



EL SISTEMA DE INTERRUPCIONES





Resumen de contenidos

- 1 - Objetivos
- 2 - Introducción
- 3 - Estructura de las interrupciones
- 4 - Registros involucrados y fuentes de interrupción
- 5 - Estructura de los niveles de prioridad
- 6 - Gestión de las interrupciones
- 7 - Ejemplos de aplicación
- 8 - Ejercicios propuestos
- 9 - Bibliografía
- 10 - Apéndice





1 - Objetivos





1 - Objetivos

- Conocer el sistema de interrupciones de la familia MCS-51
- Diferenciar las fuentes de interrupción.
- Programar los registros involucrados para la gestión de interrupciones.
- Programar los diferentes niveles de prioridad de las interrupciones.
- Diseñar aplicaciones reales mediante la utilización de interrupciones.





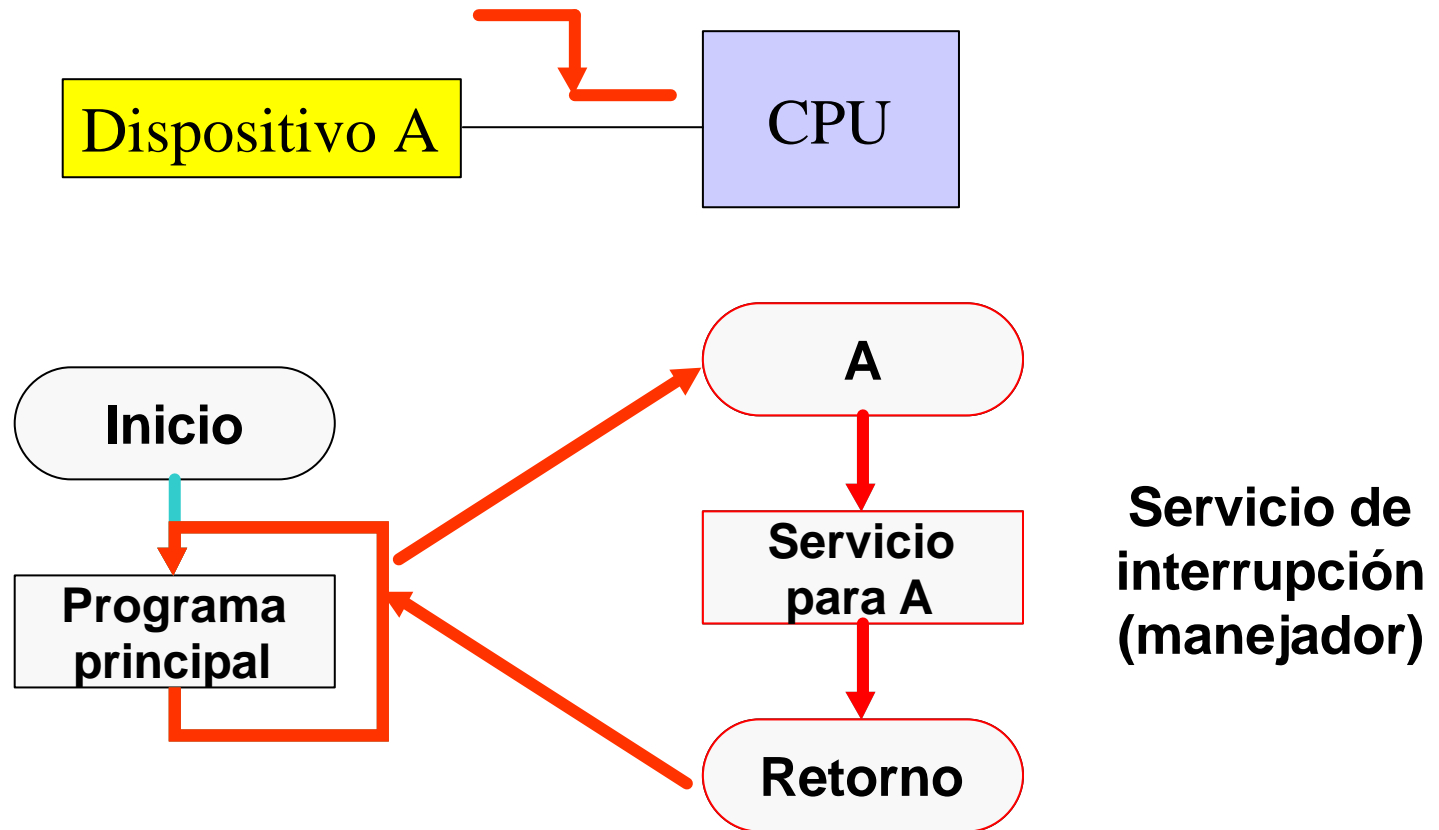
2 - Introducción





2 - Introducción

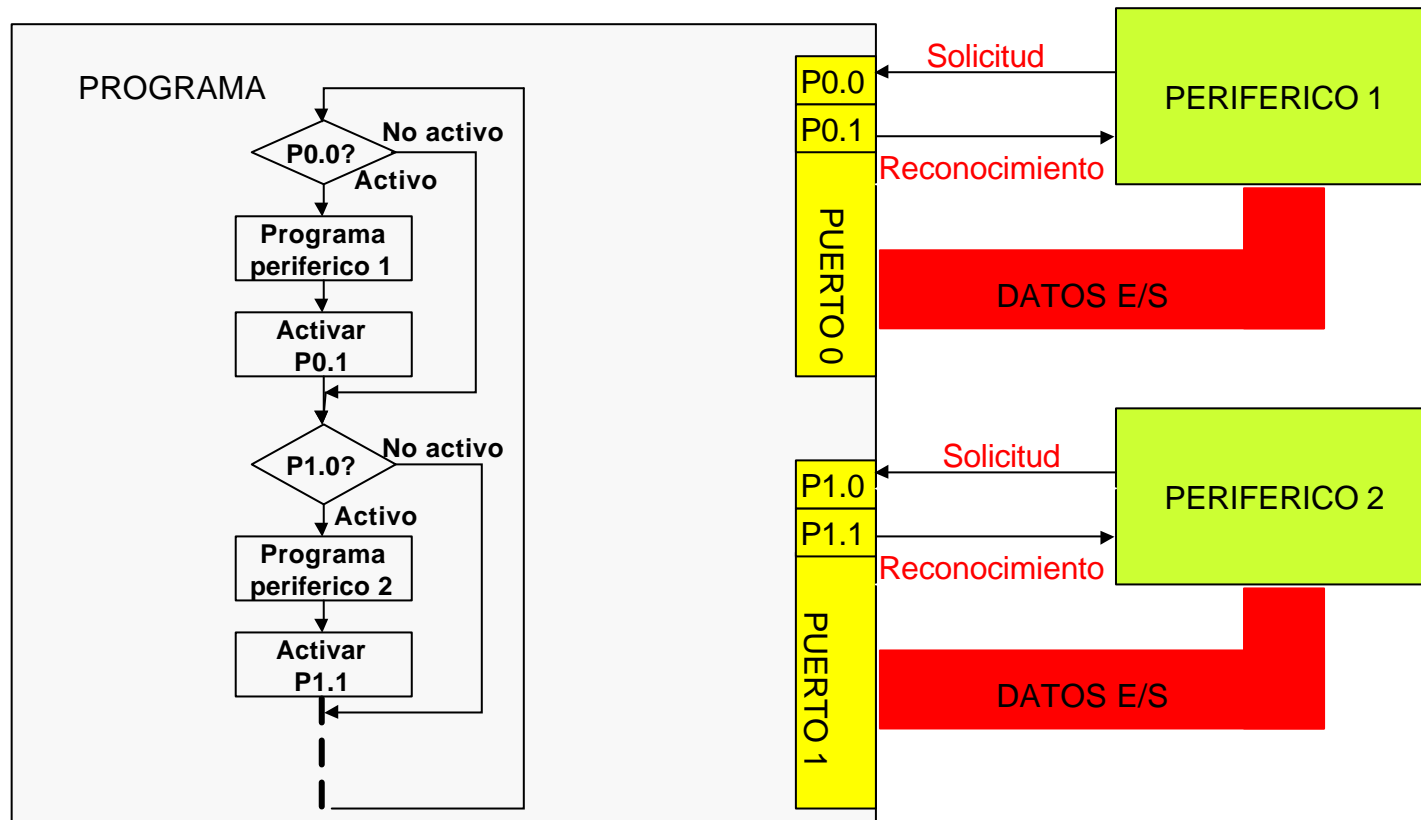
Concepto de INTERRUPCIÓN: *El programa en curso, tras un requerimiento, paraliza su ejecución y pasa a ejecutarse una rutina (manejador) de interrupción.*





2 - Introducción

- ATENCIÓN DE PERIFÉRICOS MEDIANTE CONSULTA DE ESTADO (**POLLING**)

