UNIDAD VI

MANEJO DE INTERRUPCIONES

VI.1. Tipos de interrupción

Las interrupciones son un mecanismo del microcontrolador que le permite responder a eventos asíncronos en el momento que ocurren, independientemente de lo que el microcontrolador este realizando. Esta es una parte muy importante del circuito porque proporciona una conexión entre un microcontrolador y el ambiente que lo rodea. Generalmente, cada interrupción cambia el flujo del programa, lo interrumpe y después de ejecutar una rutina de atención continúa en el mismo punto del programa.

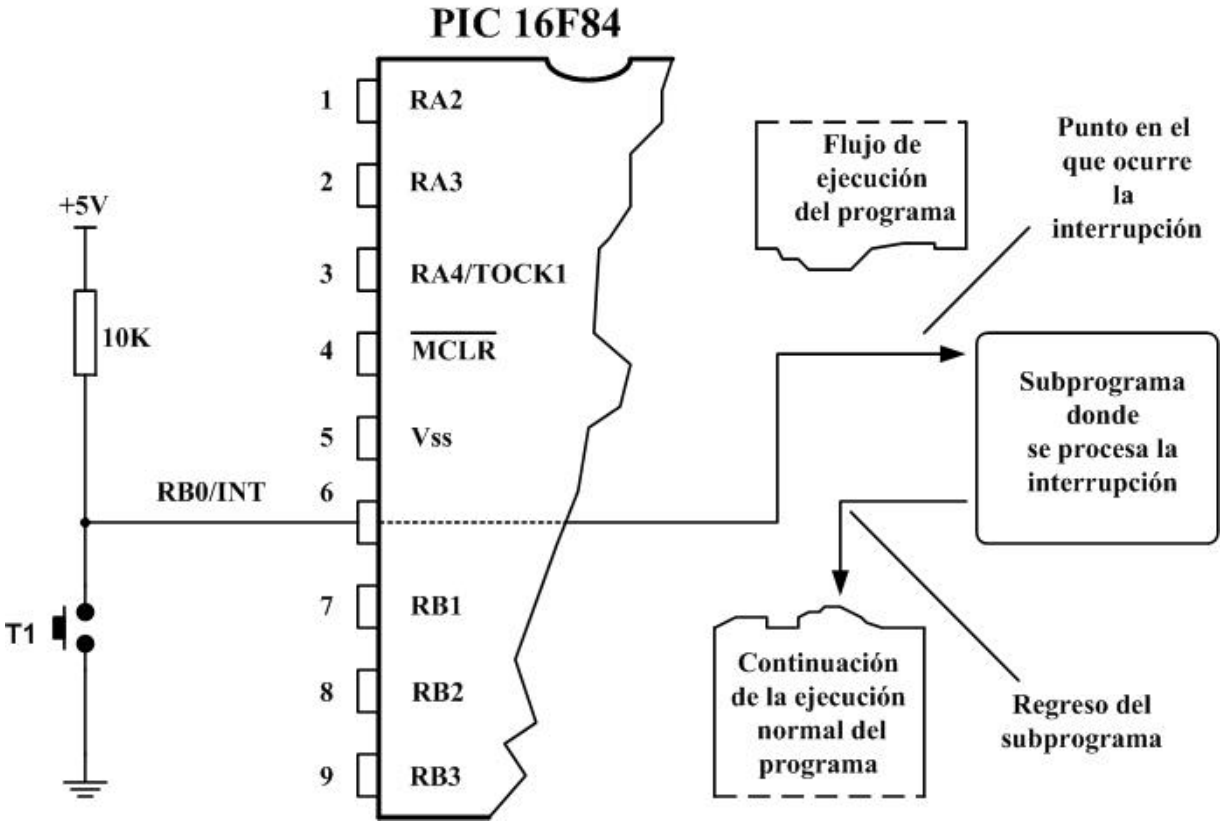

Figura VI.1. Relación entre una interrupción y el programa principal

VI.2. Modos de operación

La lógica de interrupciones del PIC16F84 tiene un registro de control llamado **INTCON** que se encuentra en la localidad de memoria RAM 0Bh. La función principal de este registro es habilitar o deshabilitar las interrupciones e indicar las solicitudes de interrupción recibidas por medio de cada uno de sus bits.

