitación de interrupción situado en el registro IE, **Interrupt Enable** (dirección 0x0A8 del SFR), de forma que el programador puede habilitarlas o deshabilitarlas cuando sea necesario.

Respecto a las prioridades, todos los componentes de la familia tienen dos niveles de prioridad que se determinan con un Único bit, para cada entrada de interrupción, situado en el registro IP, **Interrupt Priority** (dirección 0x0B8 del SFR), con excepción de las versiones 8XC51Fx y 8XX52/54/58, que tienen hasta cuatro niveles de prioridad; incorporan, para ello, un segundo registro de prioridad denominado IPH, **Interrupt Priority High** (dirección 0x0B7 del SFR). En este último caso, el nivel de prioridad de una entrada de interrupción se determina mediante sus bits correspondientes de los registros IP e IPH.

Tres de las seis fuentes de interrupción son externas al microcontrolador: /INT0, /INT1 y el puerto serie, RI y TI; las fuentes de interrupción restantes son internas y corresponden a los tres temporizadores, Timer 0, Timer 1 y Timer 2, y al **array de contadores programable** (PCA) de la familia MCS-51. La interrupción de los temporizadores se produce por un desbordamiento en su valor máximo; entonces se activan los bits de desbordamiento correspondientes: TF0, TF1 o TF2. La interrupción de la PCA (solo disponible en algunas versiones) se genera mediante cualquiera de los bits EF, CCF0, CCF1, CCF2, CCM, CCF4 y CCF5 del registro CCON, PCA **Counter Control Register**; cada uno de estos bits de interrupción tiene asociado un bit de habilitación de interrupción, bits ECF y ECCFn de los registros CMOD, **PCA Counter Mode Register** y CCAPMn, **PCA Compare/Capture Modules**.

El Timer 2 tiene un bit adicional de interrupción, EXF2, que realiza una petición de interrupción si se detecta un flanco de bajada en el terminal T2 del microcontrolador.



Al producirse una interrupción en una de las entradas externas de interrupción, /INT0 o /INT1, ya sea por **nivel lógico** o por **flanco descendente**, el bit correspondiente, IE0 o IE1 del registro TCON, se activa a uno lógico, indicando de esta manera que se ha realizado una petición de interrupción, y la CPU pone en marcha el proceso de atención a la interrupción, siempre y cuando la interrupción esté habilitada de antemano. En este proceso, los bits IE0 e IE1 se ponen a cero lógico de forma automática cuando la interrupción se ha activado por flanco descendente.

Sin embargo, si la interrupción se ha activado por nivel lógico, los bits IE0 y IE1 se deben borrar por software dentro de la rutina de RSI.

El estado de las interrupciones externas /INT0 e /INT1 se comprueba una vez cada ciclo máquina, de forma que la entrada de interrupción, para el caso de que esté activada por flanco descendente, se debe sostener a nivel lógico alto al menos un ciclo máquina, y sostener a nivel lógico bajo al menos otro ciclo máquina más, para que el flanco descendente sea detectado y se active el bit IE0 o IE1, correspondiente. En el caso de que la interrupción esté activada por nivel lógico, la entrada de interrupción se debe sostener a nivel lógico bajo al menos durante 1 ciclo máquina, para que la interrupción sea detectada por la CPU.

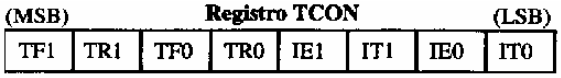
En las interrupciones internas producidas por los temporizadores Timer 0 o Timer 1, se activan los bits TF0 o TF1, según corresponda, del registro TCON a uno lógico, al realizar la petición de interrupción. Estos bits se mantienen en este estado hasta que la CPU atiende la petición, momento en el cual los pone a cero lógico de forma automática.

El temporizador Timer 2 puede producir una interrupción a través de la activación a uno lógico de los bits TF2 o EXF2. **Estos bits no se ponen a cero lógico de manera automática**, sino que los debe poner el programador por software.

Las interrupciones producidas por el puerto serie activan a uno lógico los bits TI o RI del registro SCON, cuando el microcontrolador transmite un dato o cuando recibe un dato, respectivamente. **Estos bits los debe poner a cero lógico el programador** mediante software.

La tabla siguiente muestra los bits de petición de interrupción asociados a cada una de las fuentes de interrupción.

Fuente de Interrupción	Bit que activa	Borrado por hardware	Registro	Vector de salto
/INT0	IE0	No, por nivel. Si, por flanco	TCON	0003H
Timer 0	TF0	Si	TCON	000BH
/INT1	IE1	No, por nivel. Si, por flanco	TCON	0013H
Timer 1	TF1	Si	TCON	001BH
Puerto Serie	RI, TI	No	SCON	0023H
Timer 2	TF2, EXF2	No	T2CON	002BH
PCA	CF, CCFn	No	CCON	0033H



Bit	Comentario
TF1	Bit de rebosamiento del Timer 1. Se pone a 1 por hardware en el rebosamiento. La CPU lo pone a 0 cuando procesa la interrupción y salta a la rutina de RSI.
TR1	Bit de marcha/paro del Timer 1. Se pone a 1 o 0 por software para poner en marcha o parar el Timer 1.
TF0	Bit de rebosamiento del Timer 0. Se pone a 1 por hardware en el rebosamiento. La CPU lo pone a 0 cuando procesa la interrupción y salta a la rutina de RSI.
TR0	Bit de marcha/paro del Timer 0. Se pone a 1 o 0 por software para poner en marcha o parar el Timer 0.
IE1	Bit de interrupción de la patilla /INT1. Se pone a 1 en una petición de interrupción, se pone a 0 por hardware si /INT1 está activada por flanco descendente.
IT1	Bit de selección de interrupción por nivel o por flanco descendente de /INT1. A 0 la interrupción se activa por nivel, a 1 se activa por flanco descendente.
IE0	Bit de interrupción de la patilla /INT0. Se pone a 1 en una petición de interrupción, se pone a 0 por hardware si /INT0 está activada por flanco descendente.
IT0	Bit de selección de interrupción por nivel o por flanco descendente de /INT0. A 0 la interrupción se activa por nivel, a 1 se activa por flanco descendente.

2.1.- Vectorización de interrupciones en la MCS-51.

Cuando se produce una interrupción la CPU desvía el flujo de ejecución de instrucciones hacia las rutinas de atención de interrupciones, RSI, definidas por el programador. En la MCS-51 las direcciones de salto hacia las rutinas de RSI son fijas y están predefinidas por el fabricante en lo que se denomina tabla de vectores de salto. En esta tabla, se observa cómo la interrupción /INT0 provoca un salto a la dirección 0x03 de la memoria de programas, la TIMER0 a la 0x0B, la /INT1 a la 0x13, etc. A partir de estas direcciones debe situarse la rutina de RSI que corresponda.

En la tabla también se observa que el espacio existente entre los vectores de interrupción es de tan sólo 8 bytes. Este espacio suele ser insuficiente para albergar una rutina de RSI, por lo que, en estas posiciones, se suele insertar una instrucción de salto, LJMP, AJMP o SJMP, hacia la rutina de RSI que atienda a la interrupción; así, se puede situar esta rutina en cualquier zona del espacio de memoria disponible por el microcontrolador.

El retorno de una rutina de RSI se debe realizar con la instrucción RETI, para que la CPU diferencie este retorno, del retorno de subrutina (instrucción RET).

```
;*********************************************************
; Vectorización de interrupciones
;*********************************************************
            ORG   03H          ;Posiciona instrucción siguiente en 03H
            LJMP  RSI_INT0     ;Salto a la rutina de RSI
            ORG   OBH          ;Posiciona instrucción siguiente en OBH
            LJMP  RSI-TIMER0   ;Salto a la rutina de RSI
            ORG   013H         ;Posiciona instrucción siguiente en 013H
            LJMP  RSI-INT1     ;Salto a la rutina de RSI
RSI_INT0:   MOV ....           ;Rutina de RSI para /INT0
            .
            .
```

```
            RETI
RSI_TIMER0; MOV .....         ;Rutina de RSI para TIMER0
            .
            .
            RETI
RSI_INT1:   MOV .....         ;Rutina de RSI para INT1
            .
            .
            RETI
```

La estructura del programa cuando se utilizan interrupciones estará formada por una rutina de vectorización, como la mostrada en este apartado. El registro PC se inicializa a 0x0000 tras un reset o la puesta en marcha del microcontrolador, por lo que la primera instrucción que se ejecuta es la contenida en la dirección 0x0000. El linker colocará en esta dirección un salto al **módulo STARTUP** para que inicialice la memoria y el entorno de ejecución del programa C y luego salte a la función main.

El compilador de C contiene una extensión del lenguaje C que nos permite definir una interrupción (RSI) como si se tratase de una función de C, ocupándose el compilador de los detalles de llamada a la RSI y vuelta de interrupción. Empleando la palabra reservada "**interrupt**" es posible marcar una determinada función como la RSI de atención a la interrupción situada en la posición de memoria n*8+3. n es el número colocado a continuación de la palabra reservada. Por lo que, un programa típico tendrá la siguiente estructura:

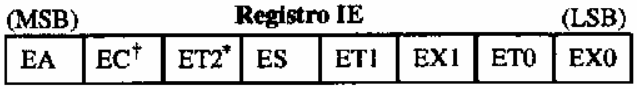
```
/*********************************************************
 Vectorización de interrupciones
*********************************************************/


      void rsi_int0(void) interrupt 0 {
      …
      … // atención a la interrupción situada en el vector n*8+3
      … // donde n es el número detrás de "interrupt"
      }

      void rsi_timer0(void) interrupt 1 {
      …
      …
      }

      void rsi_int1(void) interrupt 2 {
      …
      …
      }

      void main(void) {
      …
      … // programa principal
      …
      }
```

2.2.- Habilitación de interrupciones y establecimiento de prioridades en la MCS51.

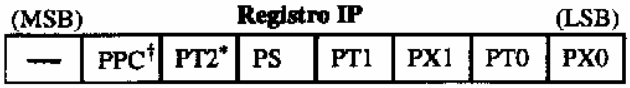
Cada una de las fuentes de interrupción de la MCS-51 dispone de un bit de habilitación/deshabilitación en el registro IE, **Interrupt Enable**, del área de SFR. En el registro IE el bit de mayor peso, IE.7 o también EA, es el bit de inhibición global y afecta de forma directa a todos los bits de habilitación de interrupciones establecidos dentro del mismo registro IE. Poniendo el bit EA a cero lógico se inhabilitan todas las interrupciones del microcontrolador, aunque existan bits de habilitación de interrupción activados; por contra, poniendo el bit EA a uno lógico, se habilitan sólo las interrupciones que tienen su bit de habilitación activado.

La mayor parte de los componentes de la MCS-51 tienen dos niveles de prioridad, con excepción de las versiones 8XC51Fx y 8XC52/54/58, que tienen hasta cuatro niveles de prioridad. El nivel de prioridad se determina con el registro de prioridades IP, **Interrupt Priority**, del área de SFR. Poniendo un bit de este registro a uno lógico se fija la prioridad a nivel alto y a cero lógico se fija a nivel bajo.

Para el caso de que dos o más interrupciones tengan el mismo nivel de prioridad y se produzca una petición simultánea de interrupción, el fabricante asigna un nivel de prioridad por defecto a cada una de las fuentes de interrupción del microcontrolador, de forma que en una petición simultánea de interrupción, la CPU atienda primero a la interrupción con mayor prioridad según la tabla siguiente.



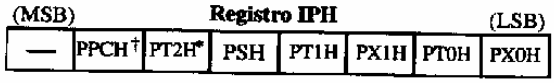
Bit	Comentario
EX0	EX0=1 habilita la interrupción en INT0 EX0=0 la inhabilita
ET0	ET0=1 habilita la interrupción del timer 0. ET0=0 la inhabilita
EX1	EX1=1 habilita la interrupción en INT1 EX1=0 la inhabilita
ET1	ET1=1 habilita la interrupción del timer 1. ET1=0 la inhabilita
ES	ES=1 habilita la interrupción del puerto serie. ES=0 la inhabilita
ET2	ET2=1 habilita la interrupción del Timer 2 ET2=0 la inhabilita
EC	EC=1 habilita la interrupción de la PCA EC=0 la inhabilita
EA	EA=1 permite todas las habilitaciones o inhabilitaciones anteriores EA=0 no reconoce ninguna interrupción



Bit	Comentario
PX0	Bit de prioridad de /INT0
PT0	Bit de prioridad del Timer 0
PX1	Bit de prioridad de /INT1
PT1	Bit de prioridad del Timer 1
PS	Bit de prioridad del puerto serie
PT2	Bit de prioridad del Timer 2
PPC	Bit de prioridad de la PCA
-	Bit reservado

Las versiones 8XC51Fx y 8XC52/54/58 tienen cuatro niveles de prioridad que se determinan por medio de los registros IP y IPH. En la tabla siguiente se indica el nivel de prioridad según el estado de los bits correspondientes a cada una de las fuentes de interrupción.

Prioridad	Fuente
(más alta) 1	/INT0
2	Timer 0
3	/INT1
4	Timer 1
5	PCA
6	Puerto Serie
(más baja) 7	Timer 2



Bit	Comentario
PX0H	Bit alto de prioridad de /INT0
PT0H	Bit alto de prioridad del Timer 0
PX1H	Bit alto de prioridad de /INT1
PT1H	Bit alto de prioridad del Timer 1
PSH	Bit alto de prioridad del puerto serie
PT2H	Bit alto de prioridad del Timer 2
PPCH	Bit alto de prioridad de la PCA
-	Bit reservado

Bits de Prioridad		Nivel de prioridad
IPH.x	IP.x	
0	0	Nivel 0 (Menor)
0	1	Nivel 1
1	0	Nivel 2
1	1	Nivel 3 (Mayor)

2.3.- Tiempos de respuesta del proceso de interrupción.

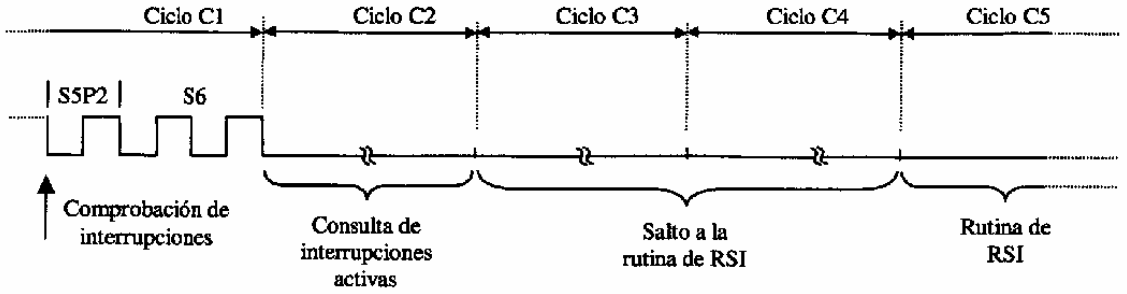
Las fuentes de interrupción en la MCS-51 se comprueban en la fase 2 del estado 5 de cada ciclo máquina, denominado S5P2, y se evalúa su estado en el siguiente ciclo máquina del microcontrolador. En consecuencia, si la CPU encuentra que una de las fuentes está activa, entonces genera un salto automático hacia la rutina de RSI correspondiente. Aunque, este salto puede ser abortado por cualquiera de las condiciones siguientes:

1. Se halla en proceso una interrupción de mayor o igual prioridad.
2. El ciclo máquina en el que se ha producido la interrupción no es el último de la instrucción en curso de ejecución por parte de la CPU, por lo que debe esperar a que se termine de ejecutar la instrucción.
3. La instrucción actual en curso es una instrucción RETI o cualquier otra que escriba en los registros IE o IP.

La lectura de las peticiones de interrupción realizadas por el microcontrolador carece de memoria, por lo que si una petición de interrupción resulta abortada por una de las condiciones anteriores, en un ciclo máquina determinado, y esta petición no se sostiene por parte del

periférico de manera que vuelva a detectarse en el ciclo maquina siguiente, entonces la interrupción no es atendida; es decir, en caso de rechazo de la interrupción, ésta debe ser sostenida por el periférico el tiempo necesario hasta que la CPU la atienda.

La CPU reconoce la petición de una interrupción cuando vectoriza la interrupción, es decir, cuando salta a la rutina de RSI correspondiente. Según el tipo de interrupción, la CPU, además, borra el bit activado en la petición de interrupción (tabla 6. l), como es el caso de los bits TF0 y TF1, o de los bits IE0 e IE1, sólo cuando la interrupción ha sido activada por flanco descendente. En otros casos el bit activado por la interrupción se debe borrar por software.

