

## CAPITULO 8

### Interrupciones

#### 8.1. INTRODUCCION

La comunicación asíncrona de los sistemas periféricos con la CPU, en ambos sentidos, se puede establecer de dos maneras fundamentales:

- a) Consultas (polling): Se comprueban cíclicamente, mediante instrucciones del programa los registros de estado de los dispositivos de E/S. Unas líneas de diálogo (handshake) establecen el protocolo de comunicación.

Así, en la Figura 8.1 se muestra cómo se establece el diálogo entre un sistema microcomputador o microcontrolador y dos periféricos mediante el procedimiento de consultas. Primero el programa pregunta si el bit del Puerto 0 está activo, esto quiere decir que el Periférico 1 solicita la atención del sistema, en ese caso la CPU atiende al Periférico 1. Cuando el proceso termina, el programa activa el bit 1 del Puerto 0 para indicar, al Periférico 1, que el proceso ha finalizado. El mismo procedimiento para el Periférico 2.

Las desventajas de este método son:

- En cada ciclo de programa éste tiene que interrogar los bits de consulta.
- Al periférico se le atiende después de realizar la consulta y no cuando solicita la intervención de la CPU.

- b) Interrupción: Servicio directo entre periféricos y CPU, siempre que ésta desee establecer el diálogo. Este servicio tiene la característica de la inmediatez, pueden eliminarse total o parcialmente los ciclos de consulta y permite inhibir la interrupción cuando se considera que es «inoportuna» y, por tanto, perjudicial para la marcha del proceso.

Esta forma de trabajo es inherente al control de procesos en tiempo real.

Así, en la Figura 8.2, cuando el Periférico 1 requiere la intervención del sistema microcomputador o microcontrolador activa la interrupción INT1, si está habilitada el sistema atenderá la petición y ejecutará el proceso correspondiente de atención en la RUTINA 1. El mismo procedimiento se establecería para el Periférico 2.

La importancia de las interrupciones nace de la necesidad de ejecutar un subproceso en el instante preciso, y por tanto se considera su intervención «urgente». Cuando termina la ejecución de este subproceso, la CPU vuelve al programa principal, continuando su tarea cíclica justo donde la dejó.

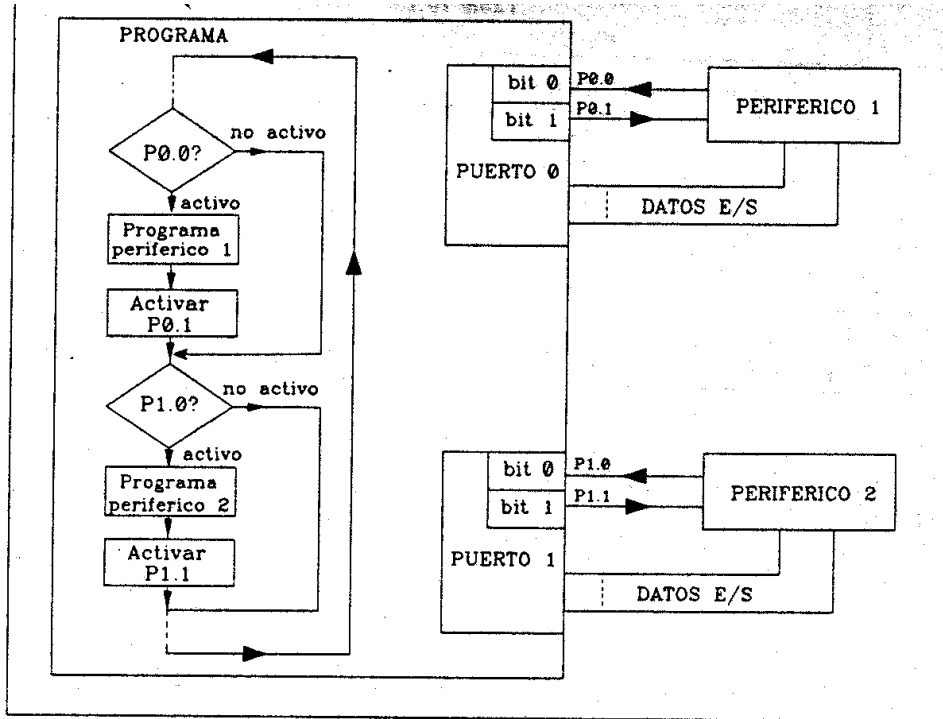


Figura 8.1

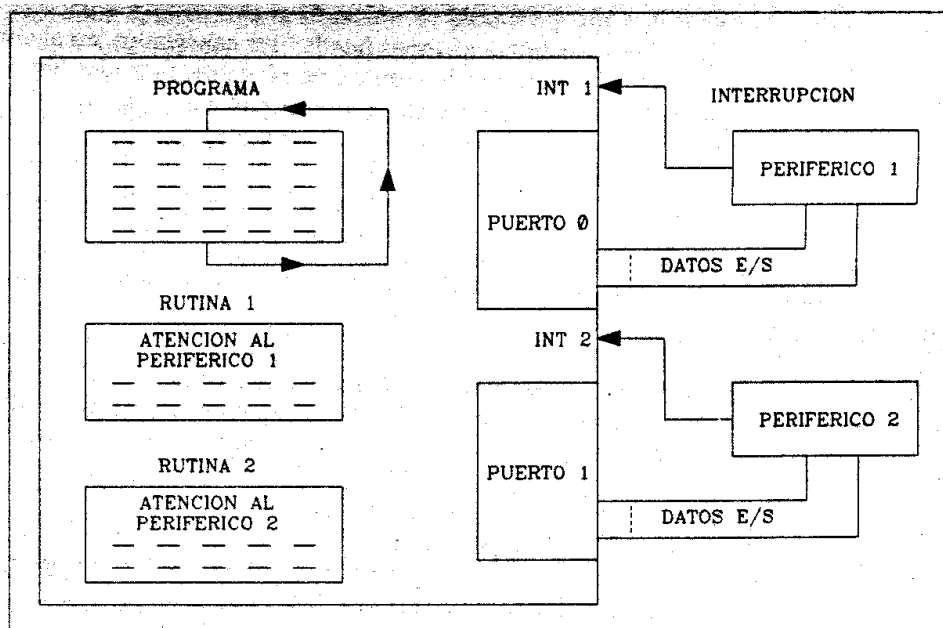


Figura 8.2

Una interrupción puede ser iniciada:

- Por un periférico o circuito externo ajeno al microprocesador o microcontrolador a través de los pines específicos del mismo. En estas condiciones la interrupción es externa.
- Dentro del propio chip microprocesador o microcontrolador. Entonces la interrupción es interna.

Toda interrupción aceptada conduce a la ejecución de un subprograma específico cuya dirección de comienzo se indica en las posiciones de memoria de una tabla, que recibe el nombre de tabla de vectorización.