VI.3. Configuración de interrupciones

El registro INTCON

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GIE	**EEIE**	**TOIE**	**INTE**	**RBIE**	**TOIF**	**INTF**	**RBIF**

R = Bit que se puede leer. W = Bit que se puede escribir.
U = Bit no usado, se lee un 0.
-n = Valor inicial al encender la fuente de alimentación.

Bit 0-RBIF (RB Port Change Interrupt Flag Bit-Bit de la bandera de la interrupción de un cambio en el puerto B)
Este bit informa si ha ocurrido un cambio en alguna de las líneas 4, 5, 6 y 7 del puerto B.
1 = al menos una línea ha cambiado su estado.
0 = no ha ocurrido cambio en alguna de las líneas.

Bit 1-INTF (INT External Interrupt Flag Bit-Bit de la bandera de la interrupción externa INT)
1 = ocurrió una interrupción de este tipo.
0 = no ocurrió una interrupción de este tipo.
Si un flanco de subida o bajada configurado se detectó en la línea RB0/INT, (el cual es seleccionado con el bit INTEDG del registro OPTION), se activa el bit INTF. Este bit debe limpiarse en la rutina de atención para detectar la siguiente interrupción.

Bit 2-TOIF (TMR0 Overflow Interrupt Flag Bit-Bit de la bandera de la interrupción de sobreflujo del TMR0)
1 = el contador cambio su cuenta de FFh a 00h.
0 = no ocurrió sobreflujo.
Este bit debe limpiarse en el programa para que pueda detectarse una siguiente interrupción de este tipo.

Bit 3-RBIE (RB Port Change Interrupt Enable Bit-Bit de habilitación de la interrupción de un cambio en el puerto B)

Habilita la interrupción de cambio de estado de las líneas 4, 5, 6 y 7 del puerto B.

1 = habilita que la interrupción ocurra con un cambio de estado.

0 = las interrupciones se deshabilitan cuando ocurra un cambio de estado.

Si RBIE y RBIF se habilitan simultáneamente, puede ocurrir una interrupción de este tipo.

Bit 4-INTE (INT External Interrupt Enable Bit-Bit de habilitación de la interrupción externa INT)

Este bit habilita la interrupción externa de la línea RB0/INT.

1 = interrupción externa habilitada.

0 = interrupción externa deshabilitada.

Bit 5-TOIE (TMR0 Overflow Interrupt Enable Bit-Bit de habilitación de la interrupción de sobreflujo del temporizador TMR0)

Este bit habilita la interrupción cuando ocurra un sobreflujo en el contador del temporizador TMR0.

1 = interrupción habilitada.

0 = interrupción deshabilitada.

Si TOIE y TOIF se habilitan simultáneamente, puede ocurrir una interrupción de este tipo.

Bit 6-EEIE (EEPROM Write Complete Interrupt Enable Bit-Bit de habilitación de la interrupción al completarse una escritura en la EEPROM)

Este bit habilita la interrupción que ocurre al terminar una rutina de escritura a la EEPROM.

1 = interrupción habilitada.

0 = interrupción deshabilitada.

Si EEIE y EEIF (que se encuentra en el registro EECON1) se habilitan simultáneamente, puede ocurrir una interrupción de este tipo.

Bit 7-GIE (Global Interrupt Enable Bit-Bit de habilitación global de interrupciones)
Este bit habilita o deshabilita todas las interrupciones.
1 = se habilitan todas las interrupciones.
0 = se deshabilitan todas las interrupciones.

