Universidad Nacional de Ingeniería
Facultad de Ciencias

Arquitectura de computadores

Interrupciones en el 8051

Prof.: Lic. César Martín Cruz S.
ce.cruz@gmail.com

2015 - I

¿Qué es una Interrupción?

Una interrupción es la aparición de un evento que causa una suspensión temporal de un programa, mientras que el evento es atendido por una sección de código conocido como rutina de servicio de interrupción. El servicio de interrupción puede ser causada por la pulsación de una tecla en un teclado, o la llegada de un mensaje de texto en un teléfono móvil.

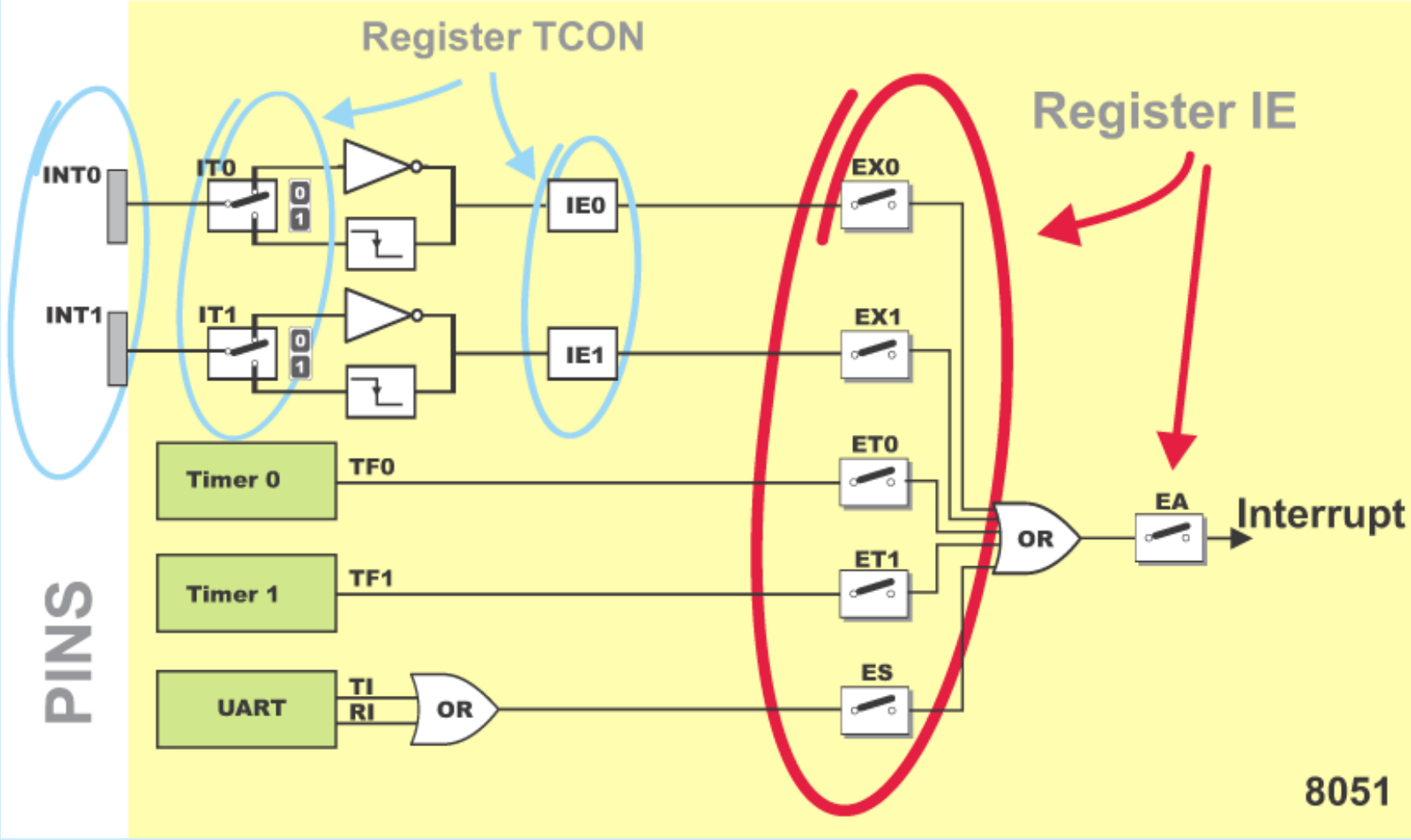
La ***rutina del servicio de interrupción*** **(ISR)** que se encarga de la interrupción es muy parecida a una subrutina, la ejecución de código salta al ISR, el ISR se ejecuta, la ejecución se reanuda de nuevo en la instrucción que sigue a la que había sido ejecutado justo antes de saltar al ISR.