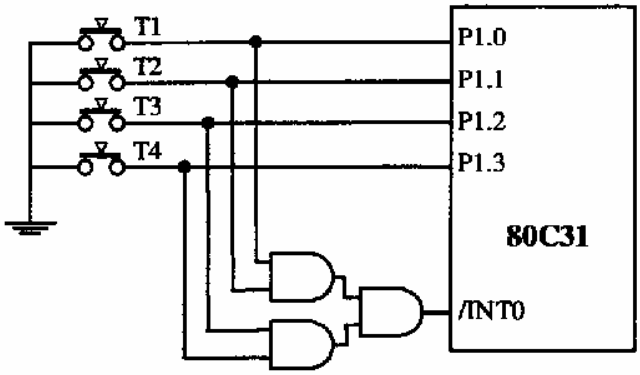


La CPU guarda en la memoria interna de la pila el contenido del registro PC (dos bytes) y, luego, salta a la rutina RSI correspondiente. La CPU ejecuta la rutina de RSI hasta que encuentra una instrucción RETI, que le indica que se ha llegado al final del proceso de interrupción; recupera entonces los dos bytes almacenados en la pila y los asigna al registro PC, por lo que la ejecución del programa regresa al punto de partida donde se inició la interrupción.

El tiempo transcurrido, desde que se produce la interrupción (S5P2 del ciclo C1), hasta que la CPU comienza a ejecutar la rutina de RSI (ciclo C5), es de **como mínimo tres ciclos máquina**. El tiempo de respuesta es mayor si el salto resulta abortado por una de las condiciones mencionadas. En el primer caso, si una interrupción de mayor o igual prioridad está en proceso, el tiempo de espera dependerá de la rutina de RSI en ejecución. En el segundo caso, cuando la CPU debe esperar a que se finalice la ejecución de la instrucción actual, el tiempo de espera, en el peor caso, no puede ser superior a tres ciclos máquina. En el tercer caso, si una instrucción RETI, o la escritura del registro IE o IP, está en progreso, el tiempo adicional de espera no puede ser superior a cinco ciclos máquina. En definitiva, **el tiempo de respuesta de una interrupción estará comprendido entre tres y nueve ciclos máquina**.

Ejemplo 1: Conexión de teclas al microcontrolador

Se conectan cuatro teclas al puerto P1 de un microcontrolador 80C31. Las entradas del puerto P1 tienen una resistencia interna de pull-up, por lo que un terminal del puerto, por ejemplo P1.0, estará en el estado 1 lógico (Vcc) cuando no se pulsa la tecla T1, y a 0 lógico (masa) cuando se pulsa T1. Luego, la detección de si una tecla ha sido pulsada o no, consiste en leer el estado lógico del puerto P1, y ver si hay algún cero en una de sus patillas. En la figura cada una de las teclas está conectada a una puerta AND, de manera que la pulsación de cualquiera de las cuatro teclas causará un 0 lógico en la entrada de interrupción /INT0, y provocará una interrupción en el caso de que esté habilitada.



En este ejemplo se debe habilitar la interrupción /INT0 y crear una rutina RSI capaz de detectar la tecla que ha sido pulsada. Para ello, se utilizará la variable "tecla" como indicador de la pulsación de una tecla, y se le asignará el valor 0x00 cuando no se haya pulsado ninguna tecla, y los valores 0x0l, 0x02, 0x03 y 0x04, cuando se pulsen las teclas T1, T2, T3 y T4, respectivamente. La interrupción /INT0, en este ejemplo, se establecerá por nivel lógico (bit IT0 del registro TCON a 0 lógico). Se debe tener en cuenta que todos los bits accesibles de los registros TCON, IE e IP quedan a 0 lógico tras un reset del microcontrolador.

```
#include <reg51.h>
/*********************************************************
 Rutina de vectorización (ejemplo 1)
*********************************************************/
//Definición de entradas
sbit Tecla_1=P1^0;
sbit Tecla_2=P1^1;
sbit Tecla_3=P1^2;
sbit Tecla_4=P1^3;

//Variables globales
unsigned char tecla;

/*********************************************************
 Rutina de RSI de /INT0
*********************************************************/
void RSI_Int0(void) interrupt 0 {
      if (!Tecla_1)           tecla=0x01;
      else if (!Tecla_2)      tecla=0x02;
      else if (!Tecla_3)      tecla=0x03;
      else                    tecla=0x04;
      IE0=0;            //Se pone a cero para que /INT0
//Final de interrupción
}
/*********************************************************
 Rutina de inicio.
 (habilita interrupciones y /INT0 flanco descendente)
*********************************************************/
```

```
void inicio(void) {
      PX0=1;  // Prioridad alta para /INT0 (Registro E)
      EX0=1;  // Habilitación de /INT0 (Registro E)
      EA =1;  // Habilitación general (Registro E)
}
/*******************************************************
 Programa Principal
*******************************************************/
void main(void) {

inicio();
//Bucle infinito sin propósito definido
while(1) {
 }
}
```

La ejecución de las instrucciones del programa empieza por la función main, pues, en la puesta en marcha del microcontrolador, se ejecuta un reset interno que sitúa el registro PC a 0x00. La primera instrucción, entonces, es una instrucción de salto a la función STARTUP que a su vez llama a la función "main", donde se configura la forma de operar de la interrupción /INT0, mediante los registros TCON, IE y IP. A continuación, la CPU pasa a ejecutar el bucle principal, que sólo consiste en un bucle infinito, sin ningún objetivo específico. Esto es así, por el funcionamiento casi exclusivo del programa mediante interrupciones, de manera que la CPU está siempre ejecutando instrucciones de la función "main", a la espera de que las interrupciones se produzcan, lo que causa el salto automático hacia las rutinas de RSI.

En el momento que se pulsa una tecla, se provoca la interrupción /INT0 y se ejecuta la rutina "RSI_Int0". Esta rutina comprueba de forma secuencial, cuál ha sido la tecla pulsada, colocando en la variable "tecla" el valor adecuado, según los objetivos marcados en este ejemplo. Al final de la rutina, el bit IE0, debido a que la interrupción /INT0 se ha habilitado por nivel lógico, se borrará por software para que se produzca una nueva interrupción en /INT0.

La lectura de las teclas, en el programa realizado, se hace de manera secuencial, por lo que el programa es incapaz de detectar una pulsación simultánea de varias teclas, y da por válida la tecla que lee primero; es decir, en el caso de pulsarse las teclas conectadas TI y T2, en el registro B se colocada el código correspondiente a la tecla TI. Este problema se solventa en el siguiente ejemplo que consiste en una modificación del actual.

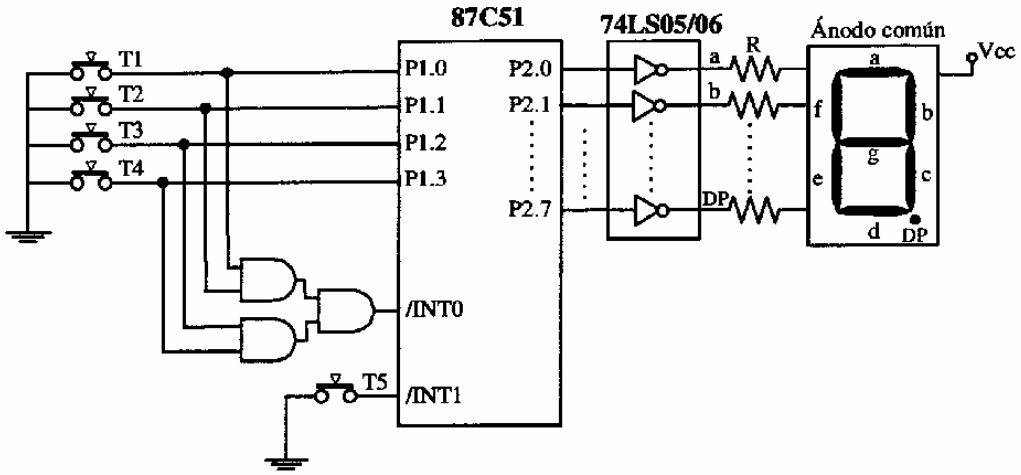
Ejemplo 2: Conexión de teclas y de un dígito de siete segmentos

Este ejemplo se basa en el anterior, con las siguientes modificaciones:

- Se incorporan un dígito de siete segmentos conectado al puerto P2 y una tecla adicional, T5, conectada a la entrada de interrupción /INT1.
- El microcontrolador empleado es el 87C51 y utiliza exclusivamente su memoria de programa interna.
- El programa debe ser capaz de detectar la pulsación de varias teclas al mismo tiempo, e indica esta situación a través del encendido de todos los leds del dígito conectado.
- La pulsación de una tecla, T1, T2, T3 o T4, se indicará poniendo el número 1, 2, 3 o 4, en el dígito, respectivamente. En el dígito siempre se mostrará la última tecla pulsada.
- La pulsación de T5 apagará todos los leds del dígito T5.

En la figura siguiente se emplea el circuito integrado 7405 que contiene hasta 6 puertas inversoras en colector abierto. La puerta inversora está formada por un único transistor, del cual se dispone de su colector, tal y como se ve en el detalle de la figura. La puerta inversora puede

soportar una comente máxima de 25mA, valor más que suficiente para encender de manera adecuada el diodo LED que tiene asociado.



La interrupción /INT1 se habilita por flanco descendente, mientras que la interrupción /INT0 se habilita por nivel lógico, de manera que en la rutina RSI de /INT0 se debe borrar el bit IE0. Sin embargo, en la rutina RSI de /INT1 el bit IE1 se borra de forma automática.

```
#include <reg51.h>
/*************************************************************
 Rutina de vectorización (ejemplo 2)
*************************************************************/
unsigned char tecla;
/*************************************************************
 Rutina de RSI de /INT0
*************************************************************/
void RSI_Int0(void) interrupt 0 {
if ((P1|0xF0)==0xFE) tecla=0x01;
else if ((P1|0xF0)==0xFD) tecla=0x02;
else if ((P1|0xF0)==0xFB) tecla=0x03;
else if ((P1|0xF0)==0xF7) tecla=0x04;
// se han pulsado varias teclas
tecla=0x05;
IE0=0;
}

/*************************************************************
 Rutina de RSI de /INT1
*************************************************************/
void RSI_Int1(void) interrupt 2 {
tecla=0x00; //Borra la variable tecla
// No es necesario borrar el bit EI ya que es por flanco
}


/*************************************************************
 Rutina de inicio.
 (habilita interrupciones y /INT1 flanco descendente)
*************************************************************/
void inicio(void) {
    IT1=1;      //Interrupción /INT1 activa flanco des.
    PX1=1;      //Prioridad alta para /INT0 (Registro E)
    EX0=1;      //Habilitación de /INT0
    EX1=1;      //Habilitación de /INT1
    EA=1;       //Habilitación general (Registro E)
}
```

```
unsigned char Siete_seg[5]={0x3F,0x06,0x5B,0x4F,0x66,0x00};
/*********************************************************
 Programa Principal
*********************************************************/
void main(void) {
inicio();
while (1) {
    P2=Siete_seg[tecla];  //Codifica para mostrar el digito
    }
}
```

La función “main” de este ejemplo se encarga de leer el contenido de la variable “tecla” y de poner en el dígito el carácter correspondiente según sea el valor de “tecla”. La variable “tecla” puede valer 0x00 al principio, si no se ha pulsado ninguna tecla, o tras pulsar la tecla T5; puede valer 0x0l, 0x02, 0x03 ó 0x04, al pulsar las teclas T1, T2, T3 o T4, respectivamente; y puede valer 0x05 en el caso de que se pulsen varias teclas al mismo tiempo. Con el valor de la variable “tecla” se lee la tabla “Siete_seg”, que proporciona el código en siete segmentos, correspondiente al carácter que se quiere mostrar en el dígito del ejemplo. De todas formas, la tabla “Siete_seg” se podría haber suprimido colocando en la variable “tecla”, directamente, el código del carácter que deberá mostrar en el dígito, según sea el caso.

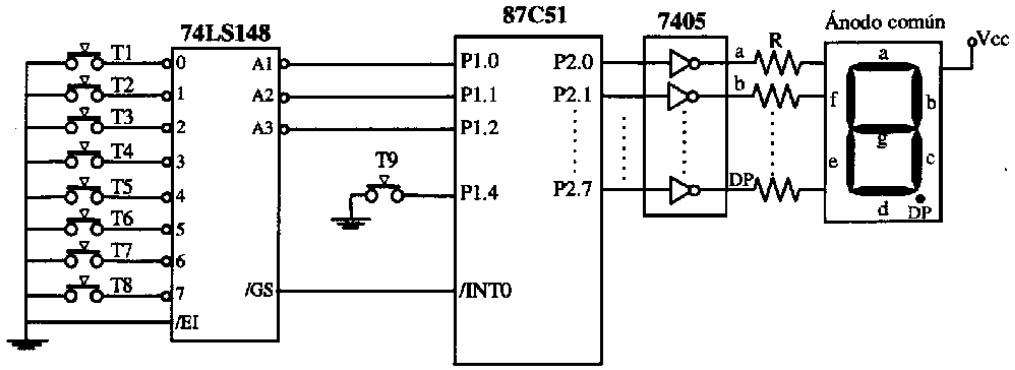
Ejemplo 3: Conexión de teclas al microcontrolador mediante el 74LS148

El 74LS148 es un circuito integrado codificador con prioridad que se puede emplear para conectar varios periféricos a una misma línea de interrupción, o bien, como en este caso, a la conexión de hasta 8 teclas al microcontrolador, empleando para ello el sistema de interrupciones.

Según el circuito de la figura, el codificador con prioridad codifica en binario la tecla pulsada, sacando el código por las tres líneas de salida: Al, A2 y A3. Con el 74LS148, si se pulsan varias teclas a la vez, aparece a su salida el código correspondiente a la entrada con mayor prioridad (la entrada 7 es la de mayor prioridad, y la entrada 0 la de menor prioridad).

Cuando se pulsa una tecla la línea de salida /GS del codificador se pone a cero lógico, de manera que puede generar una interrupción conectándola a la entrada /INT0 del microcontrolador. La entrada /EI del 74LS148 es de habilitación del circuito integrado.

/EI se conecta directamente a masa para que el codificador esté siempre habilitado.



El programa que se debe realizar en este ejemplo ha de mostrar el número de la última tecla pulsada en el dígito de siete segmentos, mientras que la pulsación de la tecla T9 debe borrar el dígito. La interrupción /INT0 se debe activar por nivel lógico.

```
#include <reg51.h>
/*********************************************************
 Rutina de vectorización (ejemplo 3)
*********************************************************/
sbit Tecla_T9=P1^4;
unsigned char tecla;
//tabla que convierte de codigo binario a 7 segmentos
unsigned char tabla_tecla[8]={0x3F,0x06,0x5B,0x4F,0x66,0x6D,
                                        0x7D,0x07};
/*********************************************************
 Rutina de RSI de /INT1
*********************************************************/
void RSI_Int1(void) interrupt 2 {
 tecla=tabla_tecla[P1&0x07]; //Máscara, P1 a 0, excepto P1.0,P1.1,P1.2
 IE0=0;                 //Borra   bit   IE0,   para   una   posterior
interrupción
}
/*********************************************************
 Rutina de inicio  (Habilitación de interrup /INT0 por nivel)
*********************************************************/
void inicio(void) {
      PX0=1;            //Prioridad alta para
      EX0=1;            //Activa bit de habilitación de /INT0
      EA=1;       //Activa el bit de habilitación general
}
/*********************************************************
 Programa principal
*********************************************************/
void main(void) {
inicio();
while(1) {
      if (Tecla_T9) P2=tecla;
      else tecla=0x00;
      }
}
```

La rutina de RSI de /INT0 lee directamente el valor del puerto P1, aplica una máscara al valor leído, puesto que sólo interesa el valor de las patillas P1.0, P1.l y P1.2 del puerto, y pone el código del carácter en siete segmentos, correspondiente a la tecla pulsada, en la variable "tecla". La función "main" comprueba el estado de la tecla T9 con la instrucción "JB P1.4, Rut_ok", y sitúa el valor de la variable "tecla", procedente de la función "RSI_INT0", en el puerto P2, para su visualización. La variable "tecla" se pone a cero cuando se pulsa la tecla T9.

2.4.- Interrupciones externas /INT0 e /INT1.

Las interrupciones /INT0 e /INT1 posibilitan el control de periféricos externos mediante el mecanismo de interrupciones.