Así pues, el PIC16F84 tiene cuatro fuentes de interrupción:

1) Al terminar una escritura de datos a la EEPROM.
2) La interrupción causada por el sobreflujo de TMR0.
3) La interrupción al cambiar de estado cualquiera de las líneas RB4, RB5, RB6 y RB7 del puerto B.
4) La interrupción externa proveniente de la línea RB0/INT del microcontrolador.

Generalmente hablando, cada interrupción tiene asociados dos bits. Uno habilita la interrupción y el otro indica cuándo la interrupción ha ocurrido. Existe un bit común llamado GIE el cual puede usarse para habilitar o deshabilitar todas las interrupciones simultáneamente. Este bit es muy útil al escribir un programa para deshabilitar todas las interrupciones por un periodo de tiempo tal que no pueda ser interrumpida alguna parte importante o crítica del programa.

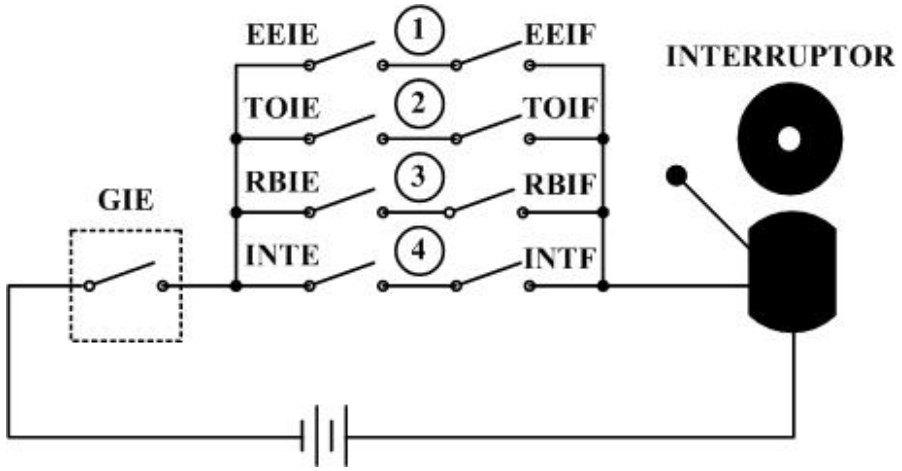

Figura VI.2. Lógica de interrupciones del PIC16F84

Cuando se atiende una interrupción se limpia automáticamente el bit GIE para deshabilitar interrupciones adicionales, la dirección de retorno del programa interrumpido se salva en el stack y se carga el program counter con la dirección 0004h y solo después de esto, la atención a la interrupción comienza. Después de procesarse la interrupción, el bit que la causó debe limpiarse o la rutina de la interrupción será procesada nuevamente al regresar al programa interrumpido.

Salvar el contenido de los registros importantes

Solamente se salva en el stack el valor de regreso del program counter durante una interrupción (la dirección de regreso del program counter es la dirección de la instrucción que debía ejecutarse pero no lo fue debido a que ocurrió la interrupción). Salvar el valor del program counter a menudo es suficiente, sin embargo, algunos registros que se están usando en el programa interrumpido se pueden usar en la rutina de la interrupción. Si esos registros no son salvados, el programa interrumpido podrá encontrar valores diferentes en esos registros al regresar de la interrupción, lo cual podrá causar errores.

En algunos microcontroladores el proceso de salvar el contenido de registros importantes antes de ir a una rutina de una interrupción se hace con instrucciones llamadas PUSH, y el proceso de restaurar su valor se hace con instrucciones llamadas POP. El PIC16F84 **NO** tiene instrucciones de este tipo y deben ser programadas en caso necesario.