## 8.2. ASPECTOS GENERALES y TIPOS DE INTERRUPCIONES

El Microcontrolador 8052 tiene seis interrupciones, mientras que el 8051 tiene cinco. La Figura 8.3 muestra los distintos tipos de interrupciones, señalando la falta de los indicadores que activan TF2 y EXF2 en el 8051 por no tener implementado el Timer 2.

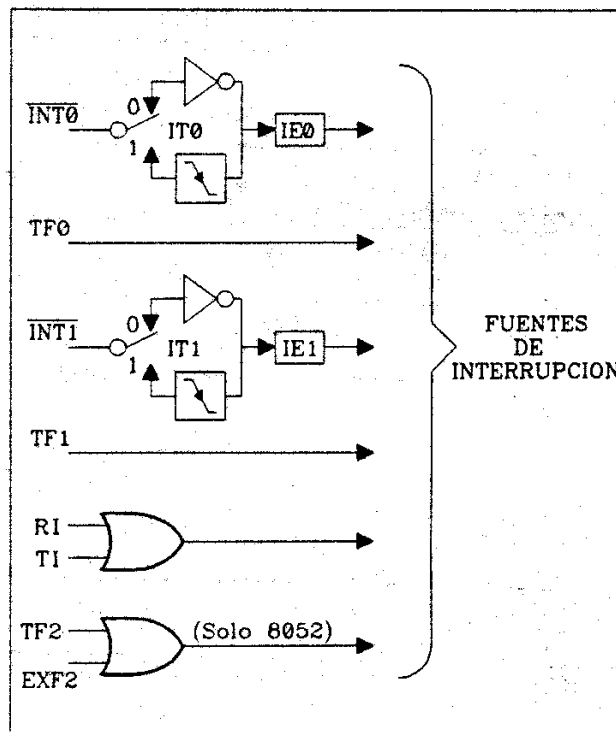


Figura 8.3

Los bits de bandera que generan las interrupciones pueden ser cancelados. en algunas interrupciones por hardware cuando éstas son vectorizadas; no obstante todos los bits pueden cancelarse por software escribiendo ceros en el registro correspondiente.

Cada una de estas fuentes de interrupción pueden ser individualmente habilitadas o inhabilitadas poniendo á «uno» o a «cero» el bit correspondiente del registro IE (Interrupt Enable Register) perteneciente a SFR (Special Function Register) (Cuadro 8.1).

**Cuadro 8.1**

IE (INTERRUPT ENABLE REGISTER)								
	b <sub>7</sub>	b <sub>6</sub>	b <sub>5</sub>	b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>	b <sub>0</sub>
	EA	X	ET2	ES	ET1	EX1	ET0	EX0
BIT	NOMBRE Y COMENTARIO							
b <sub>0</sub>	<b>EX0</b> : - Si EX0 = 1 habilita interrupción externa INT0 - Si EX0 = 0 inhabilita.							
b <sub>1</sub>	<b>ET0</b> : - Si ET0 = 1 habilita interrupción del <i>Timer 0</i> . - Si ET0 = 0 inhabilita.							
b <sub>2</sub>	<b>EX1</b> : - Si EX1 = 1 habilita interrupción externa INT1. - Si EX1 = 0 inhabilita.							
b <sub>3</sub>	<b>ET1</b> : - Si ET1 = 1 habilita interrupción del <i>Timer 1</i> . - Si ET1 = 0 inhabilita.							
b <sub>4</sub>	<b>ES</b> : - Si ES = 1 habilita interrupción del puerto serie. - Si ES = 0 inhabilita.							
b <sub>5</sub>	<b>ET2</b> : - Si ET2 = 1 habilita interrupción por sobrepasamiento o captura del <i>Timer 2</i> . - Si ET2 = 0 inhabilita.							
b <sub>6</sub>	Reservada.							
b <sub>7</sub>	<b>EA</b> : - Si EA = 1 habilita individualmente a todas las interrupciones que en este registro estan a <i>uno</i> . - Si EA = 0 no reconoce ninguna interrupción.							

Fuente de interrupción	Flag que activa
Externa 0 ..... INT0 .....	..... IE0
Timer 0 ..... TIMER0 .....	..... TF0
Externa 1 ..... INT1 .....	..... IE1
Timer 1 ..... TIMER1 .....	..... TF1
Puerto serie ..... RI .....	..... RI
Puerto serie ..... TI .....	..... TI
Timer 2 ..... TIMER2 .....	..... TF2
Timer 2 Externa 2 ..... T2EX .....	..... EXF2

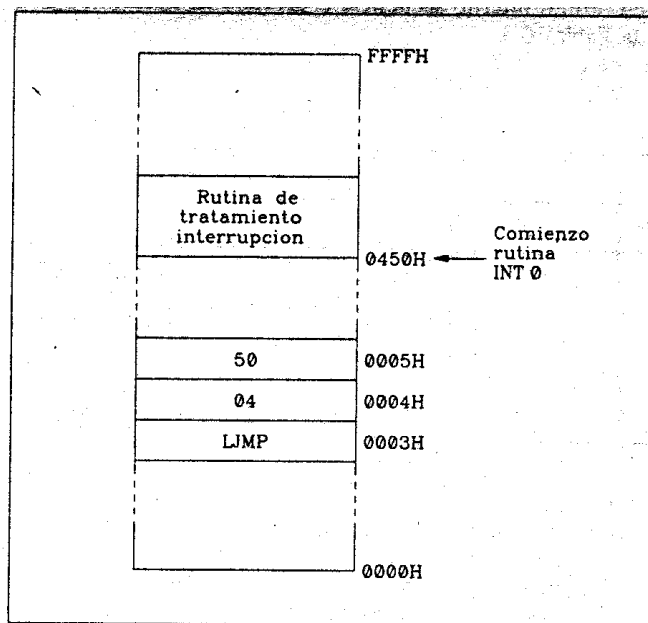
**Tabla 8.1**

Las interrupciones, hacen uso de una tabla, tabla de vectorización, en la que se reservan 8 localidades para cada una de las interrupciones en la memoria de programas (Tabla 8.2).

Tabla de vectorizaciones	
Fuente	Dirección
$\overline{\text{INT0}}$	0003H
TIMER0	000BH
INT1	0013H
TIMER1	001BH
RI + TI	0023H
TIMER2 + T2EX	002BH

**Tabla 8.2**

En estas 8 direcciones se puede escribir el programa de atención a la interrupción o, como es habitual, mediante un `JMP<dir>` se desvía a una zona de memoria de programas lo suficientemente extensa para albergar la rutina de tratamiento de la interrupción detectada. Así por ejemplo, si se habilita la interrupción exterior (bit EX0 correspondiente al pin INT0) y el comienzo del programa de tratamiento de la rutina está ubicado en la dirección 0450H, la tabla de vectorización contendrá en la localidad 0003H el código de operación de la instrucción de salto (y la parte alta de la dirección del salto si es un salto absoluto `AJMP`), en la siguiente localidad 0004H el byte alto de la dirección de salto y en la dirección 0005H el byte bajo de la dirección del salto (Figura 8.4).