Interrupciones

Hay cinco fuentes de interrupción para el 8051, lo que significa que puede reconocer 5 diferentes eventos que pueden interrumpir la ejecución del programa regular. Cada interrupción puede ser activado o desactivado configurando los bits del registro **IE**. Del mismo modo, el sistema de interrupciones del conjunto puede ser desactivado borrando el bit **EA** del mismo registro. Ver la figura de abajo.

En cuanto a las interrupciones externas **INT0** y **INT1**. Si los bits **IT0** e **IT1** del registro **TCON** se fijan a uno lógico, una interrupción se generará en la transición de alto a bajo, es decir, en el flanco descendente (sólo en ese momento). Si estos bits se borran (poner a 0 lógico), una interrupción se ejecuta continuamente en cuanto estas entradas se mantengan bajo.

Interrupciones



Interrupciones

Dos interrupciones internas, uno ocurrido en el overflow del timer 0 y otro en el overflow del timer 1.
El puerto serie genera una interrupción cuando un byte ha sido transmitido o cuando un byte es recibido.

Las banderas de interrupción y las posiciones en los registros son detalladas en la tabla siguiente:

Interrupción	Bandera	Posición
Externo 0	IE0	TCON.1
Externo 1	IE1	TCON.3
Timer 0	TF0	TCON.5
Timer 1	TF1	TCON.7
Puerto Serie Recepción	RI	SCON.0
Puerto Serie Transmisión	TI	SCON.1

Interrupciones

Cuando ocurre una interrupción pasa lo siguiente:

1. El CPU completa la ejecución de la instrucción corriente.
2. El registro PC (Contador de Programa) es guardado en la pila.
3. La dirección del ISR para esa interrupción particular es cargado en el registro PC.

En cuanto la instrucción actual haya completado su operación, el valor en el contador de programa en el paso 2 de arriba es la dirección de la siguiente instrucción que habría sido ejecutado si no hubiera ocurrido la interrupción. Por lo tanto, en este sentido, saltando a un ISR es el mismo que saltar a una subrutina.