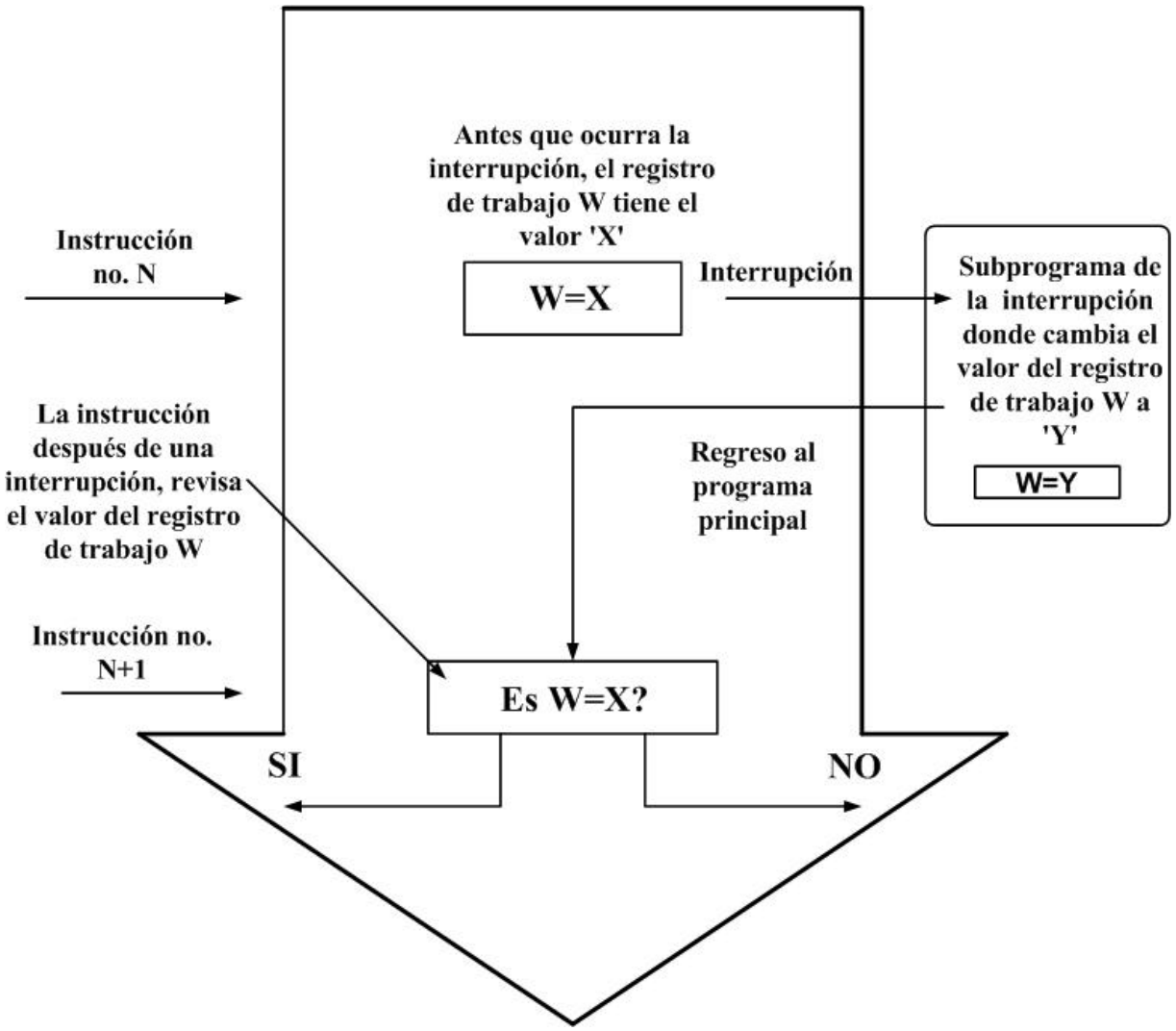

Figura VI.3. Problema que ocurre cuando no se salva el contexto antes de una rutina de interrupción

Debido a la frecuencia de uso y simplicidad de estos procesos, estas partes del programa pueden hacerse como macros. En el ejemplo siguiente, el contenido de los registros W y STATUS se salvan en las variables (o localidades de memoria) W_TEMP y STATUS_TEMP antes de la rutina de la interrupción. Para intercambiar los datos entre los registros STATUS y W se puede usar la instrucción SWAPF en lugar de la instrucción MOVF, ya que la primera no afecta el valor de los bits del registro STATUS.