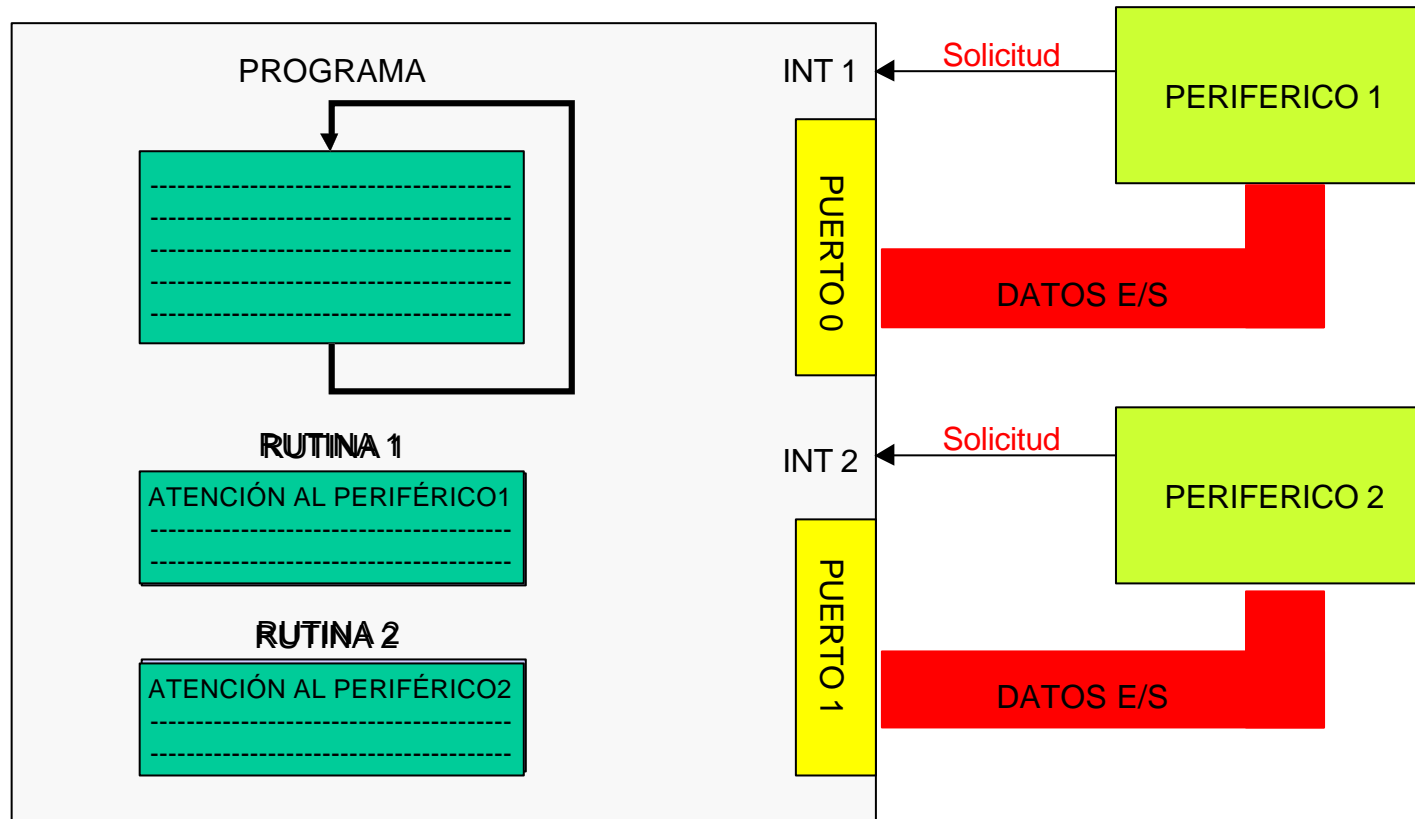




2 - Introducción

- ATENCIÓN DE PERIFÉRICOS MEDIANTE **INTERRUPCIONES**

INTERRUPCION





2 - Introducción

- El T89C51CC01 dispone de 14 fuentes de interrupción con 4 niveles de prioridad.
- Las que nos interesan ahora son:
 - Interrupciones generadas por los periféricos internos.
 - Temporizadores/Contadores 0, 1 y 2
 - Conversor A/D
 - Generadas por fuentes externas.
 - INT0, INT1
- Cada fuente de interrupción tiene asociado su propio **vector de interrupción** para localizar en memoria el **manejador**.
- La interrupción puede ser habilitada/inhibida individualmente.



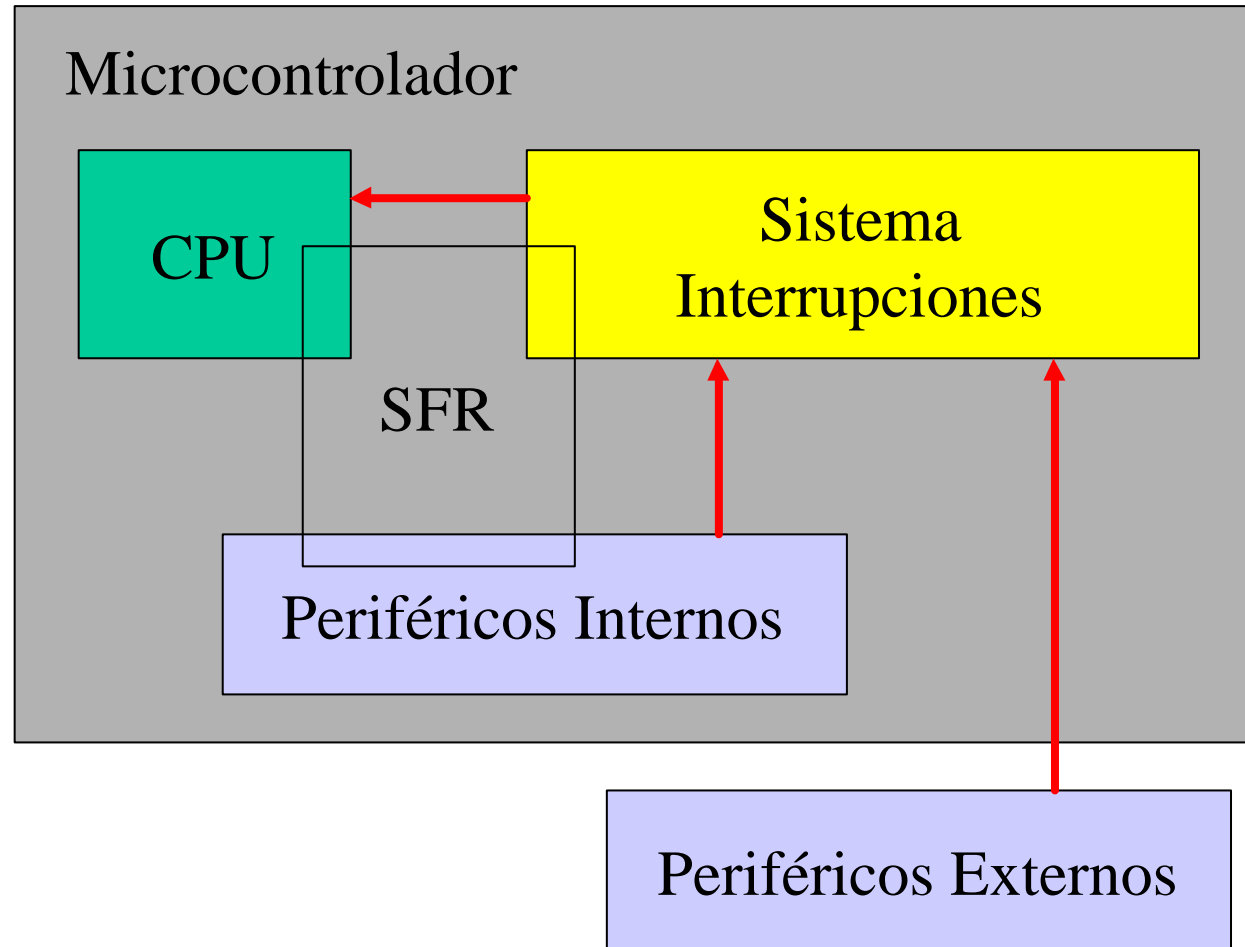


3 - Estructura de las interrupciones





3 - Estructura de las interrupciones





3 - Estructura de las interrupciones

- El microcontrolador para generar las interrupciones utiliza el siguiente mecanismo:
 - Cada fente de interrupción tiene sus propios **flags (bits)** asociados, localizados en un registro del SFR (p.e. TCON, ...), que permiten conocer cuándo se ha producido una petición de interrupción.
El flag correspondiente se pone a “1” cuando se produce la petición, aunque la interrupción no esté habilitada (p.e. overflow del timer/counter).
 - Para **habilitar/inhibir** cada interrupciones se utilizan ciertos bits de los registros: IEN0, IEN1.
 - Un bit global (IEN0.EAL) permite habilitar/inhibir **todo** el sistema de interrupciones. Se utiliza para no distorsionar en el proceso de configuración de las interrupciones.
 - Mediante los registros IPH0, IPL0, IPH1 e IPL1 se puede establece el nivel de **prioridad** asociado con cada fuente de interrupción





3 - Estructura de las interrupciones

- En la rutina de inicialización del sistema se deberán configurar las diferentes interrupciones (Establecer la prioridad y la habilitación).
- En Lenguaje C se utiliza el código siguiente para programar el manejador:

```
void manejador (void) interrupt Número_de_la_Interrupción  
{  
.....  
}
```