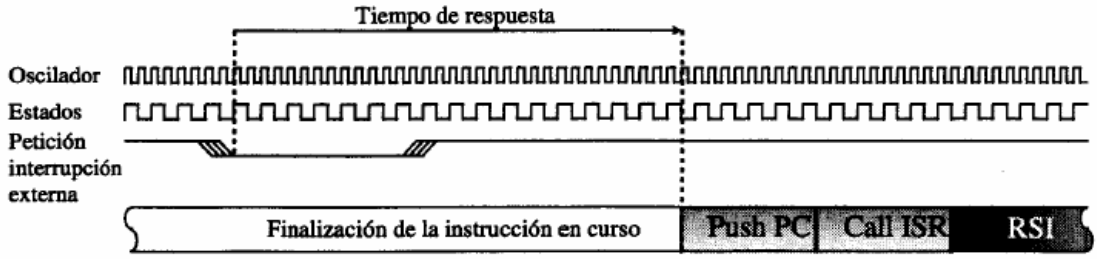
El periférico externo interrumpe al μC y activa la entrada de interrupción. El μC responderá saltando a la dirección de comienzo de la RSI asociada a la interrupción activada.

Las interrupciones externas /INT0 e /INT1 se pueden programar para ser activas por flanco descendente o por nivel bajo, dependiendo del valor de los bits IT0 e IT1 del registro TCON. Con el bit IT0 a cero lógico, la interrupción /INT0 se activa por nivel, mientras que con IT0 a uno lógico, la interrupción /INT0 se activa por flanco descendente. La misma relación se produce entre el bit IT1 y la interrupción /INT1.

Cuando se genera una interrupción externa del tipo /INT0 o /INT1, se activa el correspondiente flag de petición de interrupción, bits IE0 o IE1, del registro TCON (tabla anterior). Si la interrupción se ha activado por flanco descendente, los flags de petición de interrupción se borran por hardware cuando la CPU salta a la rutina de atención a la interrupción para proceder a su ejecución. Si la interrupción se activa por nivel, el flag de interrupción se deberá borrar mediante una instrucción dentro de la RSI.

Para este caso, la causa que ha generado la interrupción externa /INT0 o /INT1 debe desactivarse antes de que acabe la RSI o se generará una nueva petición de interrupción.

Para detectar la petición de interrupción, la CPU lee el estado de los pines asociados a las interrupciones externas /INT0 e /INT1 una vez cada ocho periodos de reloj. Tanto si la interrupción externa está programada por nivel como por flanco, será necesario mantener un nivel bajo durante al menos diez períodos para garantizar la detección de la interrupción.



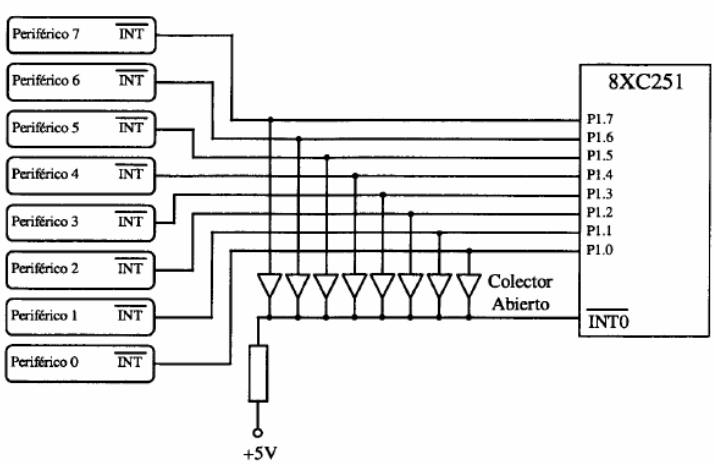
Fuente de Interrupción	Flag	Bit de configuración por flanco o nivel	Vector de interrupción
/INT0	IE0	IT0	0003H
/INT1	IE1	IT1	0013H

Si la aplicación diseñada requiere el control de más de dos periféricos externos, se deberá incluir en el diseño algún sistema, hardware y/o software, que permita determinar qué periférico o periféricos han interrumpido, y en el caso que haya interrumpido más de uno, en qué orden se deben atender.

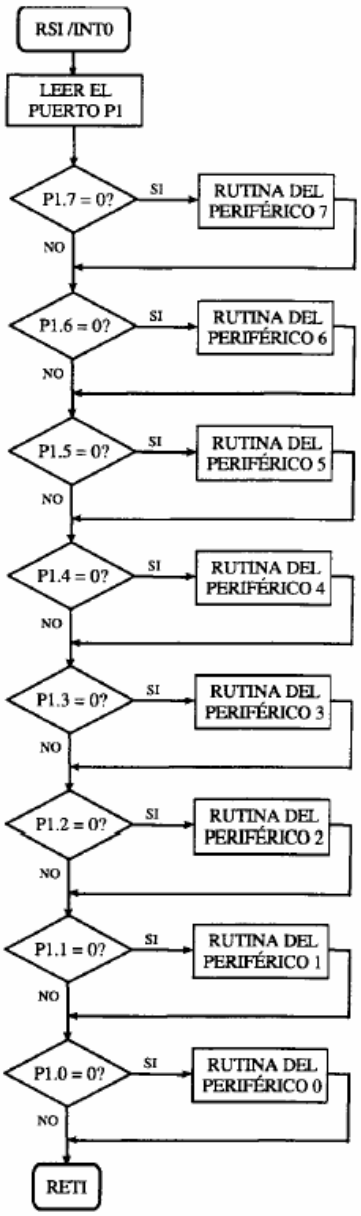
El establecimiento del orden de prioridad se puede hacer externamente, utilizando un controlador de prioridad de interrupciones (PIC), o bien, mediante software, programando en la RSI el orden de atención a los distintos periféricos.

Ejemplo 4: Control de múltiples periféricos externos mediante chequeo.

En este ejemplo se utiliza una secuencia de chequeo para determinar la prioridad de atención a ocho periféricos externos conectados a la entrada de interrupción /INT0. Las salidas de interrupción de cada periférico se han conectado a la entrada /INT0 a través de un driver de salida en colector abierto. Basta que un periférico active su salida de interrupción para que la entrada de interrupción /INT0 pase a cero lógico. El periférico que interrumpe activa al mismo tiempo un bit del puerto P1.



Cuando ocurre una interrupción, la RSI se encarga de leer el puerto P1 y averiguar por chequeo qué periférico o periféricos han interrumpido y atenderlos siguiendo un orden establecido en la propia RSI. En la figura está representado el flujograma correspondiente a este ejemplo. La prioridad de cada periférico se establece con el orden de chequeo de los bits del puerto P1. En este ejemplo, se verifica en primer lugar el bit P1.7, de forma que es el periférico 7 el más prioritario. El periférico 0 es el menos prioritario porque se verifica en último lugar.




```
#include <reg51.h>
sbit PERI1=P1^0;
sbit PERI2=P1^1;
sbit PERI3=P1^2;
sbit PERI4=P1^3;
sbit PERI5=P1^4;
sbit PERI6=P1^5;
sbit PERI7=P1^6;
/*****************************************
*
 RUTINA DE SERVICIO A LA INTERRUPCION INT0
*****************************************
/
void RSI_Int0(void) interrupt 0 {
// Si P1.7 (bit MSB) es cero salta a la
// rutina de atención del periférico 7
if (!PERI7) RSI_Per7();
else if (!PERI6) RSI_Per6();
else if (!PERI5) RSI_Per5();
else if (!PERI4) RSI_Per4();
else if (!PERI3) RSI_Per3();
else if (!PERI2) RSI_Per2();
else if (!PERI1) RSI_Per1();
}

void RSI_Per1(void) {
//...
// Rutina de atención del periférico 1
}
/* ... */
void RSI_Per7(void) {
...
// Rutina de atención del periférico 7
}
```

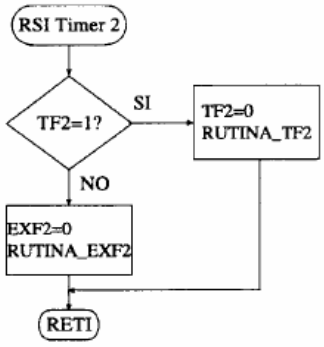
2.5.- Interrupción de los Timers.

Los flags de petición de interrupción de los Timers 0 y 1, TF0 y TF1, se activan por rebasamiento del Timer correspondiente, excepto cuando el Timer 0 está programado en modo 3, en cuyo caso se activan exclusivamente pro rebasamiento de los registros TH0 y TL0 del Timer 0. Cuando el Timer 0 ó 1 genera una interrupción, el flag de petición de interrupción se borra automáticamente cuando la CPU ejecuta la rutina de servicio a la interrupción.

El Timer 2 posee dos flags de interrupción, TF2 y EXF2, ubicados en el registro T2CON. La activación de cualquiera de estos bits genera una petición de interrupción a la CPU y su borrado debe realizarse por software. Por tanto, la rutina de servicio a la interrupción deberá determinar cuál de los dos bits ha generado la interrupción y borrarlo.

A continuación se presenta el flujograma y el listado de un ejemplo de RSI del Timer 2. Este programa incorpora las instrucciones que permiten detectar cuál de los dos flags de interrupción del Timer 2 se ha activado, y actuar en consecuencia, de forma que si el flan activo es TF2, se ejecuta la rutina RUTINA_TF2, y si el flan activo es EXF2, se ejecuta la rutina RUTINA_EXF2.

```
#include <reg52.h>
/*******************************************
 RUTINA DE SERVICIO A LA INTERRUPCION TIMER 2
*******************************************/
void RSI_Timer2(void) interrupt 5 {
//Dirección de RSI del Timer 2 es 0x2B(8*5+3)
if (TF2) {
      TF2=0;
      // Tratamiento de TF2
}
else {
      EXF2=0;
      // Tratamiento EXF2
 }
}
```

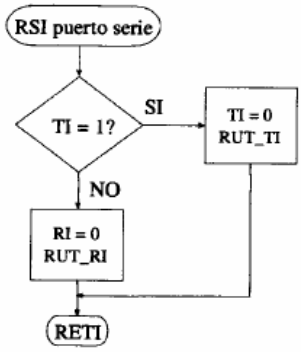


2.6.- Interrupción del puerto serie.

El puerto serie tiene asociado dos flags de interrupción: el flag de interrupción para recepción, RI, y el flag de interrupción para transmisión, TI. Ambos flags están ubicados en el registro SCON (tabla). La activación de cualquiera de estos dos flags genera una petición de interrupción a la CPU por parte del puerto serie. Tal y como ocurre con la interrupción del Timer 2 y del PCA, el borrado de los flags TI y RI debe realizarse por programa. Por tanto, en la RSI del puerto serie se deben incluir instrucciones que determinen cuál de estos flags se ha activado, y lo borren.

A continuación se representa el flujograma y el listado de un ejemplo de RSI del puerto serie. Este programa incorpora las instrucciones que permiten detectar el flag de interrupción que se ha activado y actuar en consecuencia, de forma que si el flag activo es TI, se ejecuta la parte de procesamiento de transmisión, y si el flag activo es RI, se ejecuta la otra parte de recepción.

```
#include <reg51.h>
/************************************************
RUTINA DE SERVICIO A LA INTERRUPCION DEL PUERTO SERIE
************************************************/
void RSI_Serie(void) interrupt 4 {
//La dirección de comienzo de la
//RSI del puerto serie es 0x23(8*4+3)
if (TI) {
      TI=0;
      // envio nuevo carácter
      }
else {
      RI=0;
      // recepcion nuevo carácter
      }
}
```



Ejemplo 5: Control del índice de acidez (pH) del agua de un depósito.

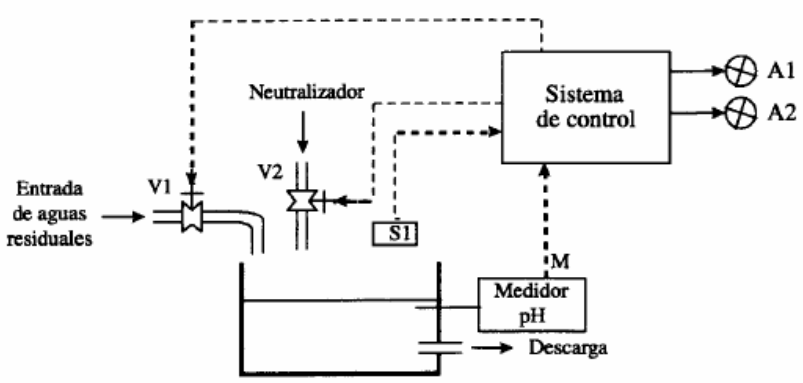
Se plantea a continuación, como ejemplo ilustrativo de la utilización del sistema de interrupciones del microcontrolador 8XC51, un caso práctico de control de una planta de tratamiento de aguas residuales. El objetivo de este ejemplo es mostrar el funcionamiento de las interrupciones por lo que respecta a la habilitación, los niveles de prioridad, el manejo de los flags de interrupción, etc.

La planta que se debe controlar dispone de un sistema que neutraliza la acidez de las aguas residuales provenientes de una planta de fabricación de papel (figura). El sistema posee un depósito donde se mezcla el agua residual con la cantidad adecuada del componente neutralizador, cuya función es disminuir la acidez del agua, de forma que el agua de salida del depósito posea un pH superior a 5.5.

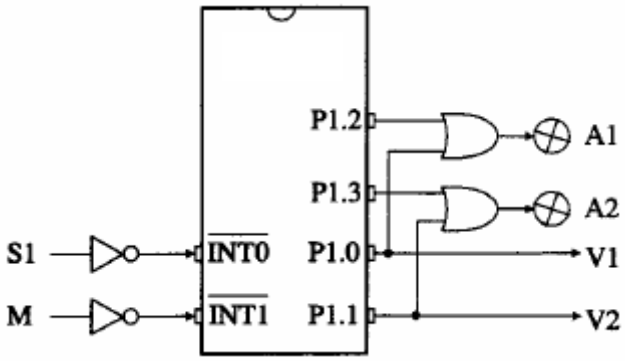
El sistema de control incorpora dos sensores activos a nivel alto:

- Un detector de rebosamiento, S1, que se activa cuando el agua contenida en el depósito supera la altura máxima permitida.
- Un detector de pH, M, que se activa cuando el pH del agua es inferior a 5.5.

El sistema de control dispone además de dos actuadores V1 y V2, activos a nivel alto, cuya función es realizar la apertura de dos válvulas. Cuando se activa el actuador V1 se abre la válvula que permite la entrada de las aguas residuales al depósito. Cuando se activa el actuador V2 se abre la válvula que permite el vertido de neutralizador en el depósito. Por último, existen dos indicadores luminosos, Al y A2, activos a nivel alto, cuya función es la de monitorizar la apertura de las válvulas V1 y V2.



En la figura siguiente está representado el esquema eléctrico donde se detalla la conexión del μC 8XC51 con los diferentes sensores y actuadores que intervienen en esta aplicación. Cabe destacar que los sensores S1 y M están conectados a las entradas de interrupción /INT0 e /INT1 a través de sendas puertas inversoras, debido a que las interrupciones externas se activan a nivel bajo, mientras que estos sensores son activos a nivel alto.



La estrategia que debe llevar a cabo el sistema de control para gestionar el funcionamiento de la planta de tratamiento de aguas residuales se puede resumir en dos puntos:

- La válvula V1 debe permanecer abierta hasta que el sensor S1 se active; entonces se cerrará durante treinta segundos. Si una vez pasado este tiempo el sensor S1 sigue activo, se repetirá la operación de cierre de la válvula V1. Mientras V1 esté abierta, el indicador Al permanecerá encendido; en caso contrario el indicador parpadeará.
- La válvula V2 debe estar cerrada hasta que se active el sensor M y entonces abrirse durante cinco segundos. Una vez transcurrido ese tiempo, si el sensor M continua activo, se repetirá la operación de apertura de la válvula V2. El indicador A2 se activa cuando la válvula V2 está abierta y parpadea cuando está cerrada.

Los recursos utilizados para resolver esta aplicación son cuatro:

- **La interrupción externa /INT0**, que se encarga de detectar la activación del sensor S1. La interrupción se activa por nivel para que pueda ser atendida por la CPU mientras el sensor S1 se encuentre a uno lógico.
- **La interrupción externa /INT1**, cuya función es detectar la activación del medidor de pH. Esta interrupción también se programa por nivel por el mismo motivo que la /INT0.
- **El Timer 0**, que se encarga de temporizar el intervalo de 30 s, tiempo que debe permanecer abierta la válvula V1 cuando se active S1.
- **El Timer 1**, que se encarga de temporizar los 5s de apertura de la válvula V2.

El programa esta compuesto de una rutina principal y de cuatro rutinas de atención a la interrupción, una por cada fuente de interrupción utilizada en la aplicación.

La función del programa principal será básicamente la de habilitar y programar los niveles de prioridad de las diversas fuentes de interrupción utilizadas, así como la de ejecutar la secuencia de parpadeo de los indicadores luminosos Al y A2.