**Figura 8.4**

10.3. EL PROCESO DE INTERRUPCION

### **8.3. EL PROCESO DE INTERRUPCION EN LOS MICROCONTROLADORES 8052-/8051**

Los indicadores de interrupción son muestreados en el estado 5, fase 2 (S5P2) de cada ciclo máquina. Las muestras son escrutadas durante el siguiente ciclo máquina. El sistema de interrupciones del microcontrolador genera un LCALL al apropiado vector de interrupciones. Esta situación se produce salvo que:

- Una interrupción de igualo mayor nivel de prioridad esté en ese momento en proceso.
- No haya finalizado la instrucción que en ese momento está procesándose.
- La instrucción en proceso es una RETI (retorno de interrupción) o se esté produciendo un acceso a los registros IE o IP; entonces asegura que una instrucción más, al menos será ejecutada antes de que la interrupción sea procesada.

El ciclo de «escrutinio» se repite con cada ciclo máquina y los valores escrutados son los que corresponden a la activación de la interrupción presente en el S5P2 del ciclo máquina anterior.

Puede suceder que una bandera de interrupción sea activada y no pueda responder el sistema por encontrarse en una de las situaciones de bloqueo comentadas anteriormente.

En estas condiciones puede suceder lo siguiente:

- a) Que desaparezca la situación de bloqueo y que la solicitud de interrupción siga activa. En este caso la interrupción será tratada por el microcontrolador.
- b) Que desaparezca la situación de bloqueo, pero que en ese intervalo la señal de solicitud de interrupción se haya desactivado. En ese caso la interrupción será denegada y deberá ser solicitada nuevamente.

El ciclo completo es mostrado en la Figura 8.5, considerando las condiciones idóneas, es decir, cuando el ciclo máquina C2 coincide con el último ciclo de instrucción que se está ejecutando y que esta instrucción no sea una RETI o un acceso a los registros IE o IP.

Puede suceder también que una interrupción de un nivel de prioridad mas alto se active antes del estado S5P2 del ciclo de máquina etiquetado C3 en la Figura 8.5; de acuerdo con lo dicho anteriormente, esta interrupción será vectorizada durante los ciclos C5 y C6, sin que ninguna instrucción de la rutina de prioridad más baja haya sido ejecutada.

Con relación al borrado de los indicadores, una vez servida la rutina de interrupción, se ha de decir que en algunos casos, el borrado es automático, aspecto denominado como «borrado por hardware»; en el otro caso, es el programador el que debe estar pendiente de su borrado; modalidad de «borrado por software».

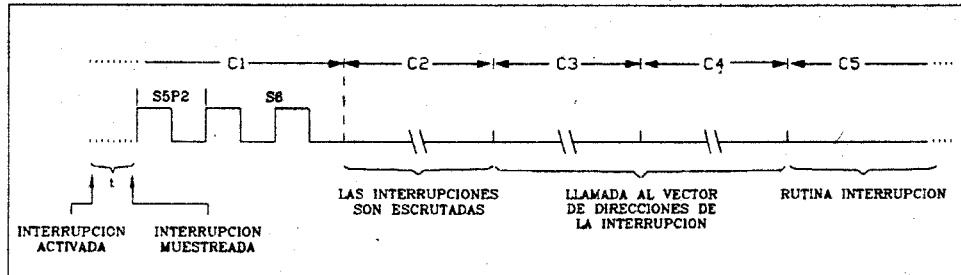


Figura 8.5

No se produce borrado por hardware en las interrupciones del puerto serie y del Timer 2.

Los flags de las interrupciones exteriores (INT0 e INT1) son borrados por hardware cuando se activan por flanco. Si están activadas por «nivel», éste debe volver a su estado de «no activación» para que «salga» de la rutina de interrupción.

Una vez validada la interrupción y producido el salto a la rutina de servicio, la CPU guarda en la memoria pila (stack) el contenido del contador de programa (PC = Program Counter) y no salvaguarda el registro de estado (PSW).

La rutina de interrupción debe finalizar con la instrucción RETI, que informa a la CPU de que la rutina ha finalizado. A renglón seguido se recuperan de la memoria de pila los dos bytes de dirección del programa principal, instrucción siguiente a la que en proceso recibió la interrupción y que concluyó antes de pasar al proceso de tratamiento. Estos dos bytes recargados en el contador de programa (PC) permiten que se siga ejecutando el programa principal, sin más que sufrir una relativa pequeña pérdida de tiempo.

#### 8.4. NIVELES DE PRIORIDAD DE LAS INTERRUPCIONES

Cada fuente de interrupción puede programarse individualmente en dos niveles de prioridad, poniendo a «1» o a «0» su bit correspondiente en el registro IP (Cuadro 8.2) del banco de registros SFR. Como se puede observar en la figura, un «1» en el bit correspondiente a la interrupción la sitúa en el nivel de prioridad más «alto» y un «0» en el nivel más «bajo».

Los bits b6 y b7 no están asignados, y lo mismo pasa con el b5 para el Microcontrolador 8051, puesto que no tiene Timer2.

Una interrupción con nivel de prioridad bajo puede ser interrumpida por otra de nivel de prioridad alto, pero no por otra de nivel de prioridad bajo. Una interrupción de prioridad alta no puede ser interrumpida por otra fuente de interrupción, salvo por la interrupción externa RESET.

**Cuadro 8.2**

IP (INTERRUPT PRIORITY REGISTER)								
	b7	b6	b5	b4	b3	b2	b1	b0
	X	X	PT2	PS	PT1	PX1	PT0	PX0
BIT	NOMBRE Y COMENTARIO							
b0	PX0 : - Si PX0 = 1 define alta prioridad a interrupción INTO							
b1	PT0 : - Si PT0 = 1 define alta prioridad a interrupción <i>Timer 0</i> .							
b2	PX1 : - Si PX1 = 1 define alta prioridad a interrupción INT1.							
b3	PT1 : - Si PT1 = 1 define alta prioridad a interrupción <i>Timer 1</i> .							
b4	PS : - Si PS = 1 define alta prioridad a interrupción puerto serie.							
b5	PT2 : - Si PT2 = 1 define alta prioridad a interrupción <i>Timer 2</i> .							
b6	Reservado							
b7	Reservado							

Si dos peticiones de interrupción de distintos niveles de prioridad son recibidas simultáneamente la interrupción de nivel de prioridad alto es servida. Si la petición corresponde a interrupciones del mismo nivel de prioridad y se producen simultáneamente, un escrutinio interno da secuencia al servicio de una manera predeterminada según se muestra en la tabla 8.3

**Tabla 8.3**

Prioridades en el mismo nivel			
Prioridad			Fuente
(más alta) . . . . .	1	$\overline{\text{INT0}}$	(Int. EXTER0)
	2	TIMER0	(Int. TIMER0)
	3	INT1	(Int. EXTER1)
	4	TIMER1	(Int. TIMER1)
	5	RI + TI	(Int. PUERTO SERIE)
(más baja) . . . . .	6	TIMER2 + T2EX	(Int. TIMER2 + FLANCO)

Así, dentro del mismo nivel de prioridad, hay una segunda estructura de prioridades determinada por el orden de escrutinio según la tabla anterior. Esta prioridad, dentro del mismo nivel, solamente es utilizada para resolver estas peticiones simultáneas.