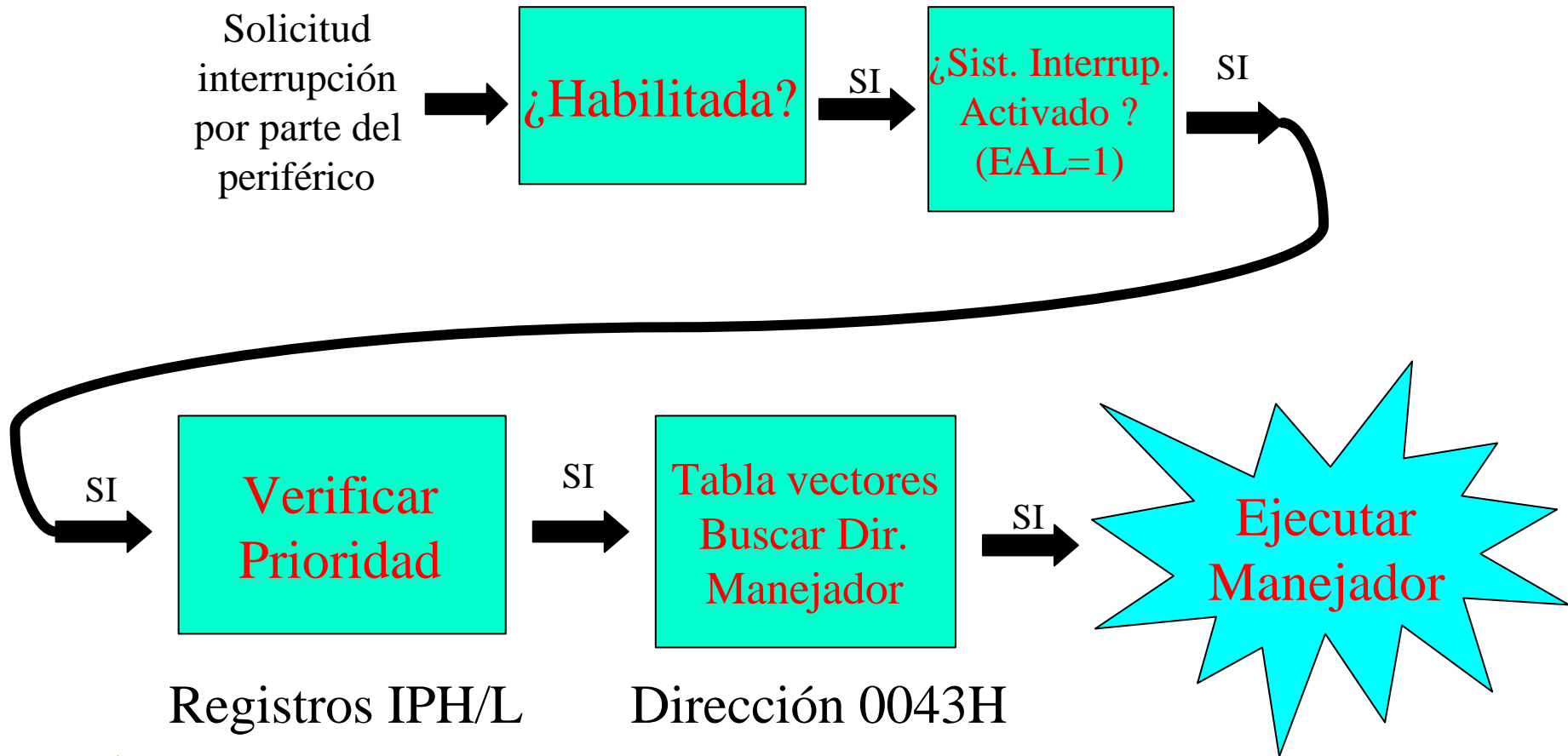




3 - Estructura de las interrupciones

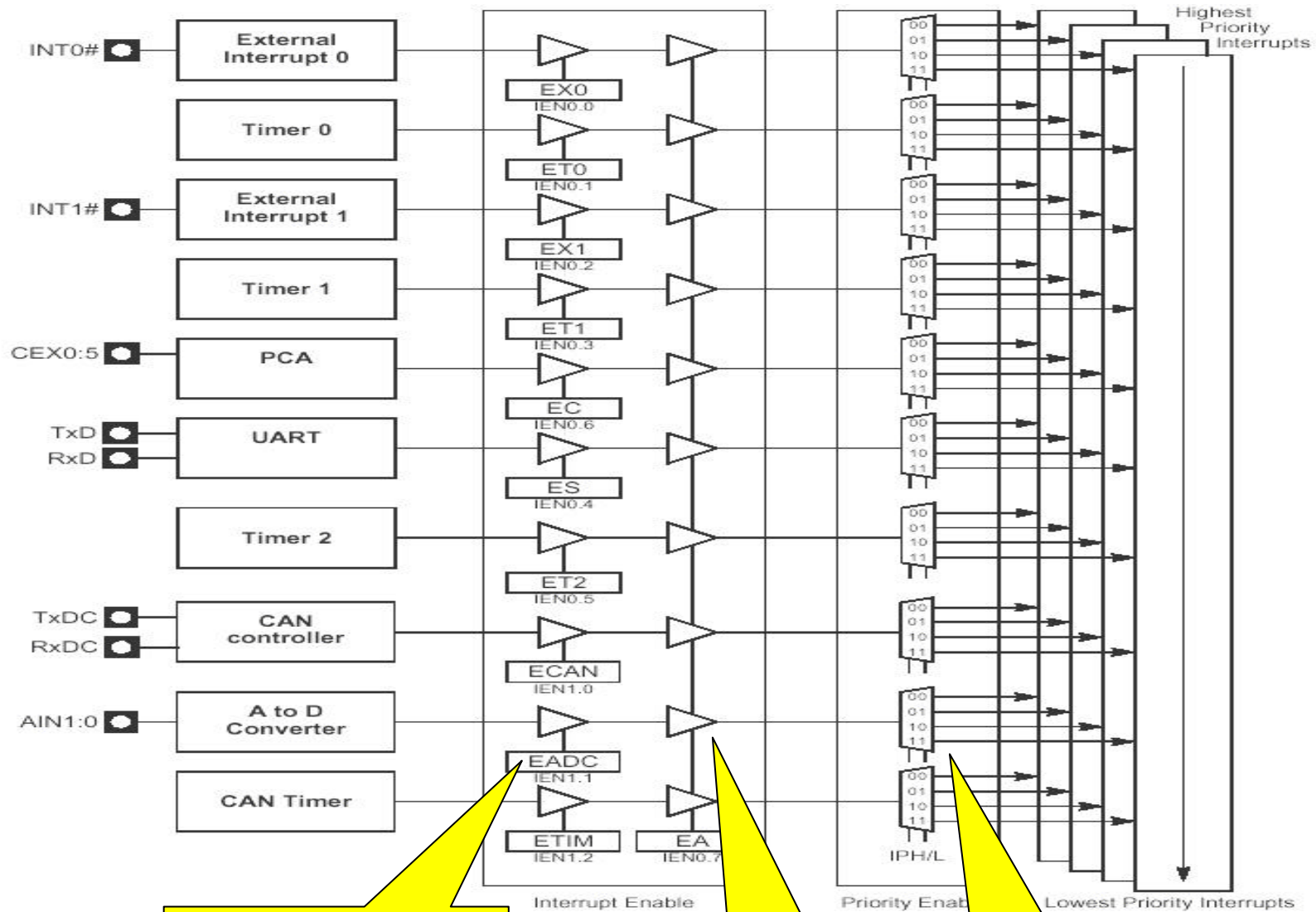
Ejemplo (Convertor A/D)