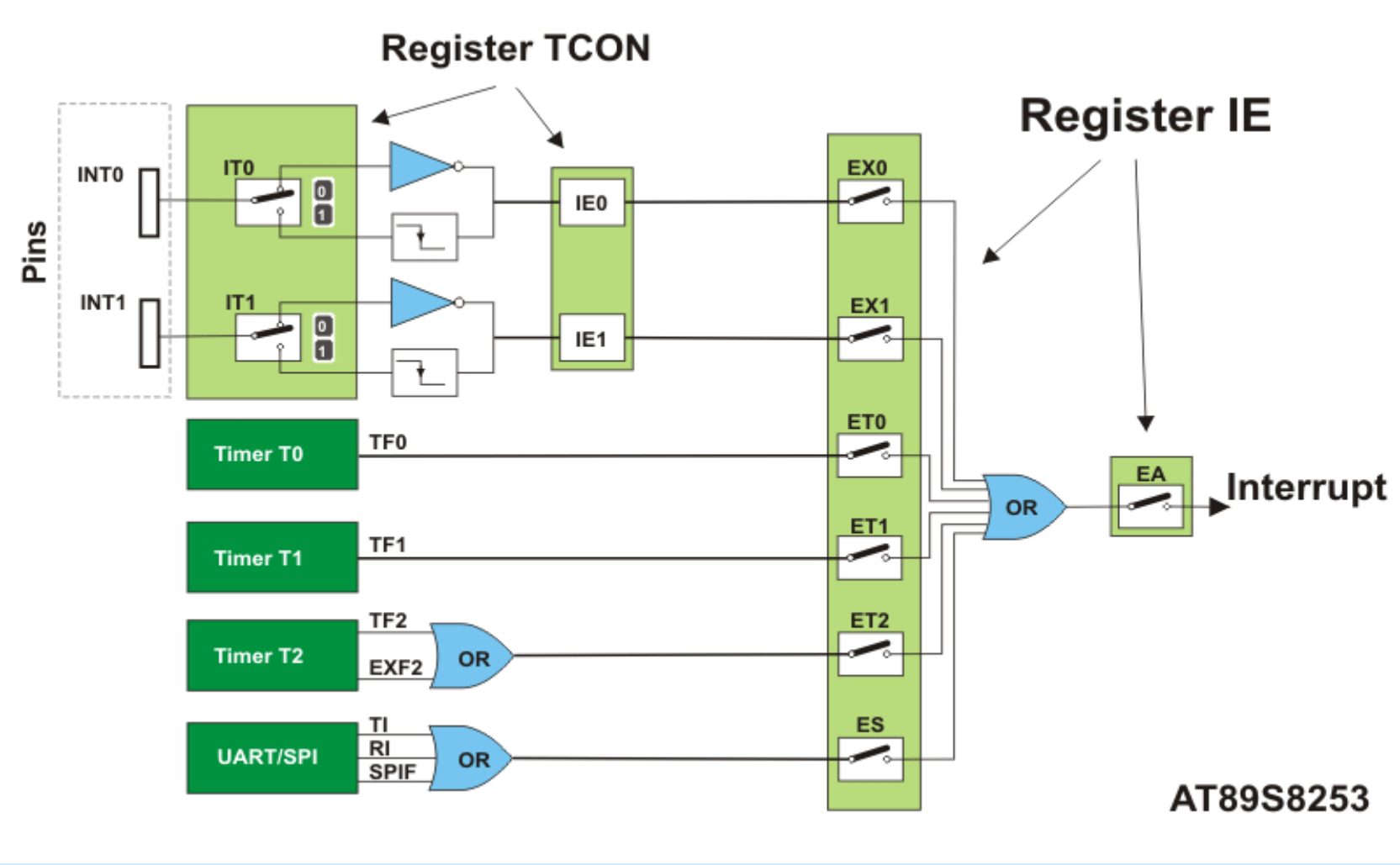
Vectores de Interrupción

Cuando una interrupción ocurre, la dirección de la rutina de servicio de la interrupción es cargada en el registro PC. Esta dirección es conocida como el *vector de interrupción*. La tabla debajo lista los vectores de interrupción para el 8051.

Interrupción	Bandera	Vector
Reset del sistema	RST	0000H
Interrupción Externa 0	IE0	0003H
Timer 0	TF0	000BH
Interrupción Externa 1	IE1	0013H
Timer 1	TF1	001BH
Puerto Serie	RI o TI	0023H

Un reset del sistema es un tipo especial de interrupción. Se interrumpe el programa que se está ejecutando y se carga el PC con el vector de dirección 0000H. Esta es la dirección de inicio que el microcontrolador ejecuta al momento que se energiza el sistema.