La Figura 8.6 resume y esquematiza los aspectos más importantes que han sido tratados con relación al tema:

- $\overline{INT0}$  e  $\overline{INT1}$  pueden ser activas por nivel o flanco.
- El registro IE (Interrupt Enable) permite «inhabilitar» todas las interrupciones o «habilitarlas» individualmente.
- El registro IP (Interrupt Priority) puede disponer cada interrupción, individualmente, en la «cola» de interrupciones de alta o baja prioridad.

Hasta aquí se ha tratado el tema de las interrupciones de una forma general. A continuación se estudia cada uno de los tipos de interrupciones, exceptuando las de los puertos de comunicaciones.

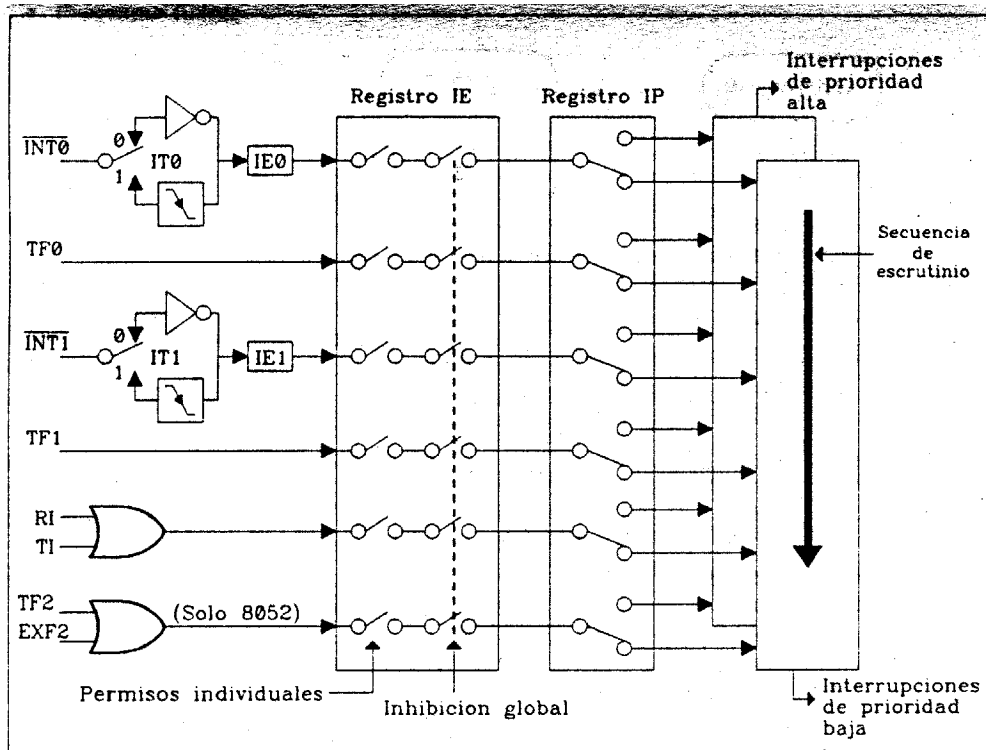


Figura 8.6

## 8.5. INTERRUPCIONES EXTERNAS (INT0) E (INT1)

Estas interrupciones externas, como se pueden observar en la Figura 8.6, pueden ser activas por nivel o por flanco descendente, dependiendo del estado del bit 0 (IT0) para INT0 y del bit 2 (IT1) para INT1 en el registro TCON (Cuadro 7.2). Los flags que activan estas interrupciones son los bit 1 (IE0) y bit 3 (IE1), también del registro TCON.

Cuando se genera una interrupción externa por flanco del tipo INT0 o INT1, el indicador se activa (se pone a «1») y se desactiva (se pone a «0»), cuando el servicio de interrupciones vectoriza para la ejecución de la rutina, entonces se dice que lo ha borrado el hardware del microcontrolador. Si la interrupción ha sido generada por nivel, la señal externa activa el indicador y la puesta a «cero» del mismo se hace por software, mediante una instrucción que pone el indicador a cero actuando sobre el propio bit en el registro correspondiente, naturalmente, una vez que la señal de activación de la interrupción puesta a nivel bajo se desactiva pasando a nivel alto.

Para ilustrar el funcionamiento y manejo de las interrupciones se propone una serie de ejercicios significativos para cada tipo de interrupción. En este apartado se muestran las interrupciones externas y prioridades.

### Ejercicio 8.1. Interrupciones externas. DESPLA0.ASM

- **Descripción del programa**

El programa consta de un cuerpo principal que pretende simular el funcionamiento de la pantalla de luces deslizantes del «Auto Increible». Un bit se desplaza de izquierda a derecha y de derecha a izquierda y excitará 8 leds situados a la salida del Puerto 0 (P0). Una rutina de temporización permite modificar la frecuencia de desplazamiento del bit.

En estas condiciones se tiene el efecto referido, pero cuando por el pin 12 ( $\overline{\text{INT0}}$ ), interrupción externa del microcontrolador, se produce un flanco descendente, y dado que está habilitado este tipo de interrupción, el microcontrolador ejecuta una rutina, asociada a la interrupción, cuyos efectos externos son hacer «parpadear» diez veces y simultáneamente todos los leds del Puerto 0 (P0) para, una vez concluida esta secuencia, continuar con el programa principal del bit deslizante.

A continuación se indican, someramente, las operaciones que van a tener lugar de una manera secuenciada:

--Cuando el microcontrolador recibe la interrupción y ésta es aceptada, por estar habilitada (IE), el contador de programas (PC) no se incrementa para ejecutar la

siguiente instrucción, sino que consulta la tabla de vectorizaciones en la localidad correspondiente al tipo de interrupción.

--El contador de programas salta a ejecutar la rutina situada en la dirección señalada en la tabla de vectorización.

-Antes de proceder a la ejecución de la rutina asociada a la interrupción, guarda en la pila el PC de la instrucción siguiente en la que fue interrumpido el sistema para, una vez concluida la rutina de interrupción, conocer la instrucción que le correspondía ejecutar y permitir, de esta manera, no romper la secuencia del programa principal.

-En este punto se recuerda al usuario que guarde en la pila el registro de estado (PSW) por si es modificado por la rutina y preste atención a los registros y variables que son utilizados en común y no se encuentre con que los valores que utilizaba en el programa principal han sido modificados por la rutina, lo que rompería la secuencia del programa principal. En este caso, proceda a guardarlos, justo al comienzo del programa de la rutina, ya recuperarlos antes de que ésta concluya.