La instrucción SWAPF STATUS,0 carga los bits 0-3 del registro STATUS en los bits o posiciones 4-7 del registro W y los bits 4-7 del registro STATUS en los bits 0-3 del registro W.

El ejemplo sigue los pasos indicados a continuación:

1) Almacena el registro W independientemente del banco actual.
2) Almacena el registro STATUS independientemente del banco actual.
3) Salva los registros deseados del banco 0.
4) Salva los registros deseados del banco 1.
5) Ejecuta la rutina de la interrupción (ISR).
6) Restaura los registros salvados del banco 0.
7) Restaura los registros salvados del banco 1.
8) Restaura el valor del registro STATUS.
9) Restaura el valor del registro W.

Si existen más variables o registros que necesitan salvarse, debe hacerse después de salvar el registro STATUS (pasos 3 y 4), y restaurarse antes del registro STATUS (pasos 6 y 7).

```
PUSH
        MOVWF       W_TEMP          ;Salva el registro W.
        SWAPF       STATUS,0        ;W ✍ STATUS.
        MOVWF       STATUS_TEMP     ;STATUS_TEMP ✍ W.
;
;***** Salva los registros del banco 0 *****
;
        BCF         STATUS,RP0      ;Cambia al banco 0.
        MOVF        TMR0,W          ;Salva el registro TMR0.
        MOVWF       TMR0_TEMP
        MOVF        PORTA,W         ;Salva el registro PORTA.
        MOVWF       PORTA_TEMP
;
;***** Salva los registros del banco 1 *****
:
        BSF         STATUS,RP0      ;Cambia al banco 1.
        MOVF        OPTION_REG,W    ;Salva el registro OPTION.
        MOVWF       OPTION_TEMP
        MOVF        TRISA,W         ;Salva el registro TRISA.
        MOVWF       TRISA_TEMP
                                    ;Termina el PUSH.
ISR_CODE
        ;
        ;(Programa de la interrupción).
        ;
POP
;
;***** Restaura los registros del banco 0 *****
;
        BCF         STATUS,RP0      ;Cambia al banco 0.
        MOVF        TMR0_TEMP,W     ;Restaura reg. TMR0.
        MOVWF       TMR0
        MOVF        PORTA_TEMP,W    ;Restaura reg. PORTA.
        MOVWF       PORTA
;
```

```
;***** Restaura los registros del banco 1 *****
;
        BSF         STATUS,RP0      ;Cambia al banco 1.
        MOVF        OPTION_TEMP,W   ;Restaura reg. OPTION.
        MOVWF       OPTION_REG
        MOVF        TRISA_TEMP,W    ;Restaura reg. TRISA.
        MOVWF       TRISA
;
        SWAPF       STATUS_TEMP,0   ;W ✍ STATUS_TEMP.
        MOVWF       STATUS          ;STATUS ✍ W.
        MOVF        W_TEMP,W        ;Restaura contenido de W.
        RETFIE                      ;Termina el POP.
```