Para habilitar las cuatro fuentes de interrupción se ponen a uno lógico los bits correspondientes del registro habilitador de interrupciones IE0: EX0, ET0, EX1 y ET1. Esto se consigue cargando en el registro IE0 el valor 0x8F (1000 1111b).

Por otra parte, se deben poner a uno lógico los bits IT0 e IT1 del registro TCON, con el objetivo de que las interrupciones externas /INT0 e /INT1 se activen por nivel. Esto se consigue almacenando en el registro TCON el valor 0x0A (0000 1010b).

A continuación se debe establecer la prioridad de las interrupciones. En principio se adjudica a las interrupciones /INT0 y Timer0 una prioridad mayor que a las interrupciones /INT1 y Timer1, dado que la situación de rebasamiento requiere una actuación más urgente que la superación del nivel de pH. Las fuentes /INT0 y Timer0 se programan con nivel de prioridad uno, mientras que las fuentes de interrupción. /INT1 y Timer1 se programan con un nivel de prioridad cero.

Por último, se debe poner el bit INTR del byte de configuración CONFIG1 a uno lógico. De esta forma, cuando la CPU ejecute una interrupción, se cargarán en la pila, automáticamente, los tres bytes del contador de programa, PC, y el registro de estado, PSW.1.

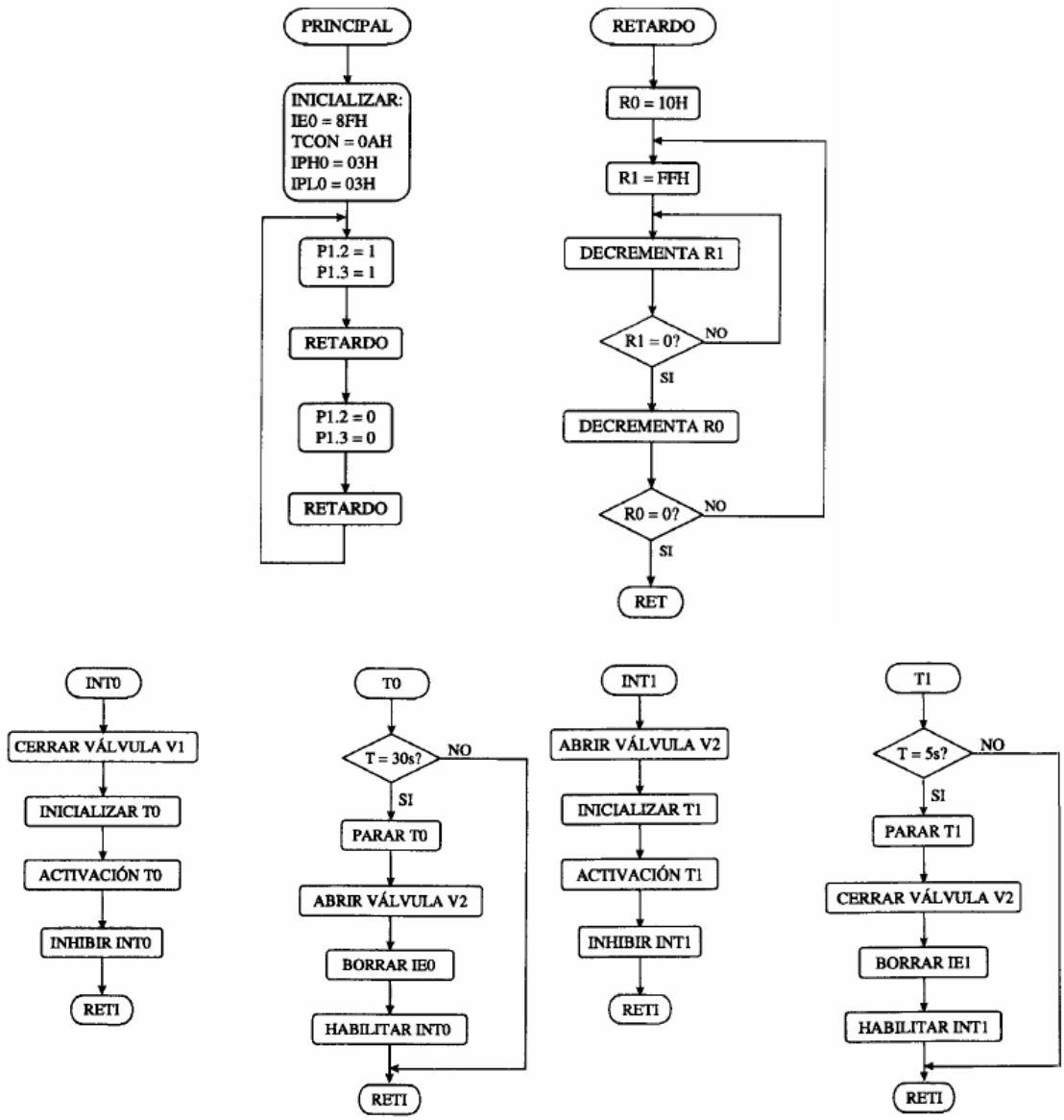
En cuanto a la rutina de retardo, que se utiliza para temporizar el parpadeo de los indicadores luminosos, estará compuesta de dos bucles anidados, basados en decrementar las variables 1 y j. Modificando el valor de las cuentas máximas se puede variar el tiempo de ejecución de la rutina y, por tanto, la frecuencia de parpadeo. Para calcular el tiempo de retardo hay que determinar el número de veces que se ejecuta cada instrucción, así como el tiempo que tarda en ejecutarse cada una de ellas.

En la figura se muestran los flujogramas correspondientes al programa principal y a la rutina de retardo. En la siguiente figura se muestran los flujogramas de las rutinas de atención a las interrupciones /INT0 y Timer0. Las tareas que debe realizar la rutina de atención a la interrupción /INT0 son:

- Cerrar la válvula V1.
- Inicializar el Timer0 para que temporice 30 s.
- Inhibir la interrupción /INT0 para impedir que se ejecute de forma repetida la RSI de la /INT0 mientras el sensor S1 se encuentra activo y no ha finalizado la temporización de 30s.

Si consideramos que la frecuencia de reloj es de 1.2 MHz, se puede determinar, mediante un cálculo sencillo, que el Timer0 debe rebasar varias veces para temporizar 30 s.

En la RSI del Timer 0 se debe contar el número de rebasamientos que sufre este temporizador. Cuando el Timer 0 rebasa un número de veces equivalente a 30 s se abre la válvula V1, se detiene el funcionamiento del Timer 0 y se habilita de nuevo la interrupción /INT0, de forma que si el sensor S1 continúa activo se repite de nuevo la secuencia de cierre de V1.




```
#include <reg51.h>
//prototipo de delay
void delay(unsigned char);
// Constantes
sbit LED_A1=P1^2;
sbit LED_A2=P1^3;
sbit VALV_V1=P1^0;
sbit VALV_V2=P1^1;
//Variables
unsigned char rebosamientos_T0,rebosamientos_T1;

/**********************************
 VECTORIZACION INTERRUPCIONES
**********************************/
//Vector de interrupción de /INT0
void RSI_Int0(void) interrupt 0 {
VALV_V1=0;        //Se cierra la válvula VI
TMOD|=0x01;       //Se programa el Timer 0 en modo 1
TH0=0x3C;         //Se pone Timer O para temporizar 0.5 s
TL0=0x0AF;
rebosamientos_T0=0;  //contador de rebasamientos del Timer 0
TR0=1;            //Puesta en marcha del Timer O
```

```
EX0=0;                  // Se inhibe la interrupción /INT0
}
//Vector de interrupción del Timer 0
void RSI_Timer0(void) interrupt 1 {
rebosamientos_T0++;     //Incrementa el contador de rebasamientos
//Si núm menor o igual que 59 continua
if (rebosamientos_T0>59) {
            TR0=0;       //Caso contrario detiene el Timer 0 y
            VALV_V1=0;  //abre la válvula V1
            EX0=1;      //Habilita de nuevo la interrupción /INT0
            IE0=0;
            }
}
//Vector de interrupción de /INT1
void RSI_Int1(void) interrupt 2 {
VALV_V2=1;         //Se abre la válvula V2.
TMOD|=0x02;        //Se programa el Timer1 en modo 1
TH1=0x3C;          //Inicializa Timer 1 para 0.5 s
TL1=0x0AF;
rebosamientos_T1=0;  //contador de rebasamientos del Timer 1
TR1=1;             //Puesta en marcha del Timer 1
EX1=0;             //Se inhibe la interrupción /INT1
}
//Vector de interrupción del Timer 1
void RSI_Timer1(void) interrupt 3 {
rebosamientos_T1++; //Incrementa el contador de rebasamientos
//Si menor o igual que 9 continua
if (rebosamientos_T1>9) {
      TR1=0;       //Se detiene el Timer 1
      VALV_V2=1;  //Se abre la válvula V2
      EX1=1;       //Se habilita la interrupción
      IE1=0;       //Se borra flag de la interrupción /INT1
      }
}

/**********************************
 PROGRAMA PRINCIPAL
**********************************/
void main(void) {
// Programación nivel prioridad de las interrupciones externas 0 y 1
IE0=0x8F;   //Habilitación de las interrupciones
TCON=0x0A;  //Se programa /INT0 e /INTl por nivel
IP=0x03;    //programación niveles de prioridad
//Secuencia de parpadeo de los indicadores luminosos Al y A2
while(1) {
      LED_A1=LED_A2=1;
      delay(50);
      LED_A1=LED_A2=0;
      delay(50);
      }
}
```

Cálculo del valor inicial del Timer 0

Para temporizar los treinta segundos se inicia el Timer 0 para que tarde 0.5 s en rebasar (de forma que se deben contabilizar 60 rebasamientos en la RSI del Timer 0 para abrir de nuevo la válvula V1). Para temporizar 0.5 s, con una frecuencia de reloj de 1.2 MHz, el Timer 0 debe incrementarse 50.000 veces (0xC350); luego se ha de cargar dicho Timer con la combinación resultante de restar a 0xFFFF el valor 0xC350:

$$0xFFFF - 0xC350 = 0x3CAF$$

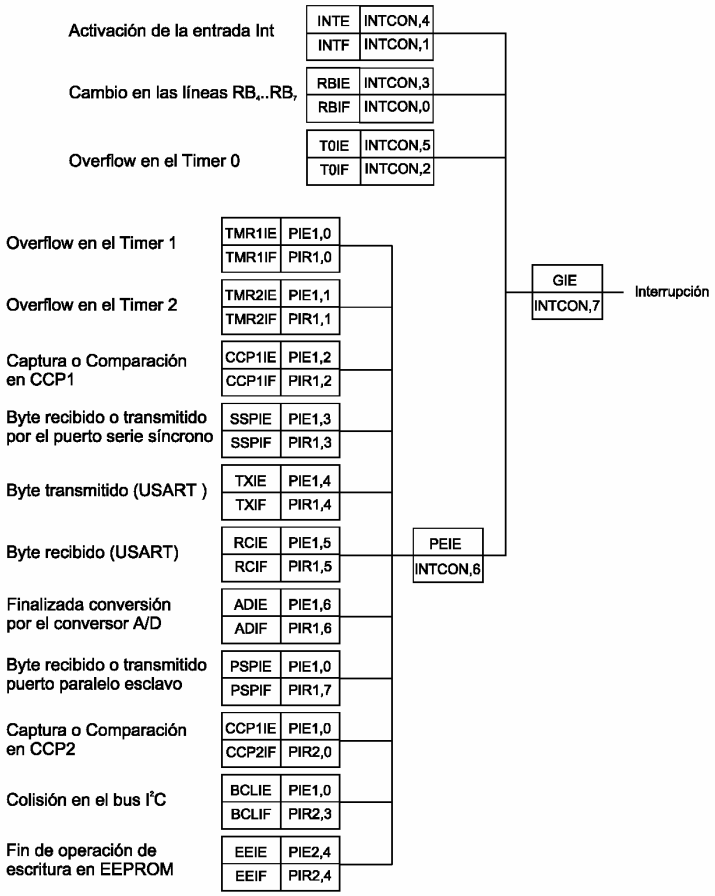
3.- Interrupciones en la familia PIC16.

Vamos a detallar como se manejan las interrupciones en el PIC 16F877, aunque en toda la familia PIC16 se realiza de la misma manera. En la figura siguiente se detallas las características básicas de este microcontrolador.

	Descripción	Características
Procesador	Núcleo	RISC, Arq. Harvard, 20 MHz. 5 MIPS
	Interrupciones	14 fuentes posibles de interrupción
	Reloj	0-20 MHz.
	Reset	Master Clear, Brown Out, Watchdog
	Instrucciones	35 instrucciones de 14 bits
Memoria	M. de programa	8 K palabras de 14 bits
	M. de datos RAM	368 registros de 8 bits
	M. de datos EEPROM	256 registros de 8 bits
	Pila	8 palabras de 13 bits
	M. de datos ext. EEPROM	Hasta 256 KBytes
Periféricos	Puertos programables de E/S	Hasta 33 bits, pueden ser usados por otros periféricos
	Timers/Counters	Dos de 8 bits y uno de 16 bits
	Puertos de captura/comparación de datos	Dos de 8 bits
	Moduladores de ancho de pulso (PWM)	Dos de 8 bits
	Conversor Analógico/Digital de 10 bits	Con un MPX de 8 canales para 8 entradas diferentes
	Puerto serie síncrono	Configurable en modo SPI e I^2C
	USART	Para conexiones RS 232
	Parallel Slave Port	8 bits + 3 bits de control

3.1.- Tipos de interrupciones.

La lista de fuentes de interrupción junto con sus bits de habilitación y de monitorización:



El microcontrolador dispone de una **PILA HARDWARE de ocho niveles** que no se controla desde el programa y permite guardar la dirección de retorno de la RSI. Si se produce un anidamiento de más de 8 niveles se pierde la última dirección de retorno guardada.

Todas las interrupciones, como vemos, se gestionan a través del registro INTCON que se encuentra en la posición de memoria 0x08B. El registro se configura de la siguiente forma:

GIE	PEIE	T0IE	INTE	RBIE	xxxIE	Significado
0	X	X	X	X	X	Todas las interrupciones deshabilitadas
1	0	X	X	X	X	Deshabilitadas las interrupciones de los periféricos internos salvo Timer 0
1	1	X	X	X	X	Permitidas las interrupciones de los periféricos internos. Hay un bit adicional para cada periférico.
1	X	0	X	X	X	Deshabilitada Int. Timer 0
1	X	1	X	X	X	Habilitada Int. Timer 0
1	X	X	0	X	X	Deshabilitada Int. externa
1	X	X	1	X	X	Habilitada Int. externa
1	X	X	X	0	X	Deshabilitada Int. cambio líneas RB4,..,RB7
1	X	X	X	1	X	Habilitada Int. cambio líneas RB4,..,RB7

Los bits de habilitación de las interrupciones para los periféricos se pueden manejar directamente desde código C empleando los nombres de bits que se detallan en la tabla siguiente:

Bit	Reg.	bit	Activar interrupción si	Flag	Reg.	bit
TMR1IE	PIE1	0	Overflow en el Timer 1	TMR1IF	PIR1	0
TMR2IE	PIE1	1	Overflow en el Timer 2	TMR2IF	PIR1	1
CCP1IE	PIE1	2	Captura o Comparación en CCP1	CCP1IF	PIR1	2
SSPIE	PIE1	3	Byte recibido o transmitido por el puerto serie síncrono	SSPIF	PIR1	3
TXIE	PIE1	4	Byte transmitido por la USART	TXIF	PIR1	4
RCIE	PIE1	5	Byte recibido por la USART	RCIF	PIR1	5
ADIE	PIE1	6	Finalizada conversión por el conversor A/D	ADIF	PIR1	6
PSPIE	PIE1	7	Byte recibido o transmitido por el puerto paralelo esclavo	PSPIF	PIR1	7
CCP2IE	PIE2	0	Captura o Comparación en CCP2	CCP2IF	PIR2	0
BCLIE	PIE2	3	Colisión en el bus I^2C	BCLIF	PIR2	3
EEIE	PIE2	4	Fin de operación de escritura en la EEPROM	EEIF	PIR2	4

De esta forma resulta posible y cómodo configurar el sistema de interrupciones directamente en código C.

El proceso para la atención a las interrupciones es el siguiente: cuando se produce la interrupción se escribe un nivel lógico **0 en el bit GIE**, de forma que no se atienden más interrupciones. A continuación se guarda la dirección de retorno en la pila hardware. Por último se escribe la dirección del **vector único de interrupción (0x04)** en el contador de programa. La rutina de interrupción debe:

1. Salvar en algún registro reservado al efecto el contenido del registro de estado del micro y del acumulador (esto lo hace el compilador).
2. Haciendo polling se detecta cual ha sido la fuente de interrupción (esto lo hacemos nosotros).
3. Ejecutamos la rutina de atención a la interrupción, la que hace el trabajo necesario para ese periférico (esto lo hacemos nosotros).
4. Desactivar el flag correspondiente a esa interrupción, es decir, ponerlo a cero (esto lo hacemos nosotros).
5. Ejecutar la instrucción RETFIE de retorno de interrupción y que reactiva GIE (esto lo hace el compilador).