-El regreso de la rutina se efectúa colocando al final de la misma, y justo después de recuperar los valores de los registros y variables de la pila, la instrucción RETI.

A continuación se muestra el circuito (Fig. 8.7), el mapa de memoria, y el programa fuente del ejercicio.

*Nota: Es importante Que el interruptor o pulsador utilizado para producir la interrupción sea «antirrebote» (véase Figura 6.6).*

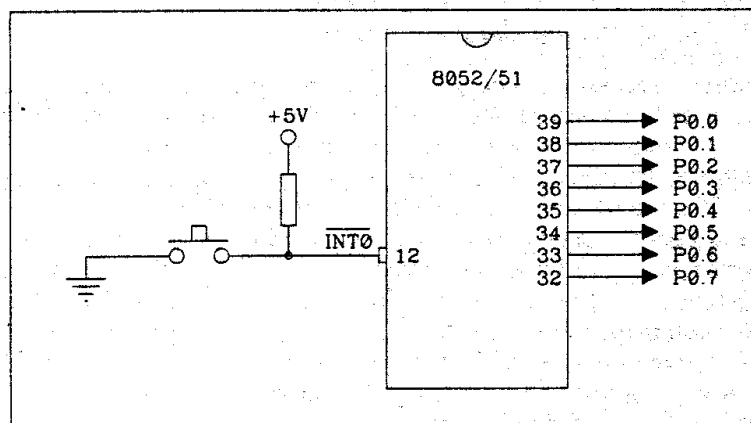
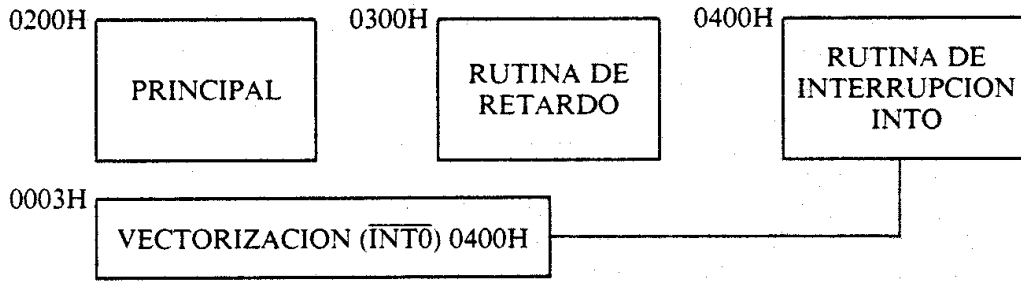


Figura 8.7

• Mapa de memoria



• Configuración de registros:



• Programa Fuente:

```

; DESPLA0.ASM
; Desplaza un bit de izda. a dcha. en el Puerto 0 (P0)
;
RETARDO EQU 0300H ; Comienzo de la RUTINA de RETARDO.
INT0 EQU 0400H ; Comienzo de la RUTINA de INTERRUPCION.
;

ORG 0003H ; Vectorización de la RUTINA DE INTERRUPCION (INT0).
JMP INT0 ; Llamada a la RUTINA DE INTERRUPCION.
;
ORG 0200H
MOV TCON, #01H ; Se programa la INTO por flanco.
MOV IE, #81H ; Habilitada interrupción INTO.
MOV A, #00H
SETB C ; Se pone el CARRY a 1.
SALO: RLC A ; Rotación de un bit ala izda.
    
```

```

MOV P0, A ; Se escribe en el Puerto 0 (P0).
CALL RETARDO ; Se llama a la rutina de RETARDO.
JNB A, 7, SAL0 ; Se testea el bit 8. Si es < > 0 se comienza el
; desplazamiento a la dcha.

SAL1: RRC A
MOV P0, A
CALL RETARDO
JNB A, 0, SAL1
JMP SAL0
;

ORG 0300H ; Comienzo de la rutina de RETARDO.
PUSH A ; Se guarda el ACC en el «stack» (PILA).
MOV A, R0
PUSH A
MOV R0, #04H
SAL2: MOV R1, #20H
SAL3: MOV R2, #FFH
SAL4: DJNZ R2, SAL4
DJNZ R1, SAL3
DJNZ R0, SAL2
POP A
MOV R0, A
POP A ; Se recupera ACC del «stack».
RET
;

ORG 0400H ; Comienzo de la rutina de interrupción INT0.
; pone en intermitencia todos los bits del P0.

PUSH A
MOV A, PSW
PUSH A ; Se guarda en el «stack» el registro de estado.
MOV A, R0
PUSH A ; Se guarda. también, el registro R0.
SAL5: MOV R0, #0AH
MOV P0, #FFH
CALL RETARDO
MOV P0, #00H
CALL RETARDO
DJNZ R0, SAL5
POP A ; Se recuperan los registros del «stack».
MOV R0, A
POP A
MOV PSW, A
POP A
RETI
END

```

## Ejercicio 8.2. Prioridades. PRIOR0.ASM

Se trata de ilustrar con un ejercicio el manejo del registro de prioridades en las interrupciones (IP).

Puesto que el usuario está familiarizado con el funcionamiento del ejercicio anterior, se va a utilizar el mismo para esta demostración, con una interrupción más, la correspondiente a la entrada exterior activa por flanco descendente (T2EX), asociada al Timer 2 (ver Figura 7.8).

Cuando se active esta interrupción se observará un cambio en la secuencia de las luces; si el programa principal genera un deslizamiento de las luces y la primera rutina (INT0) produce una secuencia de intermitencia de todos los bits del Puerto 0, la ejecución de esta rutina (T2EX) produce el intercambio alternativo de los nibbles bajo y alto del Puerto 0, como se indica a continuación:

	POH				POL			
	b7							b0
Configuración 1	1	1	1	1	0	0	0	0
Configuración 2	0	0	0	0	1	1	1	1
Configuración 3	1	1	1	1	0	0	0	0
	.....							
	.....							

Así estará oscilando entre los dos estados hasta ocho veces.

- **Esquema del circuito (Figura 8.8)**

Como se puede observar en la figura, el pulsador pone en marcha simultáneamente las dos interrupciones INT0 y T2EX, ambas activas, como se podrá ver en la «inicialización» en flanco descendente. En función del valor escrito en el registro de prioridades (IP) se estudian dos opciones:

- Que se ejecute la interrupción  $\overline{\text{INT0}}$  y posteriormente T2EX, pulsando una sola vez.
- Que se invierta el orden de ejecución.