Interrupciones para AT89S8253



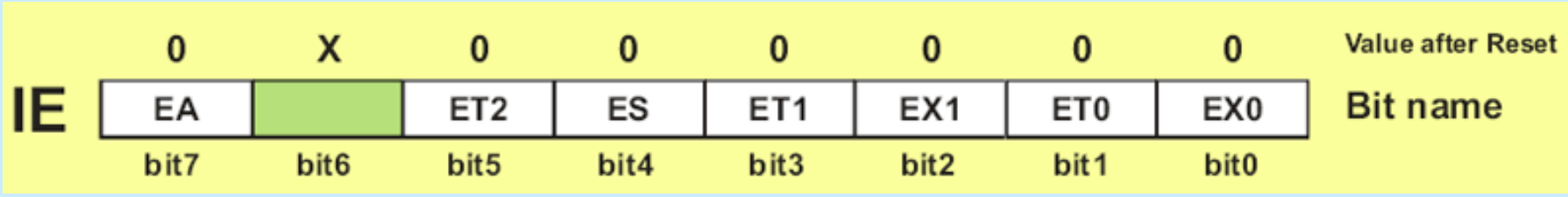
Interrupciones

El **AT89S8253** tiene un total de nueve fuentes de interrupción, lo que significa que es capaz de reconocer hasta 9 diferentes eventos que pueden interrumpir la ejecución del programa regular. Cada una de estas interrupciones se puede activar o desactivar individualmente mediante la fijación de los bits del registro **IE**.

Desde que este microcontrolador contiene el temporizador **T2** y la interfaz **SPI** (que no están contenidos en el "8051 Standard"), que pueden generar una interrupción, es necesario hacer algunos cambios en los registros de control del sistema de interrupción. Además, hay un nuevo vector de interrupción (dirección **2B**), es decir, la dirección de memoria de programa desde la que el programa prosigue con la ejecución cuando el temporizador **T2** genera una interrupción. Todos estos cambios se realizan en los bits no utilizados previamente.

Registro habilitador de interrupciones IE(Interruption Enable)

Las interrupciones son controladas mediante la escritura tanto en el registro **IE** (Interruption Enable) como en el registro **IP** (Interruption Priority):
La desactivación general de las interrupciones es efectuada mediante la escritura de un 0 lógico, en la bandera **EA**.
Con la bandera **EA = 1**, el AT89C52 está en condiciones de aceptar interrupciones, aunque la verdadera aceptación es realizada, cuando se escribe un 1 lógico, en la bandera de la interrupción correspondiente del registro de interrupciones, **IE**.



Registro IE

SÍMBOLO	BIT	FUNCIÓN
EA	IE.7	Desactiva todas las INTERRUPCIONES EA=0.
-	IE.6	No es usado.
ET2	IE.5	Activa la interrupción causada por el timer2 (ET2=1).
ES	IE.4	Activa la interrupción causada por el puerto serie.
ET1	IE.3	Activa la int. de sobreflujo causada por el timer1.
EX1	IE.2	Activa la int. causada externamente en INT 1.
ET0	IE.1	Activa la int. se sobreflujo causada por el timer 0.
EX0	IE.0	Activa la int. causada externamente en INT 0.

Por ejemplo, para habilitar la interrupción por sobreflujo para el timer 0 y el timer 1 escribiríamos el siguiente código:

Timer 0	***Timer 1***
setb ET0	**setb ET1**
setb EA	**setb EA**

Ejemplo 1:

Generar un tren de pulsos de 10khz sobre el pin P1.0 usando interrupciones del timer 0.

Sol.

Una señal de 10 khz tiene un periodo de 100 microsegundos. Por tanto, necesitamos que una interrupción ocurra cada 50 microsegundos. Para alcanzar esto, el timer 0 tendrá un desbordamiento cada 50 microsegundos y su interrupción será habilitada.

Con una frecuencia de reloj de 12Mhz, el timer contará a un paso de 1 microsegundo, por tanto necesitará contar desde 206(256-50) para desbordar. Esto puede ser hecho con el modo 2.

```
org 0000h   ;vector de reset
sjmp main
org 000Bh  ;vector interrupción del timer 0
cpl P1.0
```

```
 reti

        org 0030h           ;programa principal
main:
        mov TMOD,#2     ;Timer 0  en modo 2
        mov TH0,#206     ;carga 206 en TH0
        setb TR0            ;inicia el timer 0
        setb ET0            ;habilita interrupción del timer 0
        setb EA             ;habilita interrupción global
        sjmp $              ;se queda aquí dando vueltas esperando la interrupción
```

Cuando se desborda el timer 0 ocurre lo siguiente:

- El PC es salvado en la pila.
- El PC se carga con la dirección **000Bh** (*vector interrupción* para el timer 0).
- La ejecución empieza en **ISR**, en este caso simplemente invierte **P1.0**.
- **RETI** recupera el PC de la pila, la ejecución empieza de nuevo en el programa principal.