3.2.- Definición de rutinas de servicio de interrupción en C para PIC.

A la hora de codificar en lenguaje C una rutina de interrupción se sigue una técnica similar a como lo hacemos con la familia MCS-51 de microcontroladores.

El compilador de C dispone de una palabra reservada que nos permite marcar una función como la rutina de servicio de interrupción y como en el PIC la RSI es única solo es posible definir una en cualquier programa. Esta palabra reservada es también **interrupt**, como sucede con el MCS-51, pero esta vez solo puede haber un vector, por lo que no lleva ningún numero detrás de la palabra clave y se coloca delante de la función de interrupción.

Para definir la interrupción tan solo hay que hacer lo siguiente:

```
#include <pic.h>
interrupt void general(void) {
      if (TMR1IF) RSI_TMR1();
      if (TMR2IF) RSI_TMR2();
      if (CCP1IF) RSI_CCP1();
      if (SSPIF)  RSI_SSP();
      if (TXIF)   RSI_TX();
      if (RCIF)   RSI_RC();
      if (ADIF)   RSI_AD();
      if (PSPIF)  RSI_PSP();
      if (CCP2IF) RSI_CCP2();
      if (BCLIF)  RSI_BCL();
      if (EEIF)   RSI_EE();
}
```

Mediante consulta vamos llamando a las rutinas de servicio de interrupción pertinentes. Claro está, si no utilizamos todas las fuentes de interrupción habrá que eliminar las rutinas de tratamiento de las fuentes de interrupción no utilizadas. Incluso es posible que algunas RSI se encuentren en otros módulos diferentes. El linkador se encargar de encontrar la rutina correcta en tiempo de compilación.

Ejemplo 6: Conexión de teclas y de un dígito de siete segmentos

Este ejemplo es idéntico al anterior para el MCS-51, con las siguientes modificaciones:

- El microcontrolador empleado es un PIC16F876.
- El programa debe ser capaz de detectar la pulsación de varias teclas al mismo tiempo, e indica esta situación a través del encendido de todos los leds del dígito conectado.
- La pulsación de una tecla, T1, T2, T3 o T4, se indicará poniendo el número 1, 2, 3 o 4, en el dígito, respectivamente. En el dígito siempre se mostrará la última tecla pulsada.
- La pulsación de T5 apagará todos los leds del dígito T5.

La interrupción se habilita por flanco descendente, mientras que la interrupción /INT0 se habilita por nivel lógico, de manera que en la rutina RSI de /INT0 se debe borrar el bit IE0. Sin embargo, en la rutina RSI de /INT1 el bit IE1 se borra de forma automática.

```
#include <pic.h>
//Definicion de entradas
#define Tecla_1 RC0 //Puerto C bit 0
#define Tecla_2 RC1
```

```
#define Tecla_3 RC2
#define Tecla_4 RC3
//variables globales
unsigned char tecla;
// Rutina de tratamiento de interrupción externa
void RSI_INT(void) {
      if (!Tecla_1)           tecla=0x01;
      else if (!Tecla_2)      tecla=0x02;
      else if (!Tecla_3)      tecla=0x03;
      else                    tecla=0x04;
      INTF=0;  //Bajamos la bandera de interrupción
}
// Rutina global de interrupcion
interrupt void global(void) {
if (INTF) RSI_INT();
//..Aqui se colocan otras causas de interrupcion
}
/*********************************************************
 Rutina de inicio.
 (habilita interrupciones y INT EXT)
*********************************************************/
void inicio(void) {
      INTE=1;  // Habilita interrupcion externa
      PEIE=0;  // Deshabilita resto de interrupciones
      GIE =1;     // Habilitación general (Registro E)
}

unsigned char Siete_seg[5]={0x3F,0x06,0x5B,0x4F,0x66,0x00};

/*********************************************************
 Programa Principal
*********************************************************/
void main(void) {
inicio();
//Bucle infinito sin propósito definido
while(1) {
      P2=Siete_seg[tecla];  //Codifica para mostrar el digito
 }
}
```

La función “main” de este ejemplo se encarga de leer el contenido de la variable “tecla” y de poner en el dígito el carácter correspondiente según sea el valor de “tecla”. La variable “tecla” puede valer 0x00 al principio, si no se ha pulsado ninguna tecla, o tras pulsar la tecla T5; puede valer 0x0l, 0x02, 0x03 ó 0x04, al pulsar las teclas T1, T2, T3 o T4, respectivamente; y puede valer 0x05 en el caso de que se pulsen varias teclas al mismo tiempo. Con el valor de la variable “tecla” se lee la tabla “Siete_seg”, que proporciona el código en siete segmentos, correspondiente al carácter que se quiere mostrar en el dígito del ejemplo. De todas formas, la tabla “Siete_seg” se puede suprimir colocando en la variable “tecla”, directamente, el código del carácter que deberá mostrar en el dígito, según sea el caso.

4.- Temporizadores/contadores internos.

4.1.- Introducción

Los microcontroladores de la familia MCS-51 disponen de hasta tres contadores internos de 16 bits, denominados Timer 0, Timer 1 y Timer 2, que pueden ser configurados de forma individual para operar en diversos modos de trabajo. Estos contadores, en general, se incrementan en una unidad cada vez que les entra un pulso de reloj, el cual puede provenir de la frecuencia de reloj del microcontrolador o bien de un terminal externo, por lo que, en el primer caso, al Timer se le denomina temporizador y, en el segundo caso, se le denomina contador.

Por otra parte hay microcontroladores que incorporan un contador adicional, cuya misión consiste en hacer de dispositivo de watchdog ("perro guardián"). Este contador se emplea para reiniciar la CPU cuando ésta, por problemas de interferencia electromagnética, o por algún fallo interno, pierde el rumbo habitual de ejecución de las instrucciones, fenómeno que resulta fácil de distinguir por la gestión caótica que hace la CPU de sus periféricos. El Timer de watchdog es un contador que se incrementa con cada pulso del reloj de la CPU; genera entonces un reset interno que inicializa el microcontrolador cuando llega a desbordamiento.

Con el Timer de watchdog, para que se active este reset interno, deben pasar unos pocos milisegundos, que es el tiempo necesario para que el contador de watchdog llegue al desbordamiento. De manera que, cuando el watchdog está activado, el programador debe insertar instrucciones de refresco o de inicialización del Timer cada cierto intervalo de instrucciones. Se trata, pues, de que el watchdog no llegue desbordarse, mientras el flujo de ejecución de instrucciones por parte de la CPU sea correcto. Si la CPU pierde el control del sistema, las instrucciones se ejecutan de forma caótica, por lo que la serie de instrucciones de refresco del watchdog no se ejecutan con la regularidad prevista, lo que causa el desbordamiento del Timer y un auto-reset de la CPU para inicializar el sistema.

Se debe tener en cuenta que la homologación de muchos productos basados en microcontroladores se somete a pruebas con intensas interferencias electrostáticas y electromagnéticas, con el propósito de que la CPU pierda el rumbo habitual de ejecución. En este caso, el sistema debe inicializarse.

4.2.- Temporizadores/contadores para la MCS-51

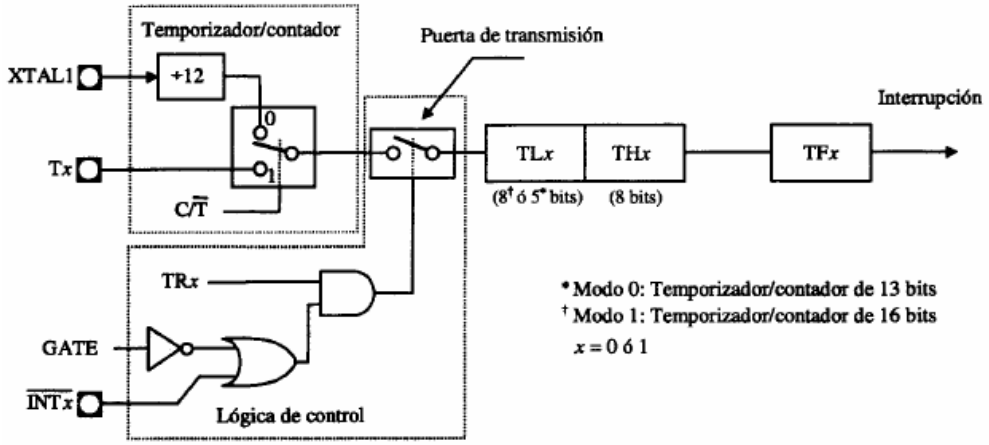
4.2.1.- Timer 0 y Timer 1

El Timer 0 y el Timer 1 son contadores de 16 bits que están presentes en todos los microcontroladores de la MCS-51. El valor de estos contadores se refleja en los registros TH0 y TL0, para el Timer 0, y en los registros TH1 y TL1, para el Timer 1. El registro THx constituye el byte alto del contador y el registro TLx el byte bajo.

El Timer 0 y el Timer 1 pueden funcionar con cuatro modos de trabajo distintos, seleccionables mediante el registro TMOD, **Timer Mode**. Los modos de trabajo son: modo 0, temporizador/contador de 13 bits; modo 1, temporizador/contador de 16 bits; modo 2, temporizador/contador de 8 bits con autorrecarga; y modo 3, varios contadores.

La figura muestra el esquema funcional del Timer 0 y del Timer 1, para los modos 0 y 1 de funcionamiento, con temporizador de 13 y 16 bits, respectivamente. En esta figura se resaltan,

con un cuadro de puntos, dos bloques: el bloque de selección de temporizador/contador y el bloque de la lógica de control. Estos bloques son comunes al Timer 0 y al Timer 1 y a los modos de funcionamiento 0, 1 y 2 de los Timers.



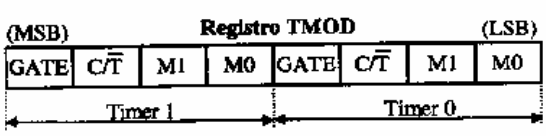
El bloque de selección de temporizador/contador, mediante el bit C/T del registro TMOD, **Timer Mode**, selecciona la fuente de entrada de pulsos del Timer, que puede provenir directamente de la señal de reloj del microcontrolador dividida por 12, para C/T a 0 lógico, o de una entrada de pulsos externa a través del terminal Tx del microcontrolador, para C/T; a 1 lógico.

El bloque de lógica de control utiliza una puerta de transmisión para dejar pasar, o no, los pulsos de reloj hacia el contador (figura 7.1). En este bloque intervienen el bit TRx, **Timer Run**, del registro TCON, el bit GATE del registro TMOD y el terminal de entrada /INTx, con el propósito de controlar el paso de pulsos de reloj hacia el contador, lo que a partir de este momento se definirá como puesta en marcha o parada del contador. Con los bits TRx y GATE se puede poner en marcha o parar el Timer de forma directa, por software, o bien, por medio del estado lógico del terminal /INTx, por hardware.

Observando el circuito de la lógica de control, se deduce que si el bit GATE se pone a 0 lógico, en la entrada de la puerta OR, habrá un 1 lógico y, por tanto, un 0 lógico en la entrada de la puerta AND; en esta situación queda claro que la puerta está gobernada por el estado lógico del bit TRx. Entonces, si TRx vale 0 lógico, la puerta queda abierta y el Timer parado, y si TRx vale 1 lógico, la puerta queda cerrada y el Timer en marcha, y se incrementa a cada pulso de entrada. En esta situación, con el bit GATE a 0 lógico, la puesta en marcha o la parada del Timer depende del estado del bit TRx, y queda, en consecuencia, gobernado por software.

Si GATE se pone a 1 lógico, a la entrada de la puerta OR, tras la puerta inversora, habrá un 0 lógico; luego, el estado lógico de la puerta OR dependerá directamente del estado lógico del terminal /INTx. Si al mismo tiempo, el bit TRx se pone a 1 lógico, queda claro que el gobierno de la puerta de transmisión dependerá del terminal /INTx, que pondrá en marcha el Timer cuando /INTx esté a 1 lógico, y parará el Timer cuando /INTx esté a 0 lógico, lo que se mencionará, a partir de ahora, como puesta en marcha o parada del contador por hardware, es decir, por el estado lógico de /INTx.

La determinación del modo de funcionamiento del Timer 0 y del Timer 1 se realiza con los bits M0 y M1 del registro TMOD: existen, en efecto, en este registro un par de bits para cada Timer. El contenido del registro TCON se muestra en la tabla siguiente.



Bit	Comentario
GATE	GATE a 0 lógico hace que el Timer se gobierne mediante TRx, con TRx a 1 lógico se pone en marcha el Timers y con TRx a 0 lógico se detiene (x=0 ó 1). GATE a 1 lógico, junto con TRx a 1, hace que el Timer se gobierne por hardware, mediante el estado lógico de la entrada /INTx
C/T	Selecciona entre pulsos de la señal de reloj o pulsos del terminal Tx. Si C/T está a 0 se toman los pulsos de la señal de reloj y si C/T está a 1 se toman de Tx
M1 M0 0 0 0 1 1 0 1 1	Selección del modo de trabajo Modo 0. Temporizador/contador de 13 bits. Modo 1. Temporizador/contador de 16 bits. Modo 2. Temporizador/contador de 8 bits con autorrecarga. Modo 1. Varios contadores.

Modo 0 Temporizador/contador de 13 bits.

En el modo 0, el Timer 0 y el Timer 1 operan como temporizadores/contadores de 13 bits, empleando, en realidad, el registro TLx como un divisor previo de 5 bits, que puede dividir la frecuencia de la señal de entrada de los Timers hasta por 32; y el registro THx como un contador de 8 bits conectado a la salida de TLx. Los tres bits altos de TLx están en un estado indeterminado, que debe ser ignorado.

El sentido del contador es siempre ascendente y se incrementa con cada pulso de entrada. El rebasamiento del Timer se produce cuando pasa de tener todos los bits de uno lógico a cero lógico; entonces se activa el bit correspondiente de rebasamiento, TFx, y se genera una interrupción a la CPU.

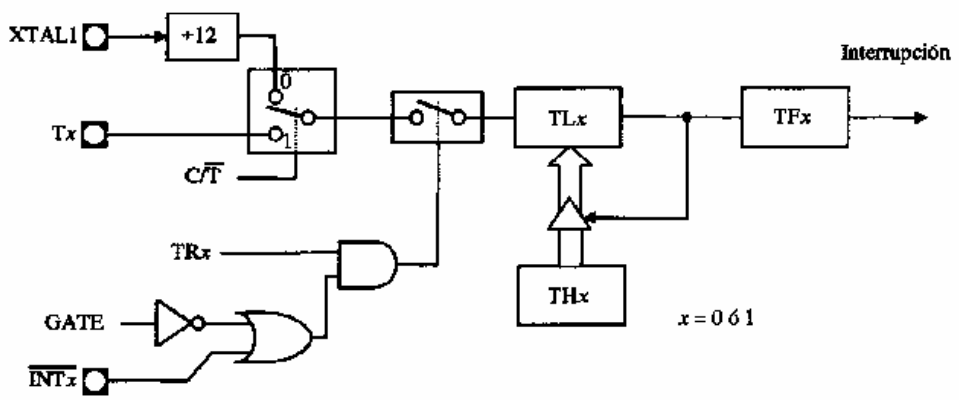
Modo 1. Temporizador/contador de 16 bits.

En el modo 1 el Timer 0 y el Timer I operan como un temporizador/contador de 16 bits. En este modo los registros TLx y THx son dos contadores de 8 bits que están conectados en cascada para formar el contador de 16 bits.

Al igual que en el modo 0, en el modo1 el sentido del contador es ascendente, y el rebasamiento se produce al pasar el contenido de los registros TLx y THx de todos los bits de uno lógico a cero lógico. El rebasamiento entonces activa el bit TFx y genera una interrupción a la CPU.

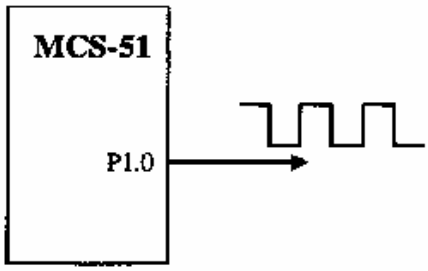
Modo 2. Temporizador/contador de 8 bits con autorrecarga.