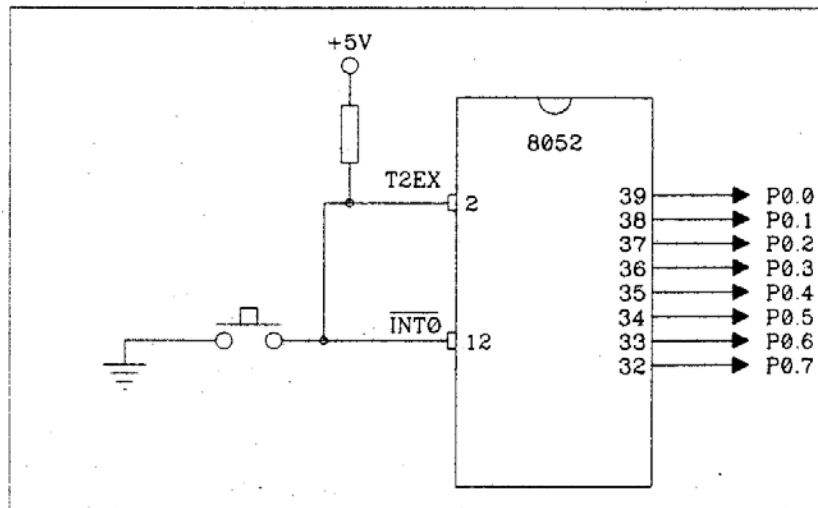
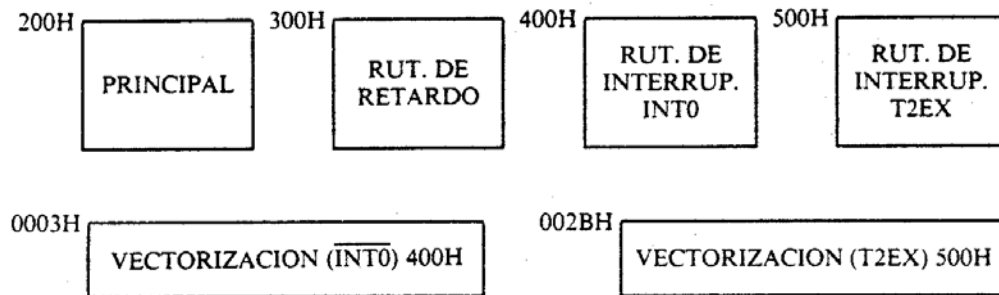


Figura 8.8

• **Mapa de memoria**



• **Algoritmo**

Como se ha señalado antes, han de observarse, de momento, dos opciones a) y b). Para que se cumpla la primera opción, no es necesario programar el registro de prioridades (IP), puesto que, por defecto, según se estudió, INT0 tiene mayor prioridad que T2EX, por esta razón, en este programa:

$$IP \leftarrow 00H$$

Para que se cumpla la segunda opción, a T2EX se le da la prioridad «alta» ya INT0 la prioridad «baja», es decir:

```

PT2 ← 1 |
          | IP ← 20H
PX0 ← 0 |

```

- **Programa fuente**

```

; PRIOR0.ASM.
; Desplaza un bit de izda. a dcha. en el Puerto 0 (P0) ,
; además, presenta otros programas alternativos
;
RETARDO EQU 0300H ; Comienzo de la RUTINA de RETARDO.
INT0 EQU 0400H ; Comienzo de la RUTINA de INTERRUPCION.
T2EX EQU 0500H ; Comienzo de la RUTINA de INTERRUPCION (T2EX).
;
-----
ORG 0003H ; Vectorización de la RUTINA de INTERRUPCION
; (INT0).
JMP INT0 ; Llamada a la RUTINA DE INTERRUPCION.
;
-----
ORG 002BH ; Vectorización de RUTINA T2EX
JMP T2EX
;
-----
ORG 0200H ; Programa principal.
MOV TCON, #01H ; (TCON←01H flanco); (TCON←00H nivel).
MOV T2CON, #08H ; Se programa la T2EX por flanco.
MOV IE, #A1H ; Habilitada interrupción INT0 y T2EX.
MOV IP, #00H ; Define prioridad. IP= 0 por defecto.
; IP= 20H alta prioridad para T2EX.
MOV A, #00H ; Pone a "0" ACC.
SETB C ; Se pone el CARRY a "1".
SAL0: RLC A ; Rotación de un bit a la izda.
MOV P0, A ; Se escribe en el Puerto 0 (P0).
CALL RETARDO ; Se llama a la rutina de RETARDO.
JNB A.7, SAL0 ; Se testea el bit 8. Si es < > 0 se comienza el
; desplazamiento a la dcha.
SAL1: RRC A ; Rotación de un «bit» a la dcha.
MOV P0, A ; Se escribe en el (P0).
CALL RETARDO
JNB A.0, SAL1 ; Salta si el bit ACC.0 es <> de «0».
JMP SAL0
;
-----
ORG 030H ; Comienzo de la rutina de RETARDO.
PUSH A ; Se guarda el ACC en el «stack» (PILA).
MOV A, R0
PUSH A
MOV A, PSW
PUSH A

```

```

MOV R0, #15H ; Rutina comentada en el Ejercicio 5.5.
SAL2: MOV R1, #20H
SAL3: MOV R2, #FFH
SAL4: DJNZ R2, SAL4
      DJNZ R1, SAL3
      DJNZ R0, SAL2
      POP A
      MOV PSW, A
      POP A
      MOV R0, A
      POP A ; Se recupera ACC del «stack».
      RET
;
;-----
ORG 0400H ; Comienzo de la rutina de interrupción INTO
; poniendo en intermitencia todos los bits del P0.
PUSH A
MOV A, PSW
PUSH A ; Se guarda en el «stack» el registro de estado.
MOV A, R0
PUSH A ; Se guarda, también, el registro R0.
MOV R0, #0AH ; Contador de eventos.
SAL5: MOV P0, #0FFH
      CALL RETARDO
      MOV P0, #00H
      CALL RETARDO
      DJNZ R0, SAL5
      POP A ; Se recuperan los registros del «stack».
      MOV R0, A
      POP A
      MOV PSW, A
      POP A
      RETI
;
;-----
ORG 0500H ; Comienzo de la rutina de interrupción (T2EX).
ANL T2CON, #BFH ; Se borra el flag EXF2 por «soft».
PUSH A
MOV A, PSW
PUSH A
MOV A, R0
PUSH A ; Hasta aquí guardar registros en el «stack».
MOV R0, #08H ; Contador de eventos.
SAL6: MOV P0, #0F0H
      CALL RETARDO
      MOV P0, #0FH
      CALL RETARDO
      DJNZ R0, SAL6
      POP A ; Se recuperan los registros del «stack».
      MOV R0, A
      POP A
      MOV PSW, A
      POP A
      RETI
;
      END

```

En este ejercicio es interesante comprobar, además de las interrupciones simultáneas, los siguientes aspectos:

- En el caso de que se esté ejecutando una interrupción, ¿qué sucederá si se activa otra de menor o igual prioridad ?

-Lógicamente. la segunda interrupción no interrumpirá la ejecución de la primera, pero será ejecutada, una vez finalizada la primera, porque la memoriza en la cola de ejecución de rutinas.