Prioridad de las Interrupciones

No es posible saber cuando llegará una petición de interrupción. Si varias interrupciones están habilitadas, puede suceder que, si bien uno de ellos está en marcha, otra se solicita. Con el fin de que el microcontrolador sepa si va continuar la operación o cumplir con una nueva petición de interrupción, hay una lista de prioridades que le instruyen qué hacer.

La lista de prioridades ofrece 3 niveles de prioridad de interrupción:

1. Reset! El maestro Absoluto. Cuando llega una petición de resetear, todo se detiene y se reinicia el microcontrolador.
2. Prioridad de interrupción alta se puede desactivar por Reset solamente.
3. Prioridad de interrupción baja se puede desactivar tanto por Reset y por prioridad de interrupción alta.

El Registro **IP** (Registro de prioridad de interrupción) especifica cual de las fuentes de interrupción tienen una prioridad mayor y cuales tienen una prioridad menor. La Prioridad de interrupción normalmente se especifica al comienzo del programa. De acuerdo con esto, hay varias posibilidades:

• Si una interrupción de mayor prioridad llega mientras una interrupción está en curso, se detiene inmediatamente y la interrupción de prioridad más alta será ejecutado primero.

• Si hay dos solicitudes de interrupción, de diferentes niveles de prioridad y llegan al mismo tiempo, entonces la interrupción de mayor prioridad es atendida primero.

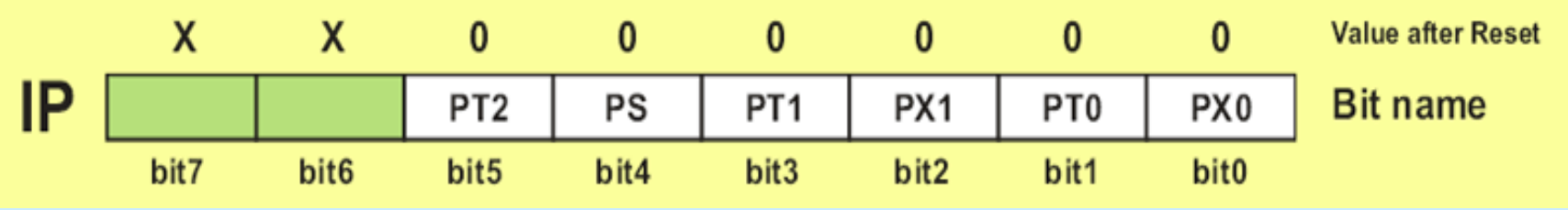
• Si varias solicitudes de interrupción, en el mismo nivel de prioridad, se producen uno tras otro, el que llegó más tarde tiene que esperar hasta que la rutina de interrupción que está en progreso termine.

• Si dos solicitudes de interrupción de igual prioridad llegan al mismo tiempo entonces la interrupción que va ser atendida se selecciona de acuerdo a la lista de prioridad siguiente:

1. Interrupción Externa INT0
2. Interrupción Timer 0
3. Interrupción Externa INT1
4. Interrupción Timer 1
5. Interrupción comunicación Serie
6. Interrupción Timer 2

Registro IP(Prioridad de Interrupciones)

Los bits del registro IP especifican el nivel de prioridad de cada interrupción (alta o baja prioridad).



Símbolo	Bit	Función
PT2	IP.5	Timer 2 PT2=1 mayor prioridad.
PS	IP.4	Define el nivel de prioridad de la int. puerto serie.
PT1	IP.3	Define el nivel de prioridad de la int. del timer 1.
PX1	IP.2	Define el nivel de prioridad de la int. 1 externa.
PT0	IP.1	Define el nivel de prioridad de la int. del timer 0.
PX0	IP.0	Define el nivel de prioridad de la int. 0 externa.