$IEN1.EADC = 1$ $IEN0.EAL = 1$





3 - Estructura de las interrupciones



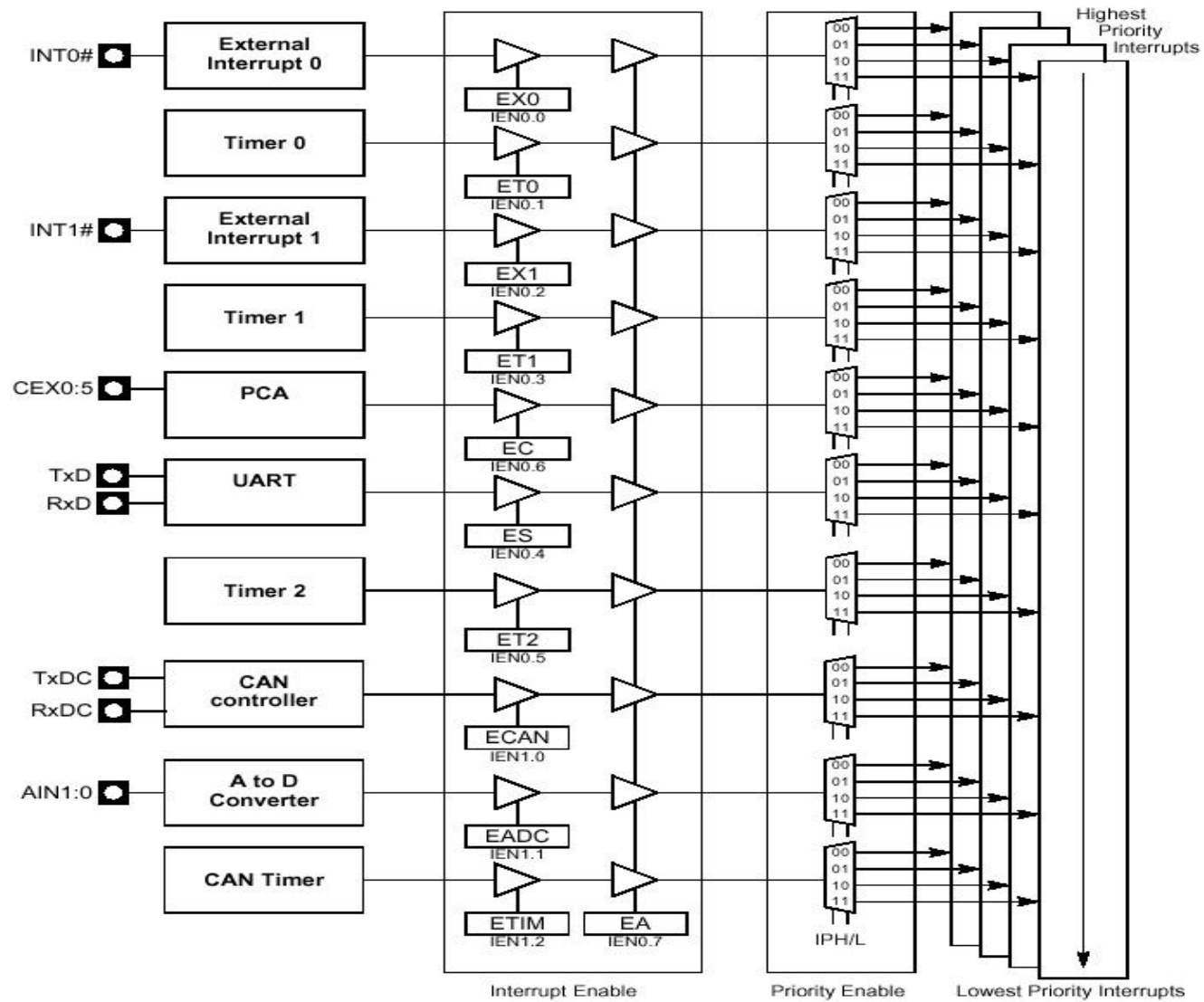
IEN1.EADC

IEN0.EAL

IPH/L



3 - Estructura de las interrupciones



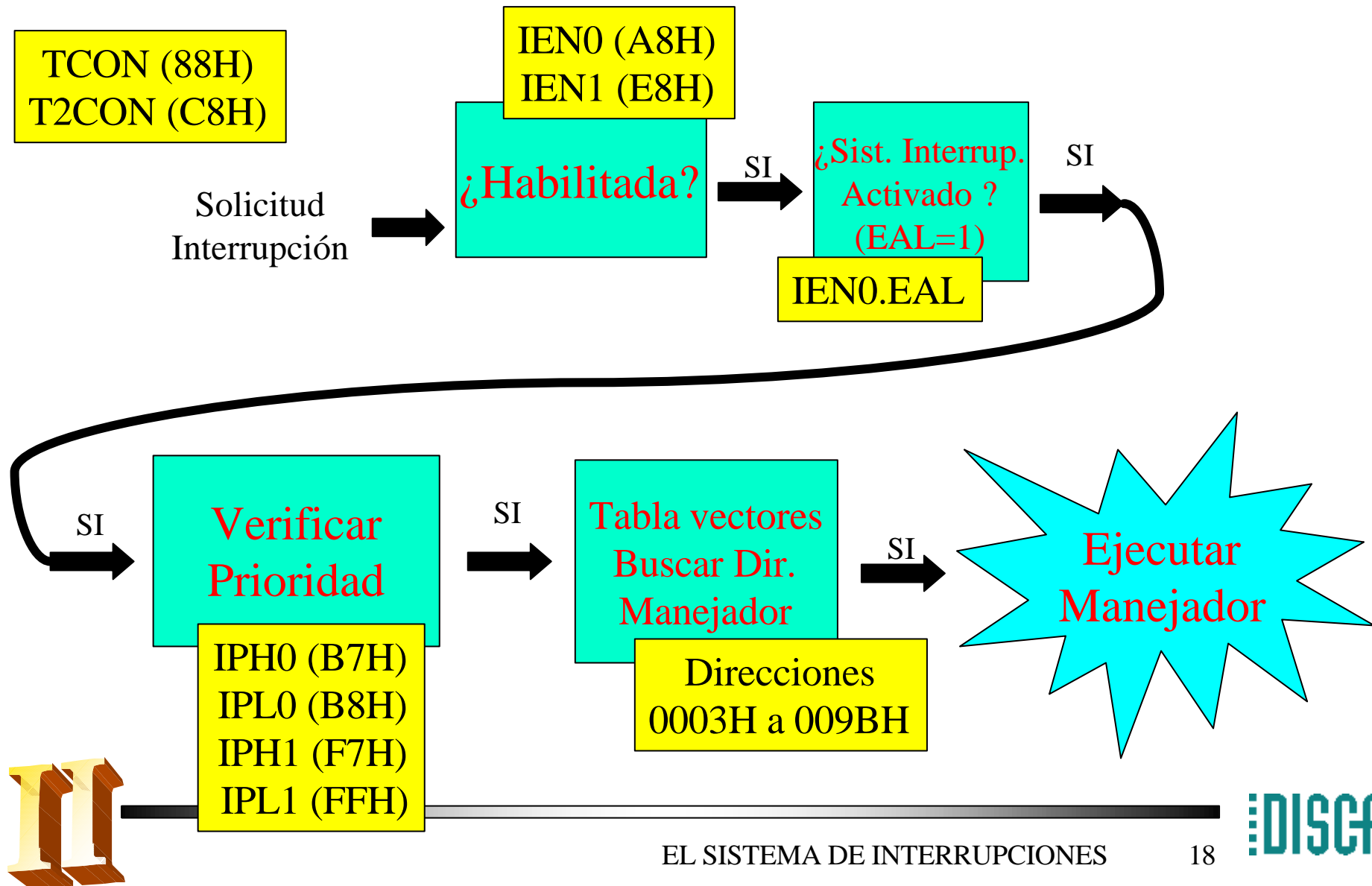


4 - Registros involucrados y fuentes de interrupción





4 - Registros involucrados y fuentes de interrupción



4 - Registros involucrados y fuentes de interrupción



- EX0
 - Permite o prohíbe la interrupción externa 0. Si EX0=0 INT0 prohibida.
- ET0
 - Permite o prohíbe la interrupción del timer 0. Si ET0=0 prohibida.
- EX1
 - Permite o prohíbe la interrupción externa 1. Si EX1=0 INT1 prohibida.
- ET1
 - Permite o prohíbe la interrupción del timer 1. Si ET1=0 prohibida.
- ES
 - Permite o prohíbe la interrupción del canal serie. Si ES=0 prohibida.
- ET2
 - Permite o prohíbe la interrupción del timer 2. Si ET2=0 prohibida.
- EC
 - Permite o prohíbe la interrupción de PCA. Si EC=0 prohibida.
- EAL
 - Permite o prohíbe todas la interrupciones. Si EAL=0 todas prohibidas, si EAL=1 cada una permitida o prohibida por su propio bit.





4 - Registros involucrados y fuentes de interrupción

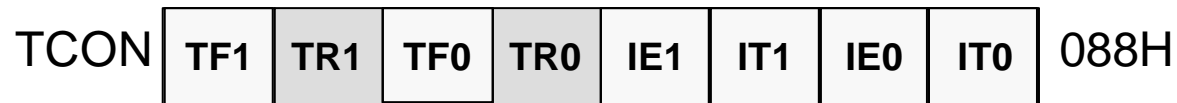


- ECAN
 - Permite o prohíbe la interrupción del CAN. Si ECAN=0 prohibida.
- EADC
 - Permite o prohíbe la interrupción del convertidor A/D. Si EADC=0 prohibida.
- ETIM
 - Permite o prohíbe la interrupción del Timer Overrun. Si ETIM=0 deshabilitada.





4 - Registros involucrados y fuentes de interrupción



- IT0
 - Selección de interrupción externa 0 (INT0) para que sea activo por flanco de bajada (“1”) o por nivel bajo (“0”)
- IE0
 - Flag de disparo de la INT 0. IE0=1 cuando se detecta int, IE0=0 cuando se procesa.
- IT1
 - Selección de interrupción externa 1 (INT1) para que sea activo por flanco de bajada (“1”) o por nivel bajo (“0”)
- IE1
 - Flag de disparo de la INT1. IE1=1 cuando se detecta int, IE1=0 cuando se procesa.
- TF0
 - Flag de desbordamiento del timer 0. Se pone a 1 por hardware cuando hay desbordamiento en el timer 0. A 0 por hardware cuando se atiende la interrupción.
- TF1
 - Flag de desbordamiento del timer 1. Se pone a 1 por hardware cuando hay desbordamiento en el timer 1. A 0 por hardware cuando se atiende la interrupción.

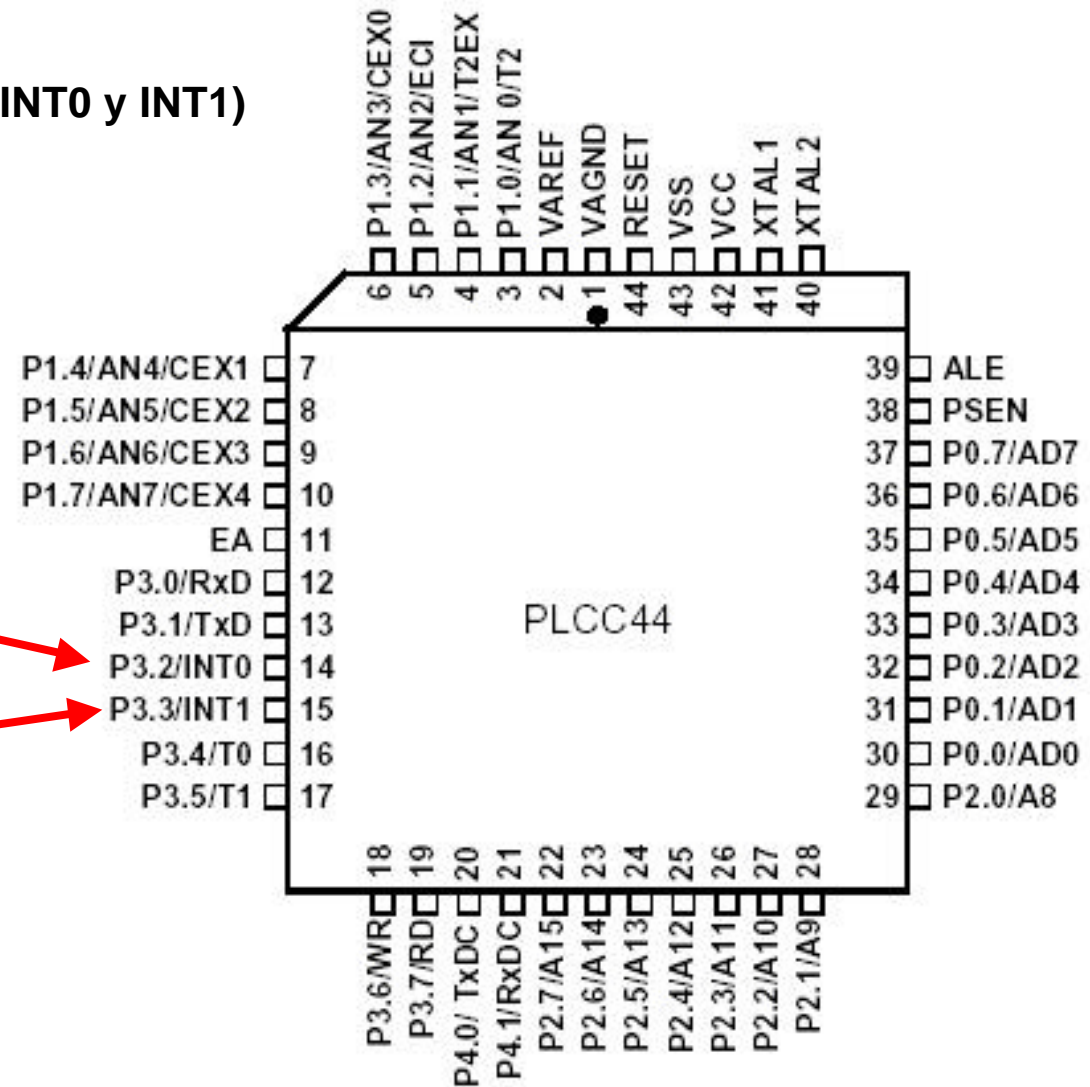


4 - Registros involucrados y fuentes de interrupción

INTERRUPCIÓN EXTERNA 0 Y 1 (INT0 y INT1)