- ¿Qué sucederá si ejecutando el programa de la rutina de interrupción se activa otra de mayor prioridad?

-Abandona la rutina de interrupción para ejecutar la segunda, y una vez finalizada ésta regresará a completar la ejecución de la primera.

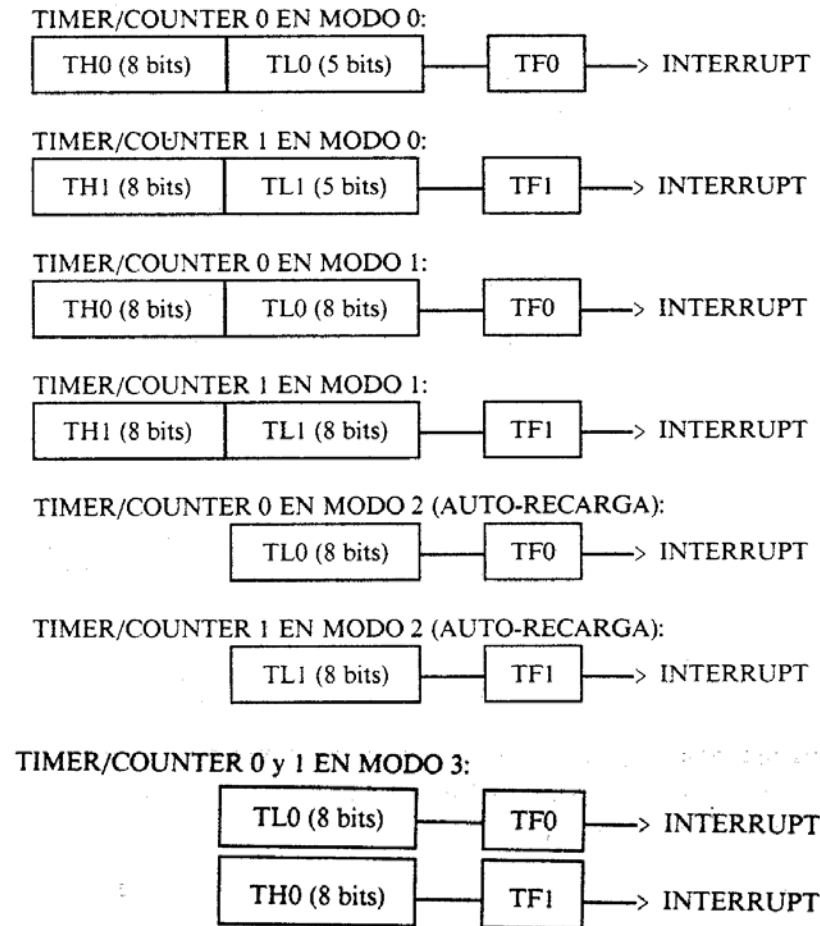
Para comprobar este comportamiento se deberá independizar las interrupciones colocando para cada entrada un pulsador.

## **8.6. INTERRUPCIONES INTERNAS PRODUCIDAS POR EL TIMER 0 y 1**

Las interrupciones de los Timers 0 y 1 son generadas por TF0 y TF1 respectivamente (Figura 8.3), al producirse un desbordamiento en el registro de conteo del Timer correspondiente.

Con la intención de servir de repaso y a modo de resumen se muestran a continuación los registros de conteo, los indicadores que son afectados según el Timer, y los modos de trabajo.

En cualquier caso, el flag activo es borrado por hardware cuando sea vectorizada la rutina de atención a la interrupción.



### Ejercicio 8.3. Interrupción Timer 1 en Modo 1 (16 bits). DESPLA1.ASM

Este ejercicio produce el desplazamiento de un bit a la izda. y se visualiza en el Puerto 0. El desplazamiento es siempre en la misma dirección. La velocidad de desplazamiento del bit depende del valor que el usuario le ponga en el Puerto P1 con los ocho conmutadores, según el peso de los bits.

- **Objetivos**

- Repasar la programación de los Timers.
- Ver cómo se activa una interrupción al producirse el «desbordamiento» en los Timers.
- Observar el mecanismo de la vectorización.

En este caso se utiliza el Timer 1, pudiendo sustituirse sin ningún inconveniente por el Timer 0.

- **Esquema del circuito (Figura 8.9)**

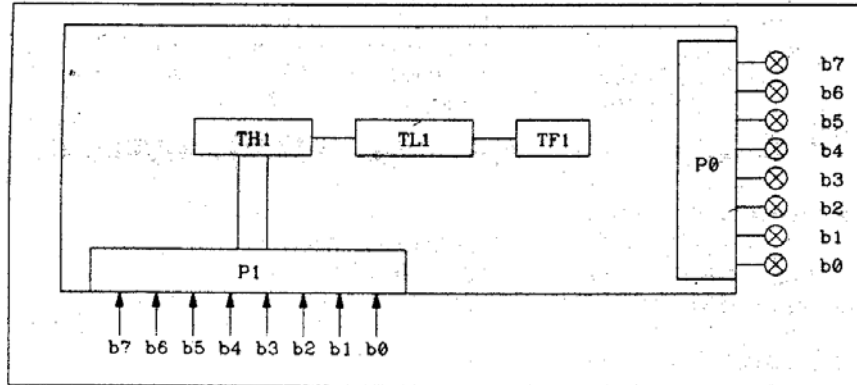


Figura 8.9

- **Programa fuente**

```

; DESPLA1.ASM.
; Desplaza un bit en el puerto P0, con velocidad variable en
; función del dato leído en el puerto de entrada P1
;
ORG 001BH
JMP 030H          ; Vector de la interrupción Timer 1.
;
ORG 0200H
MOV TMOD, #10H   ; Timer 1 como «temporizador» en Modo 1.
MOV IE, #88H     ; Validación de interrupción Timer 1.
MOV IP, #08H     ; Prioridad para el Timer 1 (no es necesaria).
MOV A, #01H      ; Se carga el ACC con 01H.
MOV TH1, #00H    ; Se carga la parte alta del Timer 1.
MOV TL1, #00H    ; Se carga la parte baja del Timer 1.
MOV TCON, #40H   ; Se arranca el Timer 1.
SAL0:
NOP
JMP SAL0         ; Se cierra en un bucle que «no hace nada».
;
;-----
ORG 0300H        ; Rutina de desplazamiento y lectura de P1
RL A             ; Desplazamiento a la izq. del ACC.
MOV P0, A       ; Saca dato por el P0
MOV TL1, P1     ; Lee P1
MOV TH1, P1
RETI
;
END

```

## 8.7. INTERRUPCIONES INTERNAS DEL TIMER 2 (sólo 8052)

En el microcontrolador 8052 la interrupción del Timer 2 puede producirse por dos caminos, según se active TF2 o/y EXF2. En el Ejercicio 8.2 se estudió la interrupción exterior por flanco descendente (T2EX), dentro del contexto de las prioridades. Ahora le toca el turno al análisis del funcionamiento de la interrupción producida por el Timer/ counter 2.