Ejemplo 2(Usando más de una interrupción):

Deseamos generar dos trenes de pulsos de diferentes frecuencias. Por ejemplo, un tren de pulsos de 20KHz sobre P3.4 y un tren de pulsos de 500Hz sobre P3.5. El programa debajo pudiera ser usado para generar estos trenes de pulsos.

```
org 0000h
sjmp main

org 000Bh  ;vector de interrupción del timer 0
```

```
cpl P3.4     ;genera un tren de pulsos de 20 khz
reti         ;retorna de la interrupción

org 001Bh ;vector de interrupción del timer 1
sjmp timer1ISR  ;salta al ISR del timer 1

org 0030h ;programa principal
main:
   mov TMOD,#12h  ;timer 0 en modo de 8 bits con autorecarga y timer 1 en modo
                  ;de 16 bits
   mov TH0,#231   ;valor de autorecarga
   setb TR0       ;arranca el timer 0
   setb TF1       ;se fuerza interrupción del timer 1
   setb ET0       ;se habilita la interrupción del timer 0
   setb ET1       ;se habilita la interrupción del timer 1
   setb EA        ;se habilita la interrupción global
   sjmp $         ;aquí está dando vueltas y espera por interrupciones
;Viene aquí cuando interrumpe el timer 1
timer1ISR:
   clr TR1        ;detiene el timer 1
```

```
mov TH1,#0FCh    ;inicializa el timer 1 con "FC18h"
mov TL1,#18h     ;(65536 – 1000=64536)(FC18h)
setb TR1         ;arranca el timer 1
cpl P3.5
reti             ;retorna de la interrupción
```

En la mayoría de sistemas, algunos eventos son más importantes que otros. Por ejemplo, revisemos nuestro programa de trenes de pulsos de 20KHz y 500Hz.
Cuando el timer 1 desborda el programa salta al ISR del timer 1. Luego, mientras está en el proceso de reinicializar el timer, el timer 0 desborda. Pero debido a que la interrupción del timer 1 está siendo manipulado, el timer 0 debe esperar. Cuando el timer 1 completa el ISR, entonces el ISR del timer 0 se ejecuta.

El problema con esto es el hecho que la frecuencia (20KHz) del tren de pulsos sobre P3.4 generada por el timer 0 es mucho más alta que la frecuencia de 500Hz generada por el timer 1. Por tanto, el retardo en invertir P3.4 puede notablemente afectar la frecuencia del tren de pulsos generado.

La solución a este problema es tratar el desborde del timer 0 como que es más importante (esto es darle una prioridad más alta) que el desborde del timer 1.

Usando prioridades en el ejemplo de trenes de pulsos

Se tiene el programa anterior modificado:

```
org 0000h
sjmp main

org 000Bh  ; vector de interrupción del timer 0
cpl P3.4   ; genera una señal de 20 khz
reti       ; retorna de la interrupción

org 001Bh      ;vector de interrupción del timer 1
sjmp timer1ISR ;salta al ISR del timer 1

org 0030h          ;programa principal
main:
   mov TMOD,#12h   ;timer 0 en modo de 8 bits con autorecarga y timer 1 en modo
                   ;de 16 bits
   mov TH0,#231    ;valor de autorecarga
   setb TR0        ;arranca el timer 0
```

```
    setb TF1          ;se fuerza interrupción del timer 1
    setb ET0          ;se habilita la interrupción del timer 0
    setb PT0          ;el timer 0 se fija a alta prioridad
    setb ET1          ;se habilita la interrupción del timer 1
    setb EA           ;se habilita la interrupción global
    sjmp $            ;aquí está dando vueltas pero espera por interrupciones
;Viene aquí cada vez que hay una interrupción del timer1
timer1ISR:
    clr TR1           ;detiene el timer 1
    mov TH1,#0FCh     ;inicializa el timer 1 con "FC18h"
    mov TL1,#18h      ;(65536 – 1000= 64536)(este número es FC18h en hexadecimal)
    setb TR1          ;arranca el timer 1
    cpl P3.5          ;se genera 500 hz
    reti              ;retorna de la interrupción
```