En el modo 2 el Timer 0 y el Timer 1 funcionan como un temporizador/contador de 8 bits con autorrecarga. El contador está formado por el registro TLx, mientras que el registro THx hace la función de registro de recarga. En este modo de trabajo, cuando se produce un desbordamiento en el contador, el bit TFx se activa de manera automática, causa una interrupción a la CPU y hace, al mismo tiempo, que el contenido de THx se cargue en el registro TLx, por lo que el contador comienza a contar pulsos a partir del valor cargado. Este proceso de recarga no modifica el valor del registro THx, mediante el cual se pueden generar períodos de tiempo precisos a partir del valor situado en THx.



Ejemplo 7: Generación de una señal periódica.

Es habitual en un sistema tener que generar señales periódicas: en el caso de la MCS-51 se pueden llevar a cabo empleando uno o varios de sus temporizadores. En este ejemplo se usa el Timer 0 en el modo 2 de 8 bits con autorrecarga, para generar una señal binaria de 5kHz de frecuencia. Para determinar el período de la señal se empleará el registro TH0 del Timer 0. La señal generada se extraerá a través de la patilla P1.0 del microcontrolador.




```
#include <reg51.h>
/**************************************
 Generación de una señal periódica
**************************************/
//Definicion salida
sbit SALIDA=P1^0;

/*****************************************
*
 Rutina de servicio del Timer0
*****************************************/
void RSI_Timer0(void) interrupt 1 {
      SALIDA=~SALIDA;   //Complementa P1.0. Pasa 1a0, y 0a1
                        //El bit TFO se borra automáticamente RETI
}
/**************************************
 Rutina de inicio
**************************************/
void inicio(void) {
      PT0=1;      //Asigna prioridad alta al Timer 0
      ET0=1;      //Habilita interrupción del Timer 0
      EA=1; //Habilita bit de interrupción general
      TMOD=0x02;  //Ml=l y MO=O (Modo 2),
                        // GATE=O y C/T=O del Timer O
      TL0=156;    //Pone valor 156 en TL0 (256-100)
      TH0=156;    //Pone valor de recarga en THO (256-100)
      TR0=1;            //Pone en marcha el contador
}
/**********************************************
 Rutina de Principal
**********************************************/
void main(void) {

inicio();
while (1) {

      }
}
```

En el programa realizado se habilita la interrupción del Timer 0 (ET0=l, EA=1), se le da prioridad alta a la interrupción del Timer 0 (FT0=l), y se configura el Timer 0 en el modo 2 de 8 bits con autorrecarga (M1=1 y M0=0, registro TMOD). La puesta en marcha del Timer 0 es por software (GATE0, registro TMOD), el Timer 0 cuenta pulsos procedentes del reloj del microcontrolador (C/T=0, registro TMOD), y, al final de la rutina Inicio, se pone en marcha el Timer (TR0=1).

El Timer 0 está gestionado completamente por interrupciones, de manera que cada vez que llega a desbordamiento se activa el bit TF0, TF0=l, se produce una interrupción a la CPU y se recarga el valor del registro TH0 en el registro TL0. En la rutina de atención a la interrupción del Timer 0, RSI_Timer0, basta con complementar el estado lógico de la patilla P1.0, para generar la frecuencia requerida. Con esta instrucción el microcontrolador lee el estado de la patilla, lo complementa (pasa de 0 lógico a 1 lógico, y viceversa) y escribe el nuevo estado en la patilla.

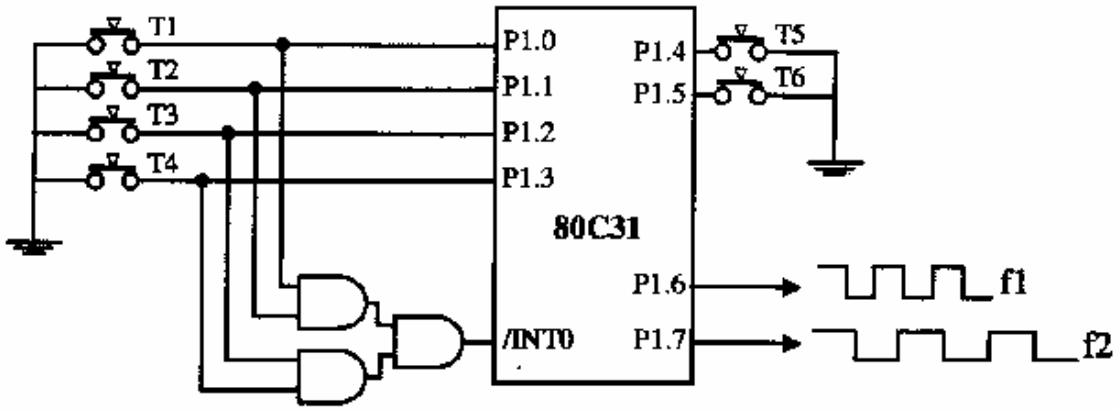
El bit de rebosamiento se activa con la petición de interrupción y se borra, de forma automática, cuando la CPU atiende a la interrupción ejecutando la rutina RSI-Timer0.

Para determinar el valor de recarga del registro TH0, se debe considerar que el período necesario para generar frecuencia de 5 kHz es de 200 μs, por lo que la patilla P1.0 se debe complementar cada 100 μs, es decir, la mitad del período. Además, desde que se produce la interrupción (estado S5P2 del ciclo máquina), hasta que se vectoriza la interrupción del Timer 0, transcurren tres ciclos máquina en el mejor de los casos, y la instrucción de complemento lógico se ejecuta aproximadamente en un ciclo máquina. Por tanto, el estado lógico de la patilla P1.0 cambiará, al menos, cuatro ciclos máquina después de que se halla activado el bit TF0.

El valor de recarga de TH0 debe ser de 156, es decir, el valor de la diferencia entre 256 (valor de rebosamiento) y el tiempo deseado, 100 μs. Se debe tener en cuenta que para un reloj de 12MHz, el Timer 0 se incrementa cada microsegundo y que un ciclo máquina tiene la duración de 1 μs.

Ejemplo 8: Generación de diversas frecuencias.

En este ejemplo se generarán distintas frecuencias por las patillas P1.6 y P1.7 del microcontrolador, en función del estado de las teclas T1, T2, T3, T4, T5 y T6. Las cuatro primeras teclas están conectadas a la entrada de interrupción /INT0 mediante puertas AND lógico, de forma que su pulsación causará una interrupción a la CPU.



En este ejemplo el microcontrolador debe generar en la patilla P1.6 una frecuencia de 4KHz, 5kHz, 10kHz y 20kHz, si se pulsan las teclas T1, T2, T3 o T4, respectivamente. Las teclas T5 y T6 afectarán a la frecuencia de la patilla P1.7. Al pulsar T5 se generará en P1.7 una frecuencia de 25kHz, al pulsar T6 la frecuencia será de 50kHz. Inicialmente, por defecto, en P1.6 y en P1.7

se generará una Frecuencia de 2kHz. Cuando se pulse una tecla se generará la frecuencia correspondiente, y cuando no se pulse ninguna se volverá a generar la frecuencia inicial de 2kHz.

```
#include <reg51.h>
/************************************
 Generación de varias señales periódicas */
// Entradas y salidas
sbit Tecla_T1=P1^0;
sbit Tecla_T2=P1^1;
sbit Tecla_T3=P1^2;
sbit Tecla_T4=P1^3;
sbit Tecla_T5=P1^4;
sbit Tecla_T6=P1^5;
sbit SAL_1=P1^6;
sbit SAL_2=P1^7;
/************************************
 Rutina de servicio de /INT0
************************************/
void RSI_Int0(void) interrupt 0 {
if (!Tecla_T1)  TH0=131;//(256- 125) valor de recarga para 4kHz
else if (!Tecla_T2) TH0=156;//(256-100) valor de recarga para 5kHz
else if (!Tecla_T3) TH0=206;//(256-50) valor de recarga para 10kHz
else if (!Tecla_T4) TH0=231;//(256-25) valor de marga para 20kHz
}
/************************************
 Rutina de servicio del Timer0
************************************/
void RSI_Timer0(void) interrupt 1 {
      SAL_1=~SAL_1; //Complementa P1.6
      //El bit TF0 se borra automáticamente
}
/************************************
 Rutina de servicio del Timer1
************************************/
void RSI_Timer1(void) interrupt 3 {
      SAL_2=~SAL_2; //Complementa P1.7
      //El bit TF1 se borra automáticamente
}
/************************************
 Rutina de Inicio
************************************/
void inicio(void) {
      IT0=1;      ///INT0 activa por flanco descendente
      PT0=1;      //Asigna prioridad alta al Timer0
      PT1=1;      //Asigna prioridad alta al Timer 1
      EX0=1;      //Habilita interrupción de /INT0
      ET0=1;      //Habilita interrupción del Timer 0
      ET1=1;      //Habilita interrupción del Timer 1
      EA=1; //Habilita bit de interrupción general
      TMOD=0x22;  //Timer 0 y 1 en Modo 2, con GATE=0 y CX=0
      TL0=6;            //(256-250) valor inicial para 2kHz
      TH0=6;            //carga valor inicial para frec. de 2kHz
      TL1=6;            //carga valor inicial para frec. de 2kHz
      TH1=6;            //carga valor inicial para frec. de 2kHz
      TR0=1;            //Pone marcha el Timer 0
      TR1=1;            //Pone marcha el Timer 1
}
```

```
/************************************
 Rutina Principal
************************************/
void main(void) {
inicio();
while (1) {
      if (!Tecla_T5) TH1=236; //(256-20) valor de recarga para 25kHz
      else if (!Tecla_T6) TH1=246; //(256-10) valor recarga para 50kHz
      TH0=6; //por defecto 2kHz en Timer 0
      TH1=6; //por defecto 2kHz en Timer 1
      }
}
```

En la rutina de inicio se ha activado la interrupción /INT0 por flanco descendente, por lo que la interrupción sólo se producirá una vez, cuando se pulse una tecla, al detectar la CPU un flanco de bajada. En la rutina de inicio se le asigna prioridad alta al Timer 0 y al Timer 1, por ser el núcleo de este ejemplo. El Timer 0 y el Timer 1 se configuran en el modo 2 de autorrecarga contando pulsos del reloj.

Para determinar el valor de los registros TH0 y TH1, de acuerdo con la frecuencia especificada, se debe tener en cuenta que la frecuencia se genera a través de la instrucción CPL en la rutina de RSI de cada Timer. La interrupción complementa el valor del terminal del puerto cada vez que un temporizador llega a rebosamiento, por lo que para generar una frecuencia se debe emplear la mitad del período de ésta. Con una frecuencia de reloj de 12 MHz, el período de cada pulso de reloj es de 1 μs, los temporizadores, pues, se incrementan cada microsegundo. Luego, para generar una frecuencia determinada en el Timer 0 ó 1, el valor del registro TH0 o TH1 debe ser:

$$\text{TH0,1} = \text{Valor de rebasamiento} - \frac{\text{Período reloj}}{2}$$

En este ejemplo, para generar una frecuencia de 2 kHz el valor de TH0 es de 236, para 4 kHz el valor es de 131, etc. El Timer 0 llega a rebasamiento cuando TL0 pasa de 0xFF a 0x00, por lo que el valor de rebasamiento, es decir, el valor máximo del Timer, es de 256 o 0xFFH+l.

La pulsación de las teclas T5 y T6 se detecta mediante software en la rutina principal, donde continuamente se lee el estado de estas teclas, y se asigna al registro TH1 el valor de 236 para generar una frecuencia de 25 kHz, o el valor 246 para generar una frecuencia de 50 kHz.

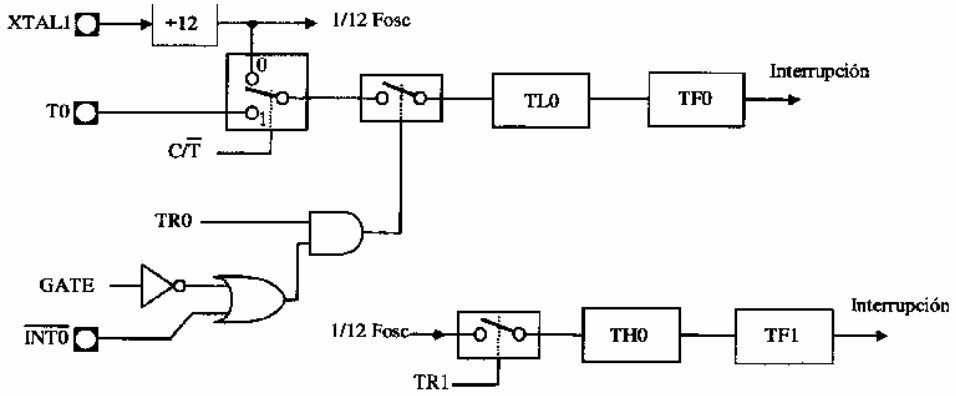
Modo 3. Varios contadores.

En el modo 3 los registros TL0 y TH0 del Timer 0 trabajan como dos contadores independientes de 8 bits cada uno; de manera que se empleará en aquellas aplicaciones que requieren de un temporizador o contador de 8 bits adicional.

En el modo 3 el contador formado por TL0 utiliza todos los bits de control propios del Timer 0, para llevar a cabo el mismo tipo de funcionamiento que el descrito para los otros modos; o sea, usa los bits C/T, GATE, TR0 y /INT0. No obstante, para el contador formado por el registro TH0 sólo se dispone del bit de control TR1, por lo que este contador tiene su funcionamiento restringido: sólo puede ponerse en marcha y pararse mediante el bit TR1. El contador TH0, además, sólo puede contar pulsos procedentes del reloj del microcontrolador.

A nivel de desbordamiento, el contador formado por TL0 afecta al bit TF0, mientras que el contador formado por TH0 afecta al bit TF1, que en los otros modos pertenece al Timer 1.

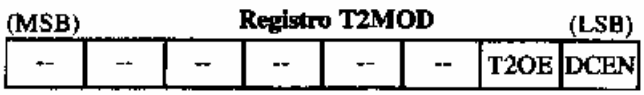
En cuanto al Timer 1, para el modo 3, es obvio que no puede utilizar los bits TR1 y TF1 en su funcionamiento usual, lo que significa que, cuando el Timer 1 se desborde, no se activará el bit TF1, y que tampoco se podrá usar el bit TR1 para activar o para parar el Timer. La forma de arrancar y parar el Timer 1 consiste en entrar y salir de su propio modo 3; es decir, entrando en el modo 3 se pone en marcha el Timer 1, y saliendo del modo 3 se detiene el Timer 1. De todos modos, el Timer 1 puede ser utilizado por el puerto serie como base para la generación de la velocidad de transmisión en baudios o en cualquier otra aplicación en donde no se necesite producir una interrupción.



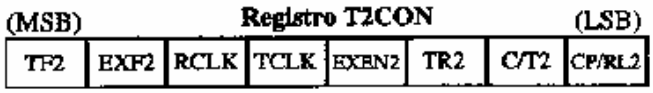
4.2.2.- Timer 2.

El Timer 2 es un contador de 16 bits que está formado a partir de los registros TL2 y TH2, de 8 bits cada uno, conectados en cascada. Este Timer está presente en todas las versiones de la MCS-51 con tres temporizadores internos, y puede trabajar hasta con cuatro modos distintos de funcionamiento: modo autorrecarga, modo captura, modo Baud Rate y modo Clock-out. La forma de trabajar con el Timer 2 se determina a partir de los registros T2MOD y T2CON.

Estos registros permiten determinar el modo de funcionamiento del Timer 2, controlar su puesta en marcha, programar su modo de operación (contador o temporizador) y detectar el desbordamiento. En la tabla siguiente se indican los cuatro modos de operación en los que puede funcionar el Timer 2, dependiendo del estado de los bits RCLK, TCLK, CP/RL2 y T2OE.



Bit	Comentario
-	Bit reservado
T2OE	Bit de habilitación del Timer 2 En el modo Clock-Out, T2OE conecta la salida de desbordamiento con el terminal T2.
DCEN	Bit de sentido de cuenta DECEN=0 hace que el sentido de la cuenta sea ascendente. DCEN=1 hace que el sentido pueda ser ascendente o descendente



Bit	Comentario
TF2	Bit de desbordamiento. TF2=1 al producirse un desbordamiento. Este bit no se activa si RCLK=1 o TCLK=1, Debe borrarse por software.
EXF2	Bit de entrada externa. EXF2 se pone a 2 lógico al producirse un flanco descendente en el terminal T2EX, siempre y cuando EXEN2 esté habilitado
RCLK	Bit de reloj en recepción. RCLK se pone a 1 lógico cuando se produce un desbordamiento en el Timer 0.
TCLK	Bit de reloj en transmisión
EXEN2	Bit de habilitación de entrada externa. En general, si EXEN2=1 permite la activación de EXF2 con un flanco descendente en T2EX. También realiza funciones especificas en todos los modos de funcionamiento del Timer
TR2	Bit de puesta en marcha y parada. TR2=1 pone en marcha el Timer 2. TR2=0 detiene el Timer2.
C/T2	Bit de selección de temporizador/contador. Con C/T2=0 el Timer cuenta pulsos de reloj (/12). Con C/T2=1 el timer cuenta pulsos de la entrada T2.
CP/RL2	Bit de captura/recarga. Con CP/RL2=1 se produce una captura al aplicar un flanco negativo en T2EX, si EXEN2=1. Con CP/RL2=0 se produce una recarga al aplicar un flanco negativo en T2EX, si EXEN2=1. Si RCLK=1 o TCLK=1 se ignora CP/RL2 y se fuerza la recarga del Timer 2 al producirse un desbordamiento en su valor

Modos de funcionamiento del Timer 2.