Este mismo ejemplo se puede realizar usando macros, obteniendo con esto un programa más legible. Los macros ya definidos pueden usarse para escribir nuevos macros. Los macros llamados BANK1 y BANK0, los cuales ya se explicaron en el tema de “Organización de memoria”, se usan en los macros PUSH y POP.

```
PUSH    macro
        MOVWF       W_TEMP          ;Salva el registro W.
        SWAPF       STATUS,W        ;W ✍ STATUS.
        MOVWF       STATUS_TEMP     ;STATUS_TEMP ✍ W.
;
;***** Salva los registros del banco 0 *****
;
        BANK0                       ;Cambia al banco 0.
        MOVF        TMR0,W          ;Salva el registro TMR0.
        MOVWF       TMR0_TEMP
        MOVF        PORTA,W         ;Salva el registro PORTA.
        MOVWF       PORTA_TEMP
;
;***** Salva los registros del banco 1 *****
:
        BANK1                       ;Cambia al banco 1.
        MOVF        OPTION_REG,W    ;Salva el registro OPTION.
        MOVWF       OPTION_TEMP
        MOVF        TRISA,W         ;Salva el registro TRISA.
        MOVWF       TRISA_TEMP
        endm                        ;Termina el macro PUSH.
```

```
POP  macro
;
;***** Restaura los registros del banco 0 *****
;
        BANK0                           ;Cambia al banco 0.
        MOVF        TMR0_TEMP,W         ;Restaura reg. TMR0.
        MOVWF       TMR0
        MOVF        PORTA_TEMP,W        ;Restaura reg. PORTA.
        MOVWF       PORTA
;
;***** Restaura los registros del banco 1 *****
;
        BANK1                           ;Cambia al banco 1.
        MOVF        OPTION_TEMP,W       ;Restaura reg. OPTION.
        MOVWF       OPTION_REG
        MOVF        TRISA_TEMP,W        ;Restaura reg. TRISA.
        MOVWF       TRISA
;
        SWAPF       STATUS_TEMP,W       ;W ✍ STATUS_TEMP.
        MOVWF       STATUS              ;STATUS ✍ W.
        MOVF        W_TEMP,W            ;Restaura contenido de W.
        endm                            ;Termina el macro POP.
```

Interrupción externa de la línea RB0/INT del microcontrolador

La interrupción externa de la línea RB0/INT es disparada por la transición de subida (si el bit INTEDG=1 en el registro OPTION<6>), o por la transición de bajada (si INTEDG=0). Cuando la señal correcta aparece en la línea INT se activa el bit INTF del registro INTCON. El bit INTF (INTCON<1>) debe limpiarse en la rutina de la interrupción para que no vuelva a ocurrir otra interrupción al regresar al programa principal. Esta es una parte importante que el programador no debe olvidar o el programa constantemente estará pasando a la rutina de la interrupción. La interrupción puede apagarse limpiando el bit INTE (INTCON<4>).

Interrupción de sobreflujo del contador de TMR0

Un sobreflujo del contador de TMR0 (cuando la cuenta pasa de FFh a 00h) activará el bit TOIF (INTCON<2>). Una de las aplicaciones de esta interrupción es para medir tiempo. Si se conoce cuanto tiempo necesita el contador para completar un ciclo de 00h a FFh, entonces el número de interrupciones multiplicado por esa cantidad de tiempo dará como resultado el tiempo total transcurrido. En la rutina de atención de la interrupción alguna variable podrá ser incrementada en la memoria RAM, el valor de esa variable multiplicado por la cantidad de tiempo que el contador necesita para contar un ciclo completo dará como resultado el tiempo transcurrido. La interrupción puede apagarse o encenderse limpiando o activando el bit TOIE (INTCON<5>).

Interrupción de cambio de estado de las líneas 4, 5, 6 y 7 del puerto B

El cambio de la señal de entrada de PORTB <7:4> activa el bit RBIF (INTCON<0>). Las cuatro línea RB7, RB6, RB5 y RB4 del puerto B pueden disparar una interrupción la cual ocurre cuando el estado de ellas cambia de uno a cero lógico o viceversa. Para que las líneas sean sensitivas a este cambio, deben ser definidas como entradas. Si alguna de ellas es definida como salida, no se genera interrupción al ocurrir un cambio de estado en esa línea. Si son definidas como entradas, su estado actual se compara con el valor anterior que fue almacenado en la ultima lectura del puerto B. La interrupción puede apagarse o encenderse limpiando o activando el bit RBIE del registro INTCON.