**EXTERNA 0
(INT0)**

**EXTERNA 1
(INT1)**





5 - Estructura de los niveles de prioridad





5 - Estructura de los niveles de prioridad

Periférico 1

Rutina 1
Baja Prioridad

Periférico 2

Rutina 2
Alta Prioridad

Periférico 3

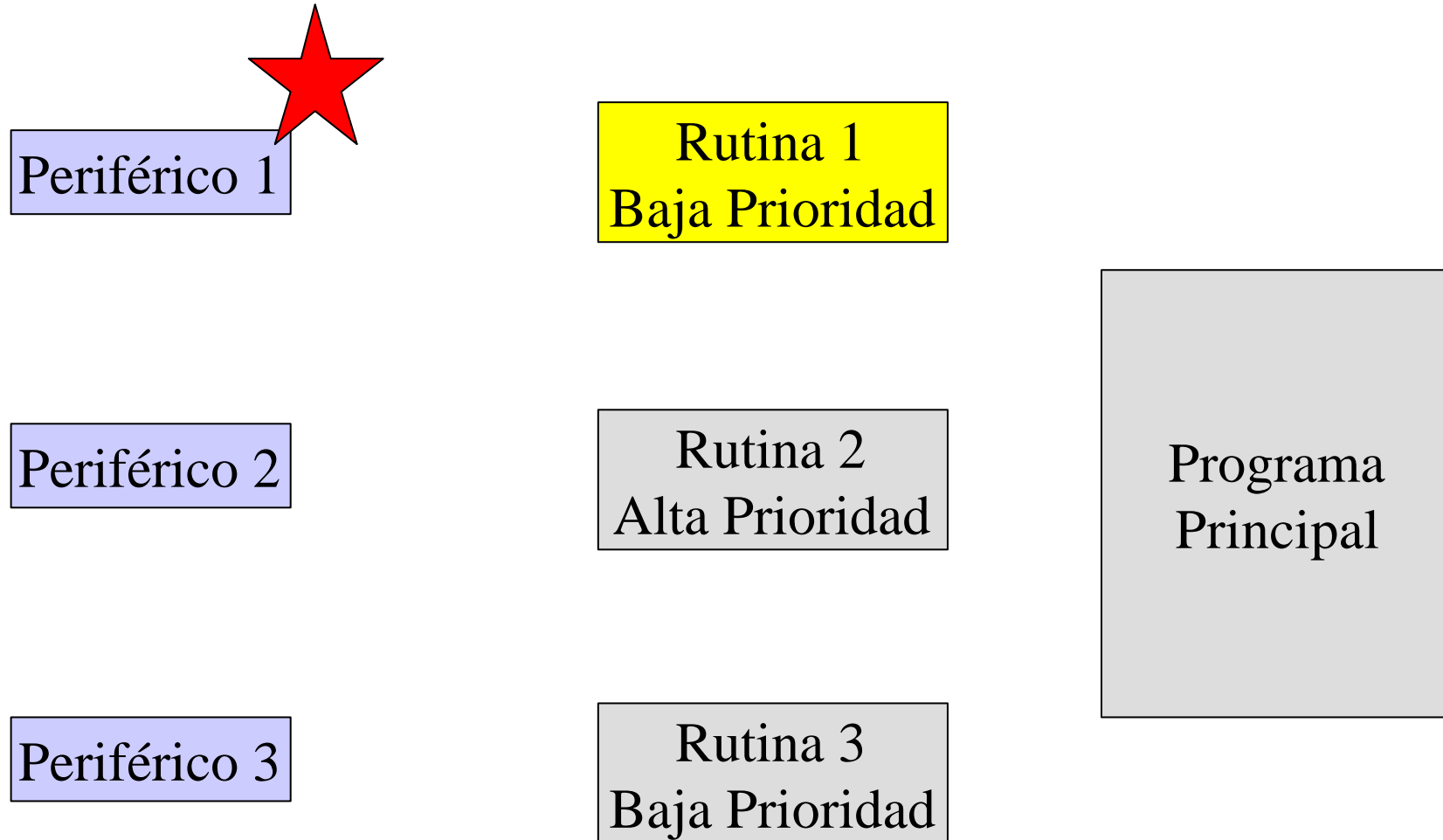
Rutina 3
Baja Prioridad

Programa
Principal



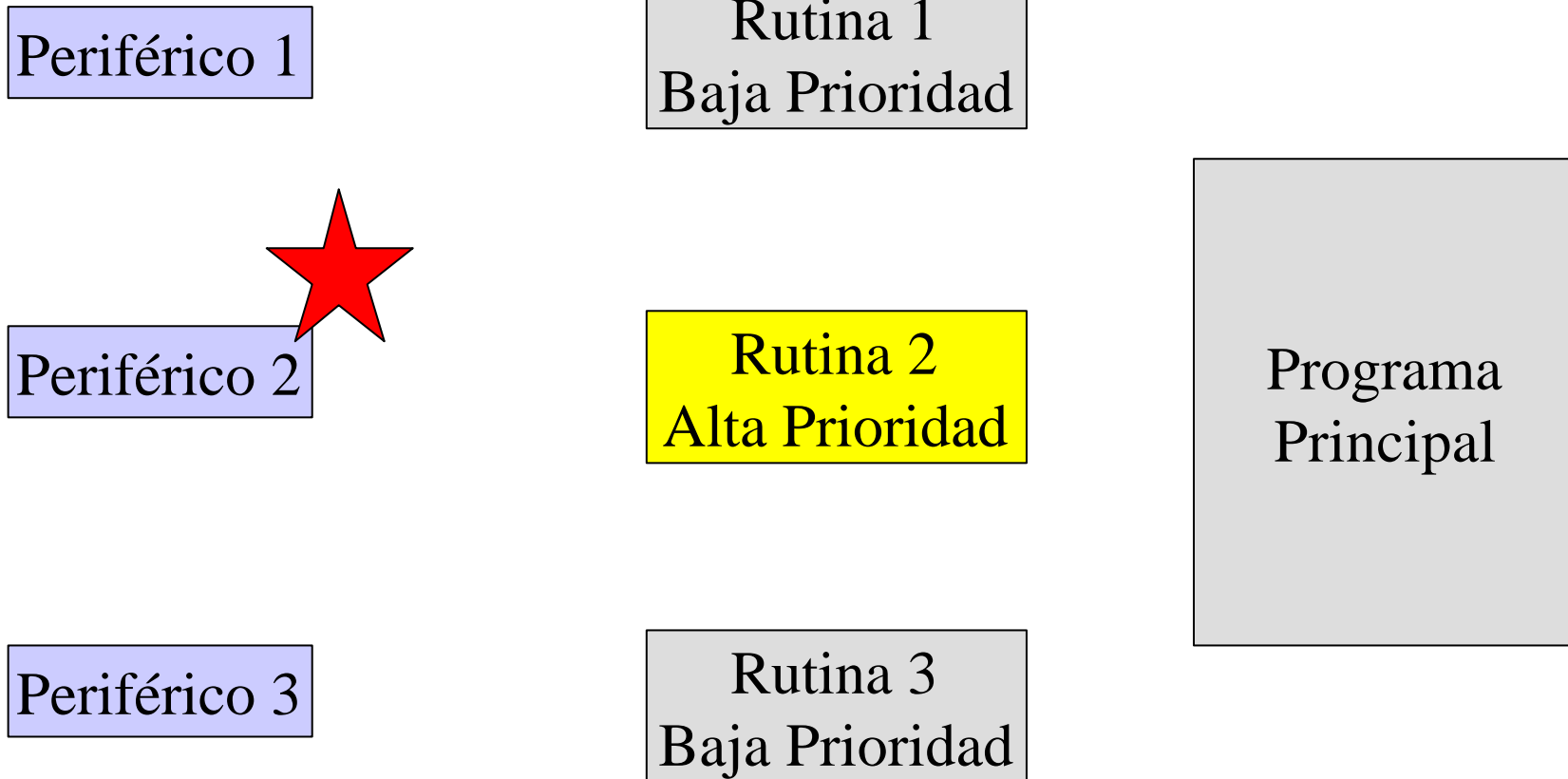


5 - Estructura de los niveles de prioridad



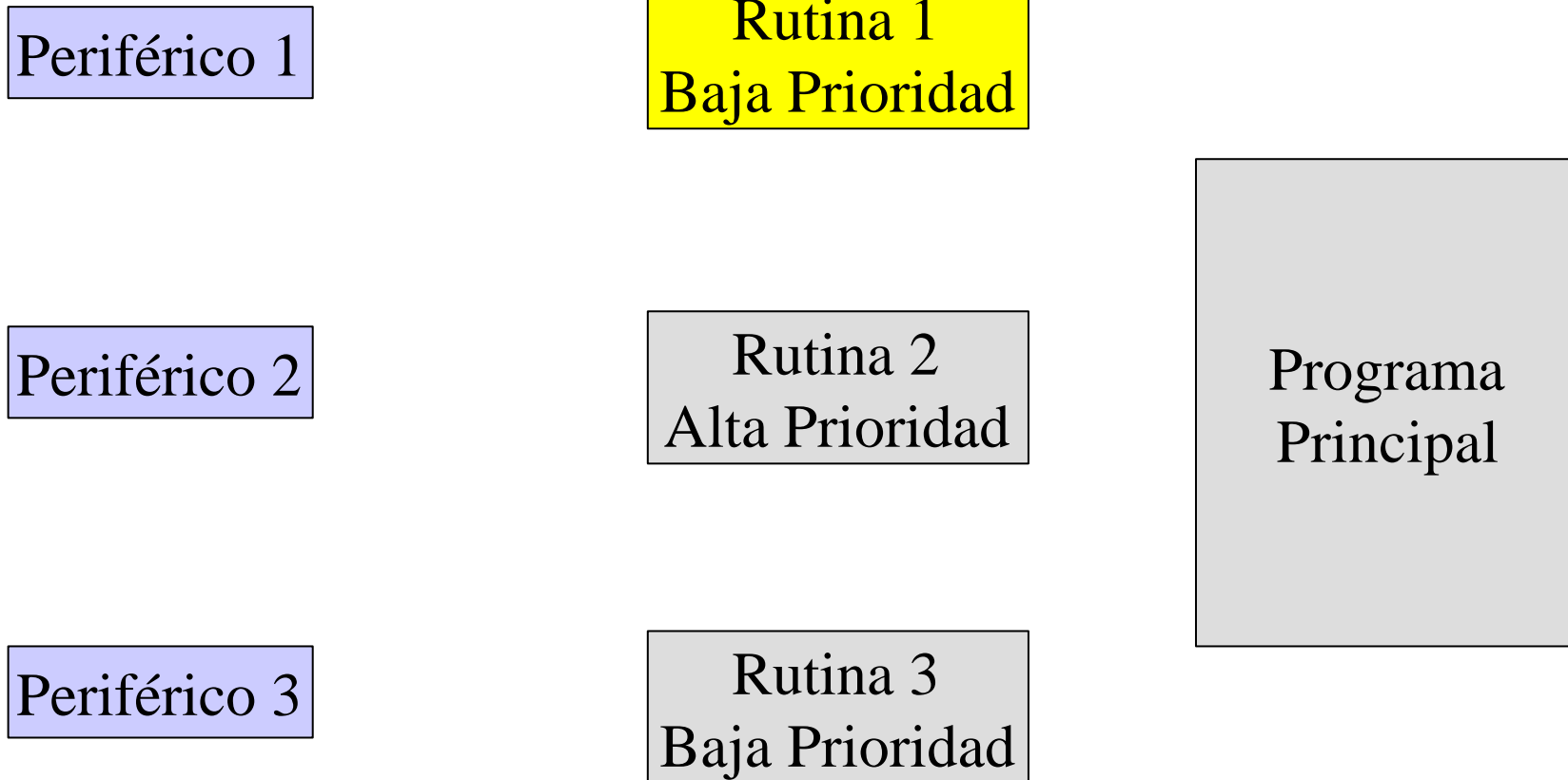


5 - Estructura de los niveles de prioridad





5 - Estructura de los niveles de prioridad





5 - Estructura de los niveles de prioridad

Periférico 1

Rutina 1
Baja Prioridad

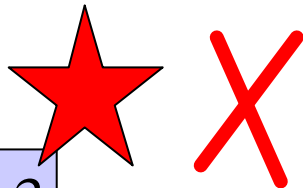
Periférico 2

Rutina 2
Alta Prioridad

Periférico 3

Rutina 3
Baja Prioridad

Programa
Principal





5 - Estructura de los niveles de prioridad

- La siguiente tabla recoge los valores que hay que dar en los registros para establecer los niveles de prioridad:

IPH.x	IPL.x	Nivel prioridad interrupción
0	0	0 (Menor)
0	1	1
1	0	2
1	1	3 (Mayor)





5 - Estructura de los niveles de prioridad

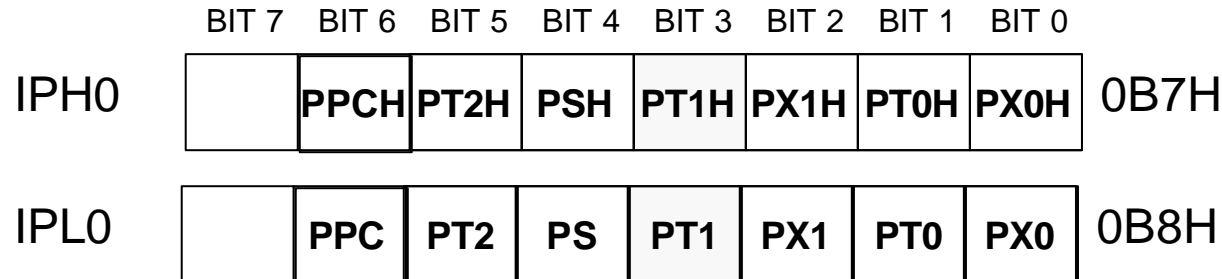
- Cada interrupción puede ser programado individualmente a uno de los 4 posibles niveles de prioridad, poniendo a “1” o “0” un bit del SFR IPH.x y del IPL.x.
- Una interrupción puede ser interrumpida por otra de mayor nivel de prioridad (no de igual o menor).
- Si se solicitan simultáneamente más de una interrupción, se atenderá la de mayor prioridad, si son del mismo nivel internamente se establece una secuencia.





5 - Estructura de los niveles de prioridad

- La estructura de los registros IP0 e IP1 es:



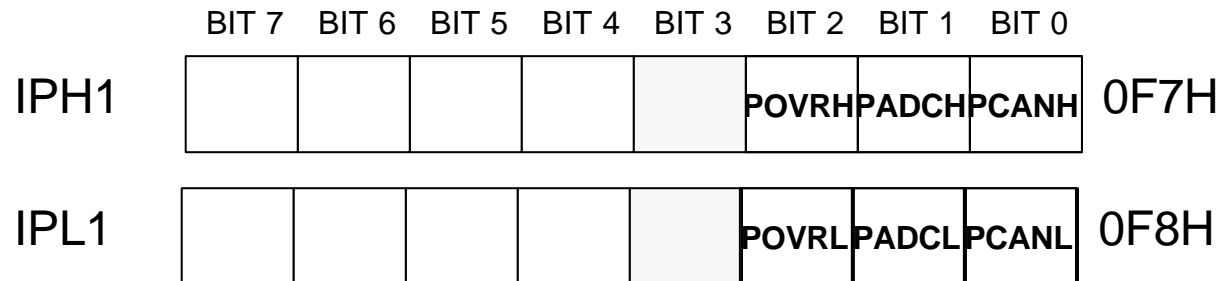
- PX0H - PX0
 - Prioridad interrupción externa 0.
- PT0H - PT0
 - Prioridad interrupción desbordamiento Temporizador/Contador 0.
- PX1H - PX1
 - Prioridad interrupción externa 1.
- PT1H - PT1
 - Prioridad interrupción desbordamiento Temporizador/Contador 1.
- PSH - PS
 - Prioridad interrupción puerto serie.
- PT2H - PT2
 - Prioridad interrupción desbordamiento Temporizador/Contador 2.
- PPCH - PPC
 - Prioridad interrupción PCA.





5 - Estructura de los niveles de prioridad

- La estructura de los registros IP0 e IP1 es:

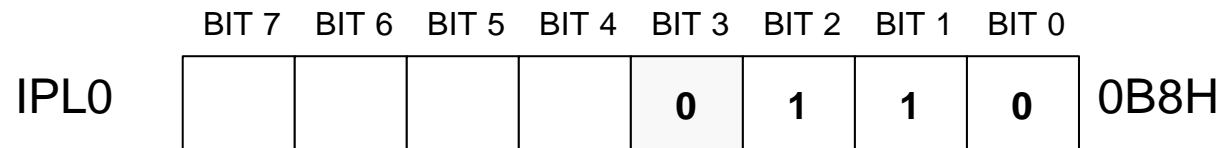
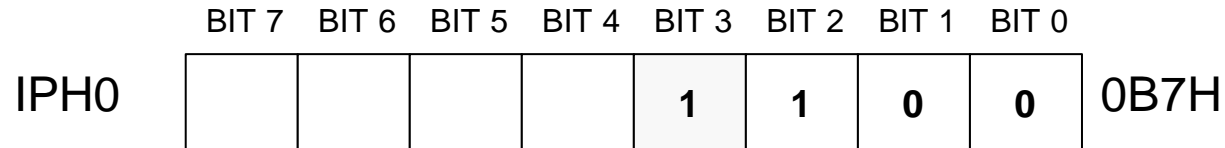


- PCANH - PCANL
 - Prioridad interrupción bus CAN.
- PADCH - PADCL
 - Prioridad interrupción convertidor AD.
- POVRH - POVRL
 - Prioridad interrupción overrun temporizador.





5 - Estructura de los niveles de prioridad



IPH.x	IPL.x	
0	0	Asignar el nivel de prioridad 0 (bajo)
0	1	Asignar el nivel de prioridad 1
1	0	Asignar el nivel de prioridad 2
1	1	Asignar el nivel de prioridad 3 (alto)

- A la vista de los valores asignados a IPH0 e IPL0:
 - Interrupción externa 0 con prioridad 0 (la menor).
 - Interrupción Temporizador/Contador 0 con prioridad 1.
 - Interrupción externa 1 con prioridad 3 (la mayor).
 - Interrupción Temporizador/Contador 1 con prioridad 2.





5 - Estructura de los niveles de prioridad

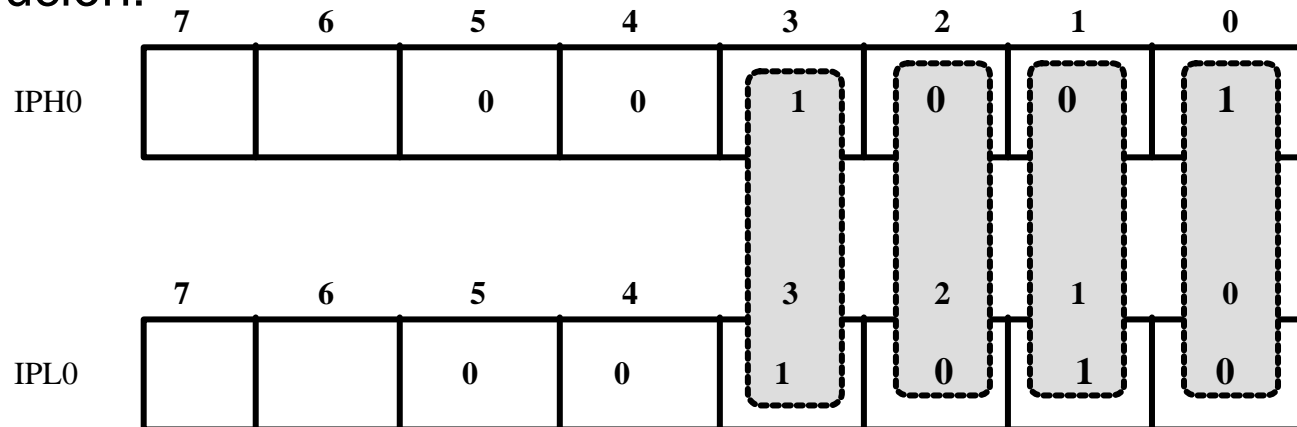
• ACTIVIDAD:

– Se desea asignar los siguientes niveles de prioridad a ciertas interrupciones:

- Interrupción externa-0: Prioridad 2
- Interrupción timer/counter-0: Prioridad 1
- Interrupción externa 1: Prioridad 0
- Interrupción timer/counter 1: Prioridad 3

† Indicar a qué valor deberán inicializarse los registros IPH0 e IPL0 para conseguirlo

• Solución:



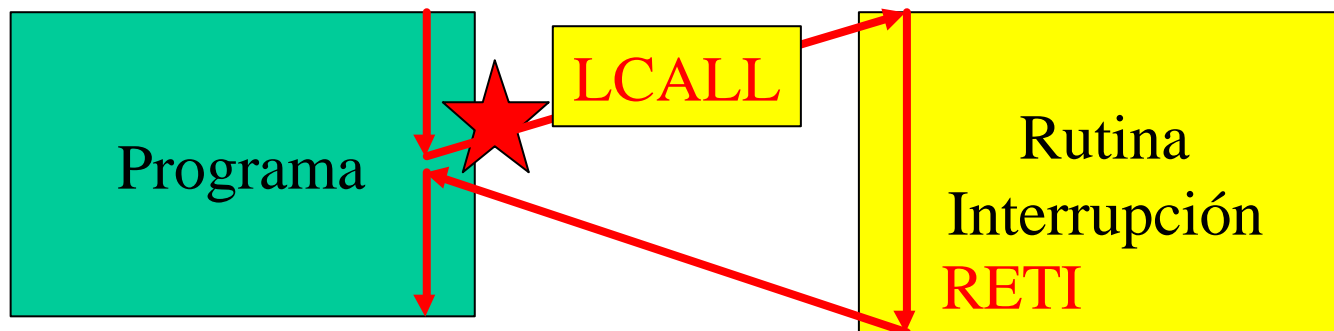


6 - Gestión de las Interrupciones



6 - Gestión de las Interrupciones

- Si alguno de los bits (siguiendo la prioridad) está a “1”, se genera una instrucción “**LCALL**” al manejador asociado.