Modo	RCLK o TCLK	CP/RL2	T2OE
Modo autorrecarga	0	0	0
Modo captura	0	1	0
Modo Baud Rate	1	X	X
Modo Clock-out	X	0	1

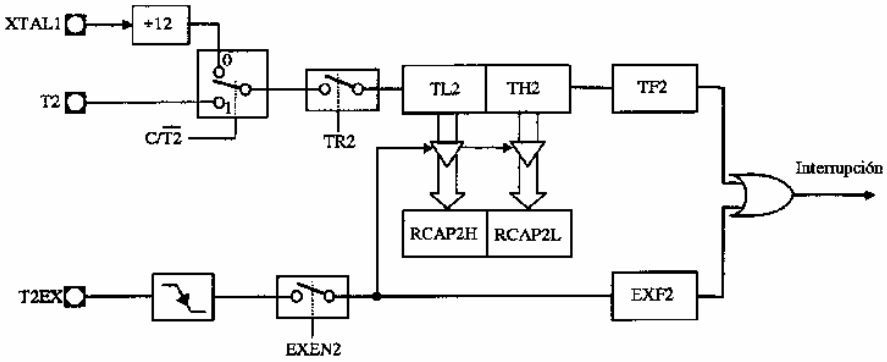
Modo captura

El Timer 2 en el modo captura funciona como un temporizador o contador de 16 bits. Cuando se produce un desbordamiento, al pasar todos los bits de TL2 y TH2 de uno a cero lógico, se activa el bit de rebasamiento TF2 del registro T2CON y se genera una interrupción automática.

En este modo, si el bit EXEN2 del registro T2CON se pone a 1 lógico y se produce un flanco de bajada en el terminal T2EX del microcontrolador, el valor actual de los registros TH2 y TL2 se

copia en los registros RCAP2H y RCAF'2L del área de SFR, hecho al que se le denomina captura. Al mismo tiempo, el flanco de bajada producido hace que el bit EXF2 del registro T2CON se ponga a 1 lógico y se genera una interrupción automática. Así pues, en este modo las fuentes de interrupción hacia la CPU pueden ser por medio del bit de rebasamiento TF2, o bien por medio del bit EXF2. En este último caso, se puede considerar al terminal T2EX como una nueva fuente de interrupción externa, al igual que /INT0 y /INT1. En el caso de que produzca una interrupción, la rutina de RSI que se ejecuta es la misma, tanto para el bit TF2, como para el bit EXF2, de forma que la rutina de RSI debe comprobar cuál de las dos posibles interrupciones se ha activado.

La puesta en marcha y la parada del Timer 2 se realiza con el bit TR2 del registro T2CON. Con TR2 a 1 lógico se pone en marcha el Timer, que queda habilitado para contar pulsos; con TR2 a 0 lógico se detiene el Timer.

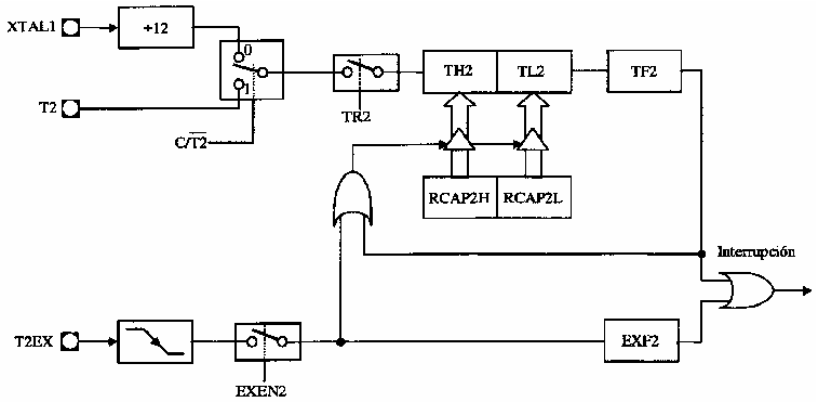


Modo autorrecarga

El modo autorrecarga configura al Timer 2 como un temporizador o contador de 16 bits, con recarga automática del valor contenido en los registros RCAP2L y RACP2H hacia los registros TL2 y TH2, respectivamente. En este modo de funcionamiento, el Timer 2 puede operar de dos formas diferentes: como un contador ascendente, o bien como un contador ascendente ó descendente, según sea el valor del bit DCEN, **Down Counter Enable** bit del registro T2MOD. Para este Último caso, si DCEN está a 0 lógico el sentido de la cuenta es ascendente, y si DCEN está a 1 lógico el sentido de la cuenta es descendente. Se debe tener en cuenta que el bit DCEN por defecto, tras un reset o al iniciar el microcontrolador, se pone a 0 lógico.

a) Contador ascendente (DCEN=0)
Cuando DCEN se pone a 0 lógico, el Timer 2 opera como un contador ascendente de 16 bits. En este caso, la recarga del Timer 2, es decir, la copia de los registros RCAF'2H y RCAP2L en los registros TH2 y TL2, respectivamente, se puede hacer por el desbordamiento del Timer, al activarse el bit TF2, o bien, al producirse un flanco de bajada en el terminal T2EX, siempre y cuando el estado del bit EXEN2 lo permita. En este Último caso, el flanco de bajada también puede causar la activación del bit EXF2 y provocar una interrupción.

El bit DCEN, tras un reset del microcontrolador, se pone a 0 lógico, así que, por defecto el Timer 2 en el modo autorrecarga funciona como un contador ascendente de 16 bits.



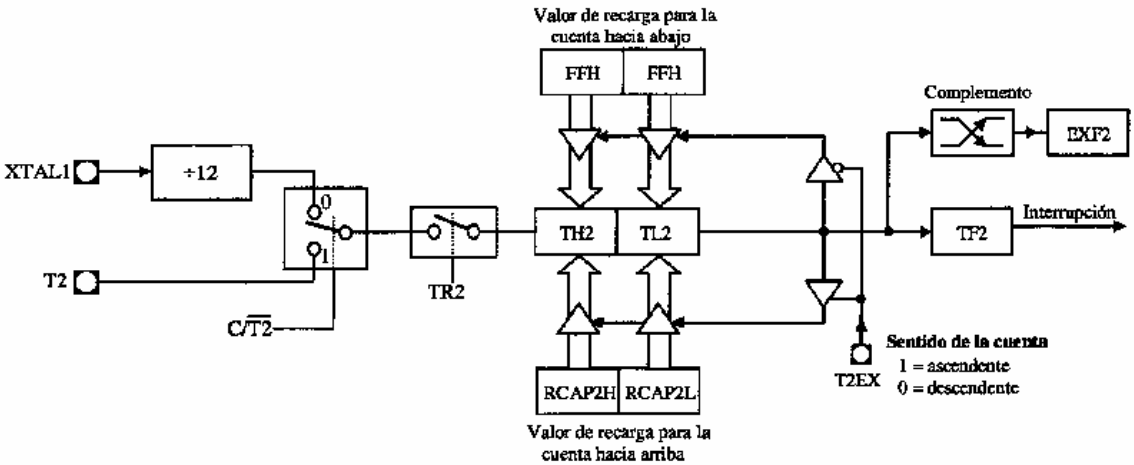
b) Contador ascendente/descendente (DCEN=1)

Cuando DCEN se pone a 1 lógico, el Timer 2 opera como un contador ascendente/descendente, donde el sentido de la cuenta se determina mediante el estado lógico aplicado del terminal T2EX del microcontrolador.

Cuando T2EX está a 1 lógico, el Timer 2 cuenta de forma ascendente, en cuyo caso, el Timer puede llegar a desbordamiento; se activa entonces el bit TF2, se produce una recarga del valor de los registros RCAP2H y RCAP2L hacia TH2 y TL2, respectivamente, y se genera una interrupción automática siempre y cuando se haya habilitado de forma previa.

Cuando T2EX está a 0 lógico, el Timer 2 cuenta en sentido descendente, es decir, se decrementa su valor con cada pulso de entrada. En este sentido de la cuenta, el Timer 2 también puede llegar a un rebasamiento, que se produce cuando el contenido de los registros TH2 y TL2 llega a ser igual al valor almacenado en los registros RCAP2H y RCAPLL, respectivamente. En este caso, se activa el bit TF2, y se produce una petición de interrupción y una recarga del valor FFH en cada uno de los registros, TH2 y TL2.

Al mismo tiempo, cuando se produce cualquiera de los dos tipos de desbordamiento mencionados, el bit EXF2 complementa su valor; es decir, si EXF2 está a 1 lógico, cambia su valor a 0 lógico, y viceversa. El bit EXF2 no genera petición de interrupción al microcontrolador, y se puede utilizar como el bit 17 del contador, es decir, se contempla el contador como un Timer de 17 bits (donde EXF2 es el bit 17).



Modo generador de baudios para el puerto serie (Baud Rate Generator Mode)

En este modo, el Timer 2 queda configurado como generador de baudios para fijar la velocidad de transmisión del puerto de comunicación serie del microcontrolador. Este modo se selecciona al activar los bits RCLK y/o TCLK de registro T2CON (tabla anterior).

Modo Clock-out

En el modo Clock-out el Timer 2 genera una señal de reloj por el terminal T2 del microcontrolador de frecuencia variable y simétrica. El Timer 2 se incrementa con una frecuencia de valor $F_{OSC}/2$ (la mitad de la frecuencia de reloj del microcontrolador), hasta que sufre un desbordamiento, y causa la recarga automática de los registros TH2 y TL2 con el valor contenido en los registros RCAP2H y RCAPLL, respectivamente.

En este modo el desbordamiento del Timer no genera petición de interrupción, como ocurre con los otros modos. La interrupción, sin embargo, sí que se genera al detectar un flanco de bajada en la entrada T2EX, siempre y cuando el bit EXEN2 esté activado.

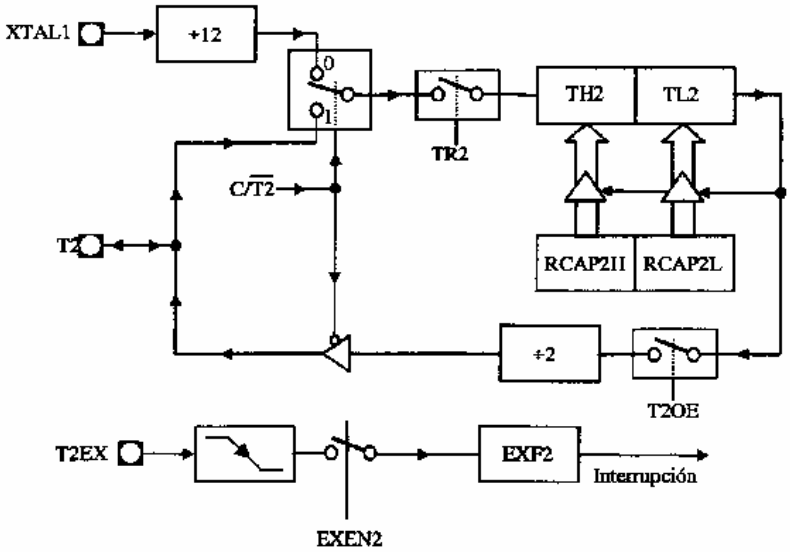
La frecuencia de la señal de reloj generada se establece modificando el valor de los registros RCAP2H y RCAP2L, según la siguiente ecuación:

$$Frec\_reloj = \frac{F_{osc}}{4(65535 - RCAP2H,RCAP2L)}$$

El rango de frecuencias que se puede generar está comprendido entre los 61kHz y los 4MHz, para una frecuencia de reloj de 16MHz.

A modo de resumen, para que el Timer 2 trabaje en modo Clock-out, el bit T2OE del registro T2MOD se debe poner a 1 lógico, el bit C/T2 del registro T2CON se debe poner a 0 lógico, y el bit TR2 debe estar a 1 lógico.

El Timer 2 se puede usar como generador de baudios para el puerto serie y, también de forma simultánea, como generador de reloj, aunque las frecuencias generadas no se pueden determinar de forma independiente, pues comparten los mismos registros de recarga.



4.2.3.- Timer 0, 1 y 2 como contador.

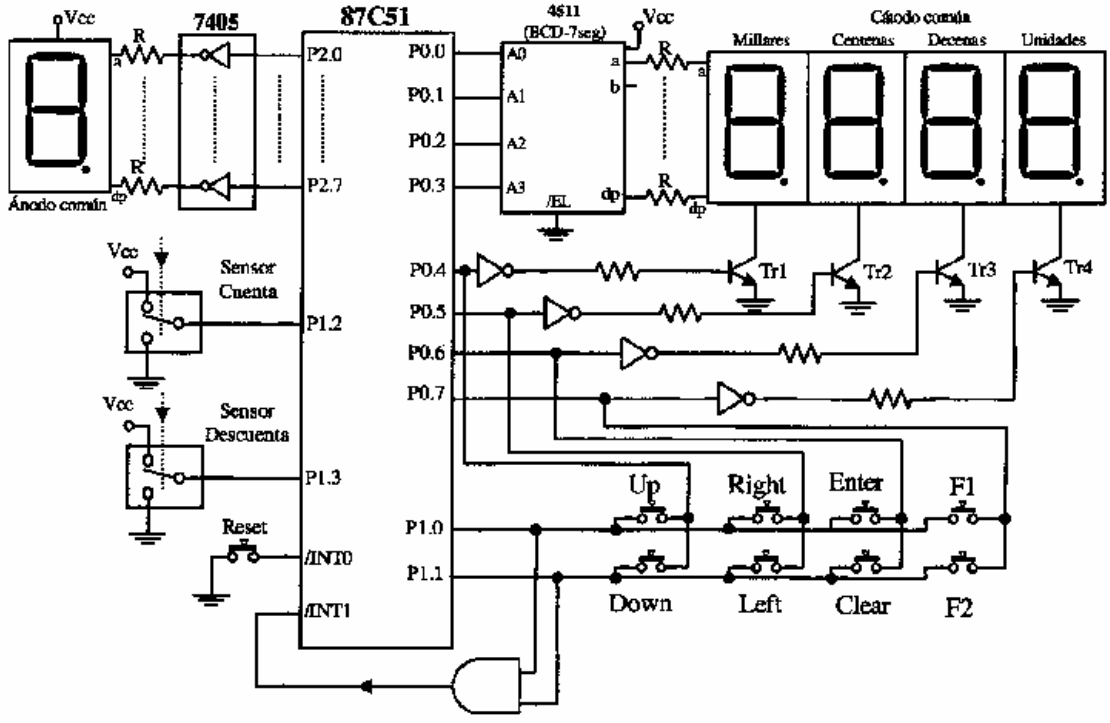
Los Timers 0, 1 y 2 pueden contar pulsos procedentes de los terminales T0, T1 y T2, respectivamente. Para ello los bits $C/\overline{T}$ del registro TMOD, en el caso de los Timers 0 y 1, y el bit $C/\overline{T2}$ para el Timer 2, deben estar a 1 lógico. Los registros de los Timers se incrementan cada vez que se detecta, en los terminales T0, T1 y T2, una transición de 1 a 0 lógico.

El estado lógico de cada uno de estos terminales de entrada se comprueba en la fase 2 del estado 5, S5P2, de cada ciclo máquina. El valor de los registros de los Timers se incrementa cuando en un ciclo máquina se detecta un 1 lógico y un 0 lógico en el siguiente ciclo. El nuevo valor de los registros se muestra en el estado S3P1 del ciclo máquina siguiente al que se ha detectado la transición.

La máxima frecuencia del terminal de entrada es fclock/24, puesto que se tarda dos ciclos máquina en reconocer una transición de 1 a 0 lógico. Para una frecuencia de reloj de 12MHz la máxima frecuencia de terminal T0, T1 o T2 es de 500kHz.