Interrupción al terminar la subrutina de escritura a la EEPROM

Debido a que escribir a una localidad de la EEPROM toma aproximadamente 10 ms (el cual es un tiempo grande para el microcontrolador), muchas veces no tiene caso que el microcontrolador espere ocioso que termine la escritura. El PIC16F84 cuenta con un mecanismo de interrupción que le permite continuar ejecutando el programa principal mientras se realiza una escritura a la EEPROM en segundo plano. Al terminar la escritura, una interrupción informa al microcontrolador dicho evento. El bit EEIF, por medio del cual se informa el final de la escritura, se encuentra en el registro EECON1. La ocurrencia de la interrupción se puede deshabilitar limpiando el bit EEIE del registro INTCON.

Inicialización de una interrupción

Para usar el mecanismo de interrupciones del microcontrolador es necesario realizar algunas tareas preparatorias. Esas tareas comúnmente se les conoce como inicialización. La inicialización es indicar a cuales interrupciones responderá el microcontrolador y cuales ignorará. Si no se activa el bit que permite una cierta interrupción, el programa no ejecutará la rutina asociada a esa interrupción, pudiéndose tener así el control en la ocurrencia de interrupciones.

```
clrf    INTCON      ;Deshabilita todas las interrupciones.
movlw   B´00010000´ ;Solo habilita interrupción externa.
movwf   INTCON
bsf     INTCON,GIE  ;Habilita ocurrencia de interrupciones.
```

El ejemplo anterior muestra la inicialización de la interrupción externa de la línea RB0.

El ejemplo siguiente muestra una forma típica para el manejo de interrupciones. El PIC16F84 tiene solo una localidad de memoria donde se almacena la dirección de una rutina de una interrupción. Esto implica que es necesario primero detectar cuál interrupción se ha generado (si más de una fuente de interrupción está activa) y poder ejecutar la parte del programa que se refiere a esa interrupción.

```
org     ISR_ADDR          ;ISR_ADDR es la direcc. de la rutina
                          ;de interrupción.
btfsc   INTCON,GIE        ;Esta apagado el bit GIE?
goto    ISR_ADDR          ;No, regresa al inicio.
PUSH                      ;Salva contenido de regs. importantes.
btfsc   INTCON,RBIF       ;Cambian líneas 4,5,6 y 7 de puerto B?
goto    ISR_PORTB         ;Si, salta a esa sección.
btfsc   INTCON,INTF       ;Ocurrió interrupción externa?
goto    ISR_RB0           ;Si, salta a esa parte.
btfsc   INTCON,TOIF       ;Overflow del timer TMR0?
goto    ISR_TMR0          ;Si, salta a esa sección.
BANK1                     ;EECON1 esta en el banco 1.
btfsc   EECON1,EEIF       ;Se completo escritura a EEPROM?
goto    ISR_EEPROM        ;Si, salta a esa sección.
BANK0                     ;Banco 0.
;
ISR_PORTB
 ;
                          ;Sección de código procesado
                          ;por la interrupción.
 ;
        goto   END_ISR    ;Salta al final de la interrupción.
ISR_RB0
 ;
                          ;Sección de código procesado
                          ;por la interrupción.
 ;
```

```
        goto  END_ISR       ;Salta al final de la interrupción.
ISR_TMR0
 ;
                            ;Sección de código procesado
                            ;por la interrupción.
 ;
        goto  END_ISR       ;Salta al final de la interrupción.
ISR_EEPROM
 ;
                            ;Sección de código procesado
                            ;por la interrupción.
 ;
        goto  END_ISR       ;Salta al final de la interrupción.
END_ISR
        POP                 ;Restaura el contenido de los
                            ;registros importantes.
        RETFIE              ;Regresa y activa bit GIE.
```

El regreso de una rutina de atención a una interrupción se puede realizar con las instrucciones RETURN, RETLW y RETFIE. Se recomienda usar la instrucción RETFIE, ya que esta instrucción es la única que automáticamente activa el bit GIE el cual permite que ocurran nuevas interrupciones.