- Para que se cumpla lo anterior deben darse tres condiciones:
 - No estar ejecutando un manejador de otra interrupción de igual o mayor prioridad.
 - Haber terminado la ejecución de la instrucción en curso.
 - La instrucción en ejecución NO debe ser: “RETI” o cualquier acceso a los registros IEN1, IEN2, IPHx, IPLx.



6 - Gestión de las Interrupciones

TABLA DE VECTORES DE INTERRUPCIÓN:

Flags petición interrupción	Dirección vector interrupción	Número Interrupción	Fuente de Interrupción
IE0	0003H	0	Interrupción externa 0
TF0	000BH	1	Timer/Counter 0 overflow
IE1	0013H	2	Interrupción externa 1
TF1	001BH	3	Timer/Counter 1 overflow
RI0/TI0	0023H	4	Canal serie 0
TF2/EXF2	002BH	5	T/C 2 overflow/recarga ext.
IADC	0043H	8	Convertidor A/D
CAN Timer Ovf	004BH	9	Overflow Timer CAN
CAN	003BH	7	CAN
PCA	0033H	6	PCA

```
void manejador (void) interrupt N°  
{  
  
.....  
}
```





6 - Gestión de las Interrupciones

- La instrucción “**LCALL**” generada por el Hardware cuando hay una interrupción, guarda en la pila el valor del PC (Program Counter) y lo actualiza con el contenido del vector de interrupción correspondiente:

Flags petición interrupción	Dirección vector interrupción	Número Interrupción	Fuente de Interrupción
IE0	0003H	0	Interrupción externa 0
TF0	000BH	1	Timer/Counter 0 overflow
IE1	0013H	2	Interrupción externa 1
TF1	001BH	3	Timer/Counter 1 overflow
RI0/TI0	0023H	4	Canal serie 0
TF2/EXF2	002BH	5	T/C 2 overflow/recarga ext.
IADC	0043H	8	Convertidor A/D
CAN Timer Ovf	004BH	9	Overflow Timer CAN
CAN	003BH	7	CAN
PCA	0033H	6	PCA

- La ejecución comienza desde esa posición de memoria y sigue hasta encontrar la instrucción “**RETI**”, devolviendo al PC su valor original (sacar valor de la pila).





7 - Ejemplos de aplicación





7 - Ejemplos de aplicación

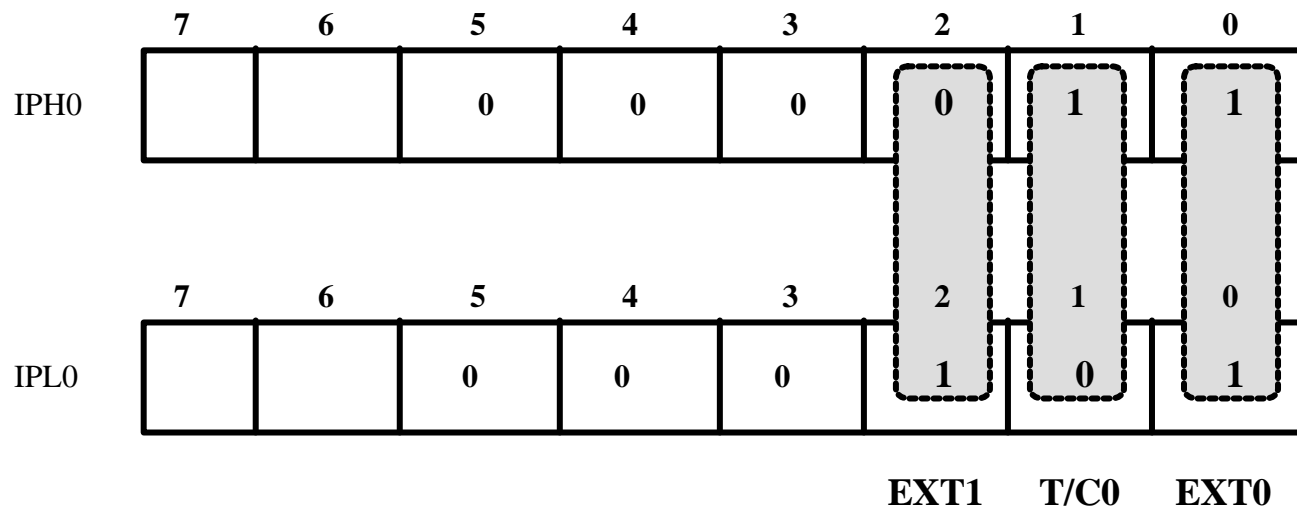
- **Actividad:**

- Se desea asignar los siguientes niveles de prioridad a ciertas interrupciones:

- Interrupción externa-0: Prioridad 3
- Interrupción Timer/Counter-0: Prioridad 2
- Interrupción externa-1: Prioridad 1

- † Indicar a qué valor deberán inicializarse los registros IP0 e IP1 para conseguirlo

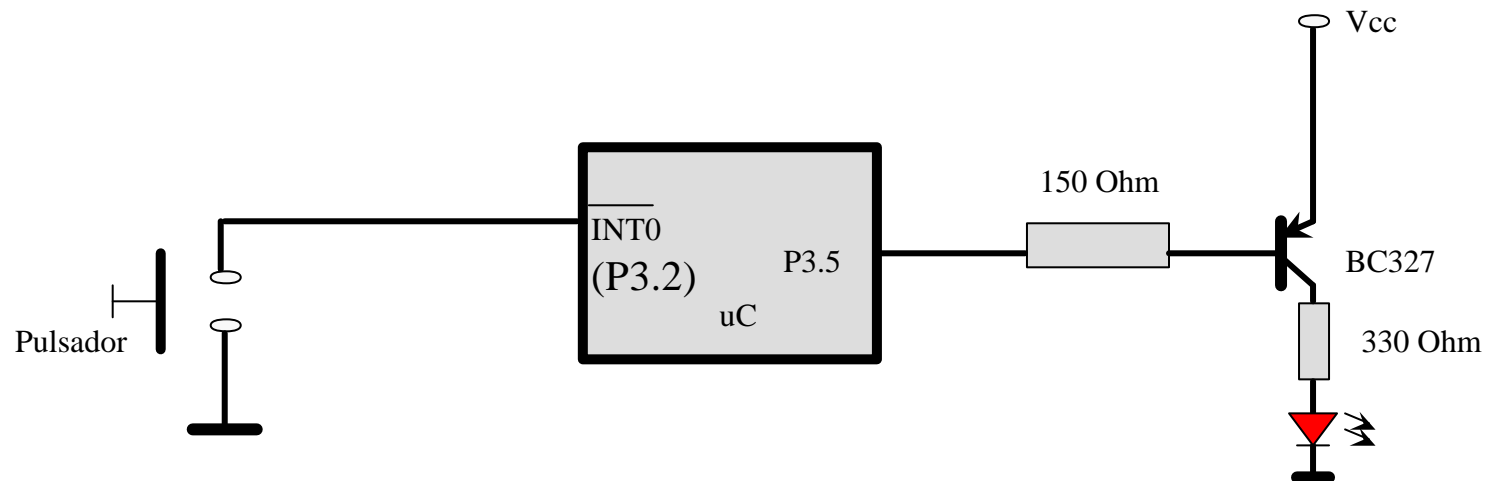
- **Solución:**





7 - Ejemplos de aplicación

- **Actividad:**
 - Sea el esquema de la figura:



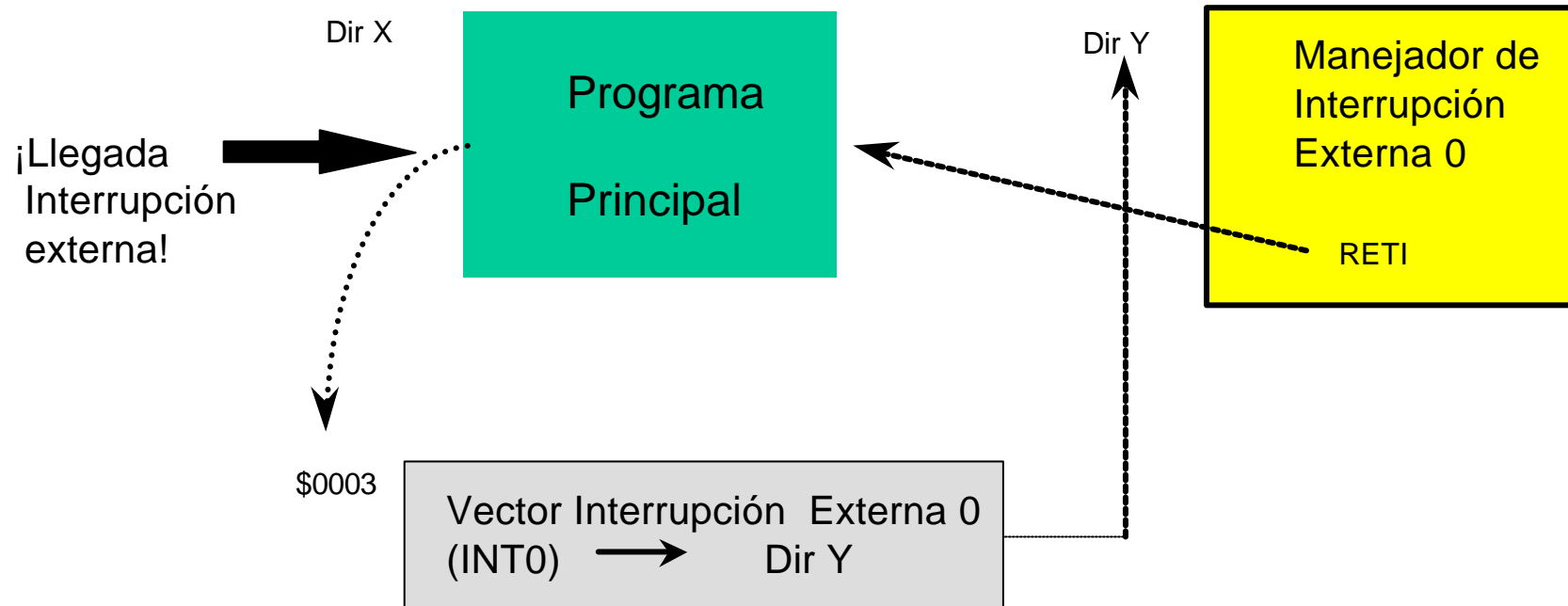
- Realizar un programa que detecte mediante una interrupción la pulsación del pulsador.
- Cada vez que se pulse, se deberá ejecutar el manejador asociado a la INT0 que invertirá el valor del terminal de salida P3.5, a la vez que llevará la cuenta de las veces que se ha presionado el pulsador.