Ejemplo 9: Refresco de cuatro dígitos de siete segmentos y teclado matricial

En el capítulo anterior se explicaba un ejemplo en el que se efectuaba la lectura de un teclado matricial, y también un ejemplo de un contador de piezas para una máquina de un proceso industrial. Estos dos ejemplos se pueden combinar en uno solo: es decir, un contador de piezas con un teclado matricial que da capacidad de configuración al usuario. La figura siguiente muestra el circuito, en el cual se adjunta al contador de piezas un teclado matricial de 8 teclas, formado por dos filas con cuatro teclas cada una.



Los cuatro dígitos de la figura se deben encender de manera secuenciada para que el número se distinga correctamente. Al mismo tiempo, la lectura del teclado matricial también precisa de una secuencia de escrutinio, en la cual se comprueba, columna a columna, si se han pulsado las teclas asociadas. En consecuencia, en el circuito de la figura, se desea utilizar la secuencia de

refresco de los cuatro dígitos de siete segmentos, para efectuar, a la vez, la secuencia de escrutinio de las columnas del teclado matricial. Esta secuencia se llevará a cabo automáticamente mediante las interrupciones del Timer 2.

Las filas del teclado matricial pueden activar la entrada de interrupción /INT1 mediante la puerta AND de la figura. La secuencia situará un 0 lógico en la patilla P0.7, mientras las patillas P0.4, P0.5 y P0.6 permanecen a 1 lógico, pondrá el transistor Tr4 en saturación y encenderá el dígito correspondiente a las unidades. Si en este momento se pulsan las teclas F1 o F2, se provocará una interrupción en /INT1. Transcurrido un tiempo determinado, el 0 lógico se sitúa en el terminal P0.5, mientras los otros terminales están a 1 lógico; se pone, así, el transistor Tr3 en saturación y se enciende el dígito de las decenas. De esta manera se procede hasta que se pone un 0 lógico en la patilla P0.7 y se enciende el dígito correspondiente a los millares. En este caso, la secuencia se repetirá con una frecuencia de 1 kHz, que establecerán las interrupciones del Timer 2.

Cada vez que llega un pulso de "cuenta" se debe incrementar el valor del contador, y cuando se activa el sensor de "descuenta" se debe decrementar el contador. Si el valor del contador sobrepasa el valor 9999, éste no se incrementará, y si llega un pulso de "descuenta" cuando el valor es de 0000, el contador no se decrementará. La tecla "reset" activa la interrupción /INT0 y pone a cero el valor del contador.

Cuando se detecte que una tecla ha sido pulsada se mostrará un carácter determinado que indique, en el dígito conectado al puerto P2, la tecla pulsada. Para la tecla 'Up" se mostrará el carácter "U", para la tecla ''Rígth" se mostrará el carácter 'Y', para 'enter" el carácter "E", para "Fl" el carácter "l.", para "Down" el carácter "d", para "Left" el carácter "L", para "Clear" el carácter "C" y para "F2" el carácter "2.". El dígito conectado al puerto P2 mostrará siempre el carácter correspondiente a la Última tecla pulsada.

```
#include <reg52.h>
/*************************************************************
 Refresco de 4 dígitos de siete segmentos y de teclado matricial
*************************************************************/
// Patillas de entrada
sbit B_CUENTA=P1^2;
sbit B_DESCU=P1^3;
sbit Fila_0=P1^0;
sbit Col_1=P0^4;
sbit Col_2=P0^5;
sbit Col_3=P0^6;
sbit Col_4=P0^7;
//Patillas de salida
sbit SAL_TR1=P0^4;
sbit SAL_TR2=P0^5;
sbit SAL_TR3=P0^6;
sbit SAL_TR4=P0^7;
// variables de cuenta
unsigned char uni,dec,cen,mil;
// digito
unsigned char tecla;
/*************************************************************
 Rutina de servicio de /INT0
*************************************************************/
void RSI_Int0(void) interrupt 0 {
      uni=0;
      dec=0;
      cen=0;
      mil=0;
      P2=0;
```

```
      tecla=0;
}
/*****************************************************************
 Rutina de Servicio de /INT1
*****************************************************************/
void RSI_Int1(void) interrupt 2 {
if (!Fila_0) {//¿Ha pulsado una tecla de la fila 0?
      if (!Col_1) tecla=0x3E; //tecla UP 00111110b
      if (!Col_2) tecla=0xA0; //tecla RIGHT 01010000b
      if (!Col_3) tecla=0x79; //tecla ENTER 01111001b
      if (!Col_4) tecla=0x86; //tecla F1 10000110b
  }
else {//Si no, debe ser la fila 1
      if (!Col_1) tecla=0x5E; //tecla DOWN 01011110b
      if (!Col_2) tecla=0x38; //tecla LEFT 00111000b
      if (!Col_3) tecla=0x39; //tecla CLEAR 00111001b
      if (!Col_4) tecla=0xDB; //tecla F2 11011011b
  }
}
/*****************************************************************
 Rutina de Servicio del Timer 2
*****************************************************************/
void RSI_Timer2(void) interrupt 5 {
static char digit=0;

P0|=0x0F0;  //Apaga todos los digitos
// refresca un digito en cada interrupcion
switch (digit) {
      case 0:     {
            P0=(uni|0xF0); //Unidades en P0(4 bits altos de uni cero)
            SAL_TR4=0; //Enciende unidades. Tr4 en saturación
            break;
            }
      case 1: {
            P0=(dec|0xF0); //Decenas en P0(4 bits altos de dec cero)
            SAL_TR3=0; //Enciende decenas. Tr3 en saturación
            }
      case 2:     {
            P0=(cen|0xF0); //Centenas en P0(4 bits altos de cen cero)
            SAL_TR2=0; //Enciende centenas. Tr2 en saturación
            }
      case 3: {
            P0=(mil|0xF0); //Unidades en P0(4 bits altos de mil cero)
            SAL_TR1=0; //Enciende unidades. Tr1 en saturación
            }
      }
      digit=(++digit)&0x3; //incremento y fijo 0-3
      TF2=0; //Borra TF2
}
/*****************************************************************
 Rutina de inicio
*****************************************************************/
void inicio(void) {
      P2=0; //Apaga el dígito conectado a P2
      IT0=1;      ///INT0 activa por flanco descendente
      IT1=1;      ///INT1 activa por flanco descendente
      PX1=1;      //Asigna prioridad alta a /INTl
      EX0=1;      //Habilita interrupción de /INT0
      EX1=1;      //Habilita interrupción de /INT1
      ET2=1;      //Habilita interrupción del Timer 2
      T2CON=0;    //Timer 2 en 16 bits con autorrecarga
```

```
      T2MOD=0;          //Timer 2 (CEN=0)
      RCAP2L=0x05;      //Carga recarga cada 250 µs
      RCAP2H=0x0FF;     //en RCAP2L y RCAP2H
      TL2=0x05;
      TH2=0x0FF;
      EA=1; //Habilita bit de interrupción general
      TR2=1;      //Pone en marcha el Timer 2
}
/***********************************************
 Funcion Cuenta: incrementa contador BCD y señala alarma
***********************************************/
void Cuenta(void) {
if (uni==9) {
      if (dec==9) {
            if (cen==9)
                  if (mil==9) {;}
                  else {uni=dec=cen=0;mil++;}
            else {uni=dec=0;cen++;}
      }
      else {uni=0;dec++;}
}
else uni++;
}
/***********************************************
 Funcion Descuenta: decrementa contador BCD
***********************************************/
void Descuenta(void) {

if (uni==0) {
      if (dec==0) {
            if (cen==0)
                  if(mil==0) {;}
                  else {uni=dec=cen=9;mil--;}
            else {uni=dec=9;cen--;}
      }
      else {uni=9;dec--;}
}
else uni--;
}
/*************************************************************
 Rutina principal
*************************************************************/
void main(void) {
inicio();
while (1) {
      if (!B_CUENTA) Cuenta();
      else if (!B_DESCU) Descuenta();
      P2=tecla; //Carga digito en P2, para mostrar dígito
      }
}
```

En la rutina de “inicio” el puerto P2 se pone a cero, debido a que inicialmente está a FFH (o sea, todos sus terminales a 1 lógico); por tanto, debe ponerse a cero para que ninguno de los leds del dígito esté encendido. En la rutina, las entradas /INT0 y /INT1 se configuran activas por flanco descendente y se habilitan las interrupciones de /INT0, de /INT1 y del Timer 2. El Timer 2 se configura para que funcione como un temporizador de 16 bits con autorrecarga; para ello se colocan los bits RCLK, TCLK y CP/RL2, del registro T2CON, a 0 lógico. El Timer cuenta pulsos internos del reloj del microcontrolador, por lo que el bit C/T2 del registro T2CON se pone a 0 lógico. En el Timer 2 es necesario inhibir las interrupciones procedentes del terminal T2EX; por tanto, el bit EXEN2 también se pone a 0 lógico. Por último, para que el

temporizador funcione sólo en sentido ascendente, el bit DCEN del registro T2MOD se pone a 0 lógico.

Para que la secuencia de refresco de los dígitos sea de 1 kHz, el valor de recarga del Timer ha de ser de FF05H, pues el período de la secuencia es de 1 ms y, al haber cuatro dígitos, el tiempo que debe permanecer encendido cada dígito es de 0.250 ms. 0x0FF05 es el resultado de restar 250 (o 0x0FA) de 64k (o 0x0FFFF).

El puerto P0 inicialmente también tiene todos sus terminales en estado 1 lógico, lo que no supone ningún inconveniente, pues las puertas inversoras conectadas a los transistores Trl, Tr2, Tr3 y Tr4, aseguran que los transistores estén en corte y los dígitos permanezcan apagados.

La rutina de RSI del Timer 2 utiliza la variable "digit" como contador en base 3, y determina el dígito que se va a encender, según sea el valor de ésta. Si vale cero se enciende el dígito correspondiente a las unidades, si vale 1 se enciende el dígito de las decenas, si vale 2 se enciende el dígito de las centenas y si vale 3 se enciende el dígito de los millares. El valor de digit se actualiza con la instrucción de incremento y con la función lógica AND, de forma que cada vez que la rutina de RSI se ejecuta, la variable estática se incrementa en una unidad, excepto cuando vale 3, que se pone a cero. El contenido de las variables uni,dec,cen y mil, se fuerzan sus cuatro bits altos a 1 lógico mediante una instrucción OR, de manera que todos los dígitos estén apagados y se encienda el dígito que corresponda.

La rutina principal está formada por un bucle infinito en el cual se comprueba el estado de los sensores "cuenta" y "descuenta", y donde se pone, en el dígito conectado a P2, el carácter de la tecla pulsada en el teclado matricial.

4.3.- Temporizadores y contadores en la familia PIC16.

En el 16F877 disponemos de 3 timers, numerados del 0 al 2 y del Watch Dog Timer.

4.3.1.- Timer 0.

Es un temporizador de 8 bits que puede ser leído y escrito a través del registro TMR0 como sucede con cualquier timer en las direcciones 0x01 y 0x101. La entrada de reloj del temporizador puede seleccionarse para que sea interna o externa. Si es interna tendrá una frecuencia resultado de dividir por 4 la señal de reloj de la CPU. Como fuente externa se puede emplear la entrada RA4/T0CKI. Para seleccionar la fuente se emplea el bit T0CS (bit 5) del OPTION que se encuentra en las direcciones 0x081 y 0x0181. De forma que se está a 0 se selecciona el reloj interno y si está a 1 el reloj externo. Si se usa un reloj externo, su frecuencia como máximo será la mitad de la frecuencia de la entrada de reloj de la CPU.

Si se selecciona el reloj externo, se puede utilizar como contador de pulsos en dicha entrada. En este caso se puede seleccionar si se activa la entrada por flanco de subida o de bajada mediante el bit T0SE (bit 5) del registro OPTION situado en direcciones 0x081 y 0x0181. De forma que estando a 0 selecciona cuenta por flanco de subida y estando a 1 por flanco de bajada.

A este timer se le puede asociar un circuito divisor de la frecuencia de la entrada de reloj. Se le denomina **Prescaler**. Este circuito puede ser utilizado también por el Watch dog. En cualquier caso debe ser asignado a uno u otro. Esto se selecciona mediante el bit PSA (bit 3) del registro OPTION (direcciones 0x081 y 0x0181). De forma que si vale 0 se selecciona el Timer 0, y si vale 1 el Watch dog.

También es posible seleccionar el factor de división. Se dispone de tres bits: PS2, PS1 y PS0, (bits 2, 1 y 0) del registro OPTION (direcciones 0x081 y 0x0181) que introducen los factores de división que se muestran en la siguiente tabla:

Bits PS	Timer0	Watch dog
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

Finalmente existen los dos bits mencionados en las interrupciones que permiten generar una interrupción cada vez que se desborda el Timer0 cuando pasa de 0x0FF a 0x0. Son los bits T0IE y T0IF del registro INTCON.

4.3.2.- Watch dog Timer.

Este módulo permite inicializar la CPU (reset) cuando se rebasa el contador. Se activa por medio de los bits de configuración del microcontrolador (en el momento de la programación del chip) y permite recupera el sistema cuando se pierde el control del programa. El período con un valor de prescaler 1:1 oscila entre 7 y 33 mS, con un valor típico de 18 mS. Esta tolerancia es debida a la imprecisión del oscilador interno que depende de la temperatura y la tensión de alimentación básicamente. Este período se puede ampliar con el prescaler hasta un factor de 1:128.

4.3.3.- Timer 1.

Se trata de un temporizador de 16 bits que puede ser leído y escrito a través de los registros TMR1H y TMR1L. La entrada de reloj del temporizador también puede seleccionarse que sea interna o externa. Si se selecciona interna, disponemos de la frecuencia de reloj de la CPU dividida por 4. En caso de entrada externa se dispone de las patillas exteriores RC1/T1OSICCP2 y RC0/T1OSO/T1CKI que permiten configurar un oscilador de reloj externo.

El Bit TMR1CS (bit 1 del registro T1CON) controla el tipo de reloj empleado, de forma que si vale 0 queda seleccionado el reloj interno, si vale 1 el reloj externo. Si se usa un reloj externo, su frecuencia como máximo será la mitad de la frecuencia de la entrada de reloj de la CPU.

También con reloj externo existen dos alternativas, colocar un cristal como fuente de reloj (posible hasta unos 200 KHz) o utilizar una señal digital. En el segundo caso solo se utiliza la línea de entrada RC0/T1OSO/T1CKI. Para seleccionar una alternativa u otra, se dispone del bit T1OSCEN (bit 3 del registro T1CON). Si vale 1 se usará el cristal activando el oscilador interno, si vale 0 se usan pulsos generados externamente. En este caso puede funcionar como contador de pulsos.

Además se dispone de un bit que permite activar y desactivar el Timer: TMR1ON (bit 0 del registro T1CON). El timer dispone de un prescaler controlado por los bits T1CKPS1 y T1CKPS0 (bits 5 y 4 del registro T1CON) cuyos factores de división se muestran en la siguiente tabla:

Bits T1CKPS	Timer1
00	1:1
01	1:2
10	1:4
11	1:8

Finalmente existen los dos bits mencionados en las interrupciones que permiten generar una interrupción cada vez que se desborda el Timer1 cuando pasa de 0x0FFFF a 0x0. Son los bits TMR1IE y TMR1IF necesitando además que estén activados los bits que habilitan las interrupciones de los periféricos (PEIE) y el general (GIE).

4.3.4.- Timer 2.

Es un temporizador/contador de 8 bits Puede ser leído y escrito a través del registro TMR2. La entrada de reloj del temporizador es interna, concretamente la frecuencia de reloj CPU dividida por 4. Se dispone de un bit que permite activar y desactivar el Timer, es el TMR2ON (bit 2 de T2CON).

El timer dispone de un prescaler controlado por los bits T2CKPS1 y T2CKPS0 (bits 1 y 0 de T2CON) cuyos factores de división que se muestran en la siguiente tabla:

Bits T2CKPS	Timer2
00	1:1
01	1:4
1x	1:16

La salida del contador puede ser conectada a un **"postscaler"** (un divisor cuya entrada es la salida del contador de 8 bits y cuya salida es la interrupción), que dispone de 4 bits de control T2OUTPS3:T2OUTPS0 cuya salida pasa a controlar el flag de interrupción asociado a este Timer. Los factores de división se muestran en la siguiente tabla:

Bits T2OUTPS	Salida de interrupción
0000	1:1
0001	1:2
0010	1:3
.	.
.	.
.	.
1101	1:14
1110	1:15
1111	1:16

Finalmente existen los dos bits mencionados en las interrupciones que permiten generar una interrupción cada vez que se desborda el Timer2 cuando pasa de 0x0FFH a 0x0. Son los bits TMR2IE y TMR2IF necesitando además que estén activados los bits que habilitan las interrupciones de los periféricos (PEIE) y el general (GIE).