El flag activado por el Timer 2 deberá ser borrado por software, accediendo previamente al registro T2CON (Cuadro 7.3 y Figura 7.8).

### Ejercicio 8.4. El Timer 2 como generador de interrupciones

Este ejercicio es una variante del Ejercicio 8.1 (DESPLA0.ASM). En él se trata de variar la frecuencia de desplazamiento, según los datos fijos almacenados previamente en una tabla de tal forma que el desplazamiento se haga “rápido” en los extremos y “lento” en el centro (Figura 8.10). El Ejercicio 6.4 también utilizaba una tabla.

El valor en hexadecimal que figura entre los bits del puerto de salida (P0) (ver Figura 8.10) es el que se recargará en los registros contadores TL2 y TH2; estos valores deberán ser cargados previamente en una tabla de datos.

- **Objetivos**

- Estudiar las interrupciones del Timer 2.
- Cargar los registros de conteo con los datos procedentes de una tabla.
- Observar el borrado del flag de interrupción del Timer 2 por software

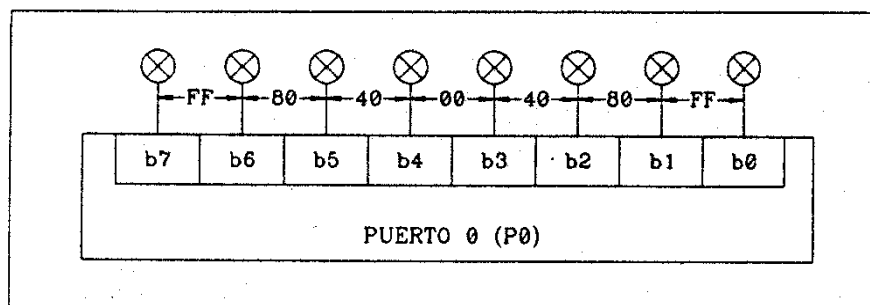


Figura 8.10

- **Esquema del circuito (Figura 8.11)**

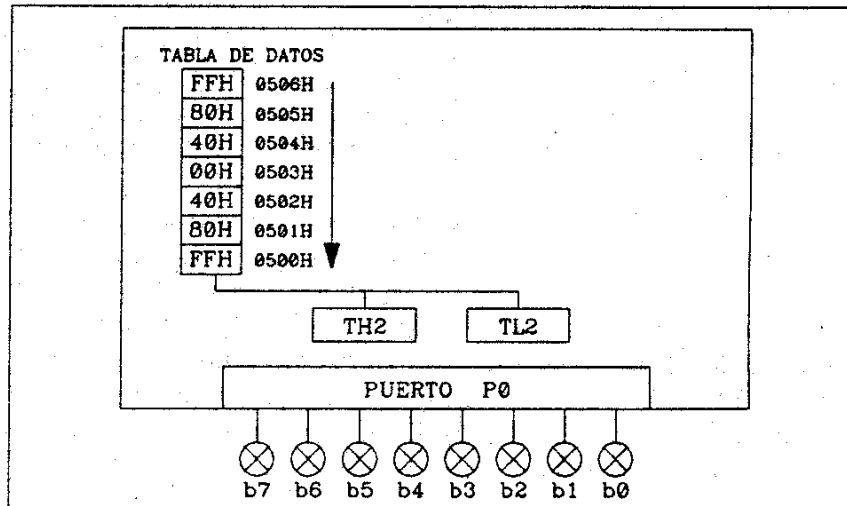


Figura 8.11

- **Programá fuente**

```

; INTETIM2.ASM.
;
; Igual que DESPLA0.ASM, pero la velocidad de desplazamiento
; es función de los valores registrados en una TABLA.
;
TABLA EQU 0500H ; Comienzo de la TABLA.
AUX EQU 40H ; Variable que indica el sentido de ROTACION.
ROTA EQU 41H ; Variable dato de salida por P0.
TIMER2 EQU 0300H ; Comienzo rutina Timer 2.
;
;-----
ORG 0500H ; Comienzo de los datos de la TABLA.
.BYTE FFH, 80H, 40H, 00H, 40H, 80H, FFH
ORG 002BH ; Vector de rutina interrupción (Timer 2).
JMP TIMER2
;
;-----
ORG 020H ; Programa de inicialización y principal.
MOV DPTR, #TABLA ; Se carga inicio de TABLA en puntero DPTR.
MOV AUX, #01H ; Se inicia movimiento rotación a izda.
MOV ROTA, #00H ; Se inicia valor salida leds por puerto PO.
SETB C ; C ← 1. Este es el bit que se desplaza.
MOV IE, #A0H ; Se habilita interrupción Timer 2.
MOV IP, #40H ; Máxima prioridad Int. Time,2 (no es necesario).
MOV TL2, #00H ; Se carga registro contador Timer 2.
MOV TH2, #00H

```

```

SAL0:    MOV    T2CON, #04H ; Se arranca Time,2.
         NOP                      ; Simula el programa principal.
         JMP    SAL0
         ;
         ;-----
         ORG    0300H            ; Comienzo de la rutina de interrupción.
         ANL    T2CON, #BFH ; Se borra la bandera de interrupción.
         PUSH  A                ; Guardar registros programa principal.
         MOV    A, AUX          ; Para saber el sentido de rotación que ...
         JZ     SAL1           ; ...corresponde. Si AUX = 01H a la izda ...
         MOV    A, ROTA        ;...si AUX=00H a la dcha.
         RLC    A
         MOV    ROTA, A
         JMP    SAL2
SALI:    MOV    A, ROTA
         RRC    A
         MOV    ROTA, A
SAL2:    MOV    P0, ROTA        ; El contenido de ROTA se saca por el P0.
         MOV    A, #00H        ; Se inicializa ACC.
         MOVC  A, @A+DPTR      ; Se obtiene dato de temporización de TABLA.
         MOV    TL2, A         ; Se carga dato TABLA en Time,2.
         MOV    TH2, A
         MOV    T2CON, #04     ; Se arranca Timer2.
         INC    DPTR           ; Se incrementa puntero de TABLA.
         MOV    A, AUX        ; Si el bit que se desplaza llegó al extremo ...
         JZ     SAL3         ; ...del recorrido se cambia el sentido de la ...
         MOV    A, ROTA       ; ...rotación. cambiando el valor de AUX.
         CJNE  A, #80H, SAL4
         MOV    AUX, #00H
         MOV    DPTR, #TABLA   ; Se inicializa puntero de TABLA.
         JMP    SAL4
SAL3:    MOV    A, ROTA
         CJNE  A, #01H, SAL4
         MOV    AUX, #01H
         MOV    DPTR, #TABLA
SAL4:    CLR    C
         POP   A
         RETI
         ;
         END

```