7 - Ejemplos de aplicación

- Solución:





7 - Ejemplos de aplicación

```
IEN0      data      0A8h
TCON      data      88h
P3        data      0B0h

          org      0000h          ; Comienzo programa
          ljmp     P_PRINCIPAL    ; Saltar al p. principal

          org      0003h          ; Inicializa vector de interrupción
          ljmp     MANEJADOR     ; Comenzar ejecutar manejador

P_PRINCIPAL:
          clr      IEN0.7        ; EAL = 0 por si estaba activo antes, no hace falta
          setb    IEN0.0        ; EX0 = 1
          mov     R7,#0         ; Contador pulsaciones a 0
          setb    TCON.0        ; Interrup. externa 0 por flanco bajada
          setb    IEN0.7        ; EAL = 1, Interrupciones activadas
BUCLE:    sjmp    BUCLE        ; Bucle sin fin

MANEJADOR:
          cpl     P3.5          ; Invertir LED
          inc     R7            ; Contar veces que se ha pulsado
          reti

          end
```





7 - Ejemplos de aplicación

```
sfr P3=0XB0;
Sbit P3_5 = P3^5;
sbit EAL=0xA8^7;
sbit EX0=0xA8^0;
sbit IT0=0x88^0;
unsigned char contador;
```

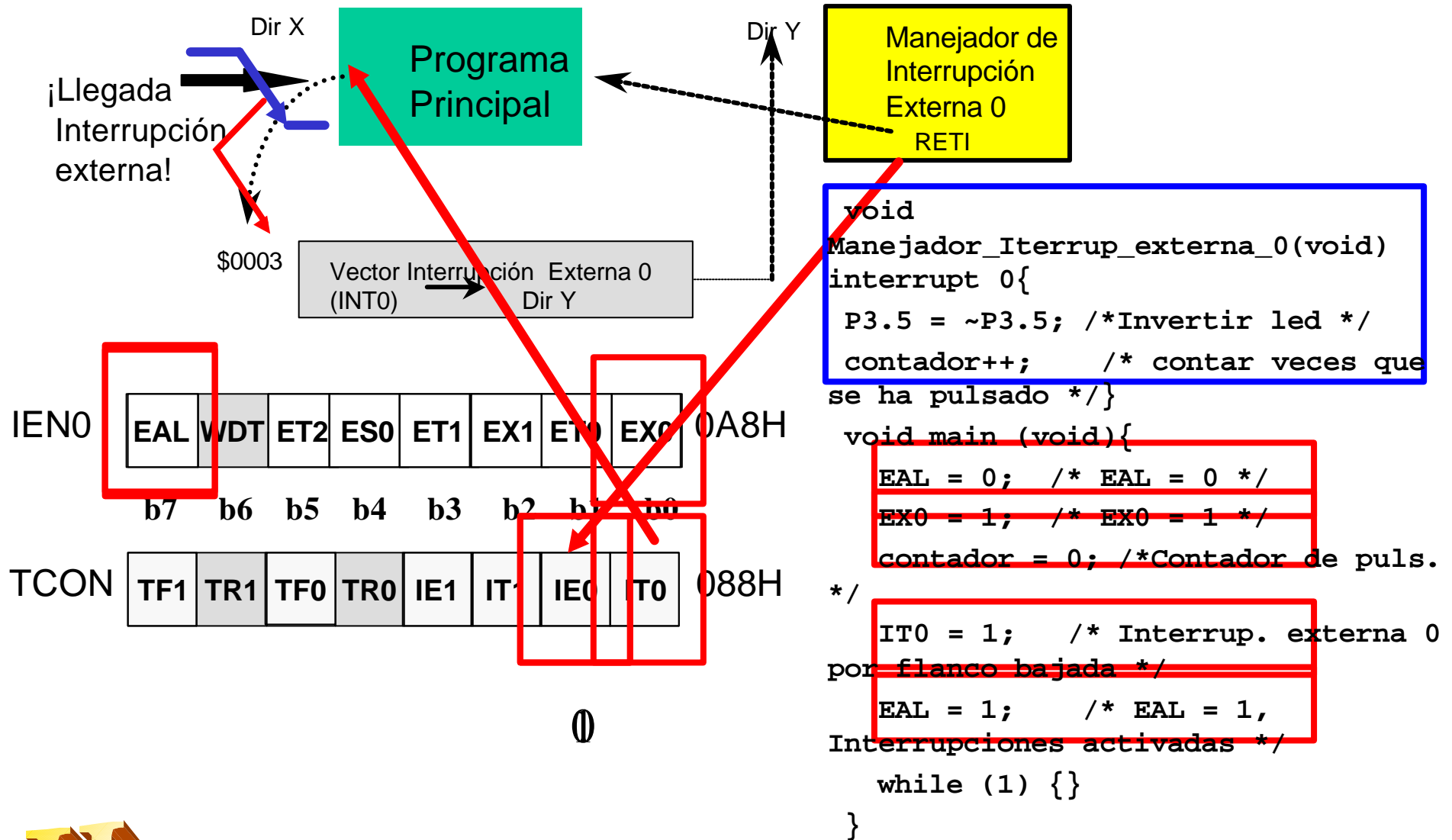
```
void Manejador_Interrupt_externa_0(void) interrupt 0
{
    P3_5 = ~P3_5; /* Invertir led */
    contador++; /* contar veces que se ha pulsado */
}
```

```
void main (void)
{
    EAL = 0; /* EAL = 0 */
    EX0 = 1; /* EX0 = 1 */
    contador = 0; /* Contador de pulsaciones */
    IT0 = 1; /* Interrup. externa 0 por flanco bajada */
    EAL = 1; /* EAL = 1, Interrupciones activadas */
    while (1) {}
}
```





7 - Ejemplos de aplicación





7 - Ejemplos de aplicación

- **Actividad:**

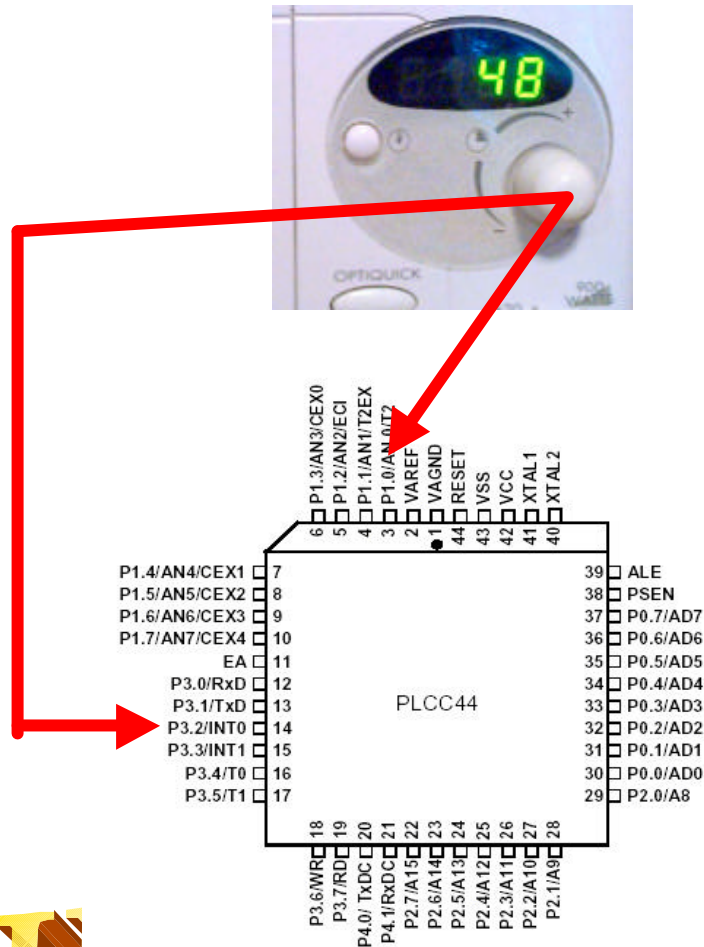
- Realizar la gestión de la visualización del display de la figura (conectado al puerto P2 de un uC MCS-51) en base al uso interrupciones de modo que cada paso de giro del potenciómetro genere la interrupción externa 0 que incrementará/decrementará la cuenta. Mediante el bit de E/S P1.0 se sabe el sentido de giro del potenciómetro (P1.0=1 gira a derechas, P1.0=0 gira a izquierdas).



7 - Ejemplos de aplicación

- **Solución:**

- Sea el esquema de la figura:



```
sfr IEN0=0xA8;
sbit EX0=IEN0^0;
sbit EAL=IEN0^7;
sfr TCON=0x88;
sbit IT0=TCON^0;
sfr P1=0x90;
sfr P2=0xA0;
sbit P1.0=P1^0;
int cuenta = 50;
```

```
void manejador(void) interrupt 0{
    if (P1.0 == 1)
        {if (cuenta < 99) cuenta++;}
    else
        {if (cuenta > 0) cuenta--;}
}
```

```
void main(void) {
    EAL=0;
    IT0=1;
    EX0=1;
    EAL=1;
    while(1)
        P2=((cuenta/10)<<4) | (cuenta%10);
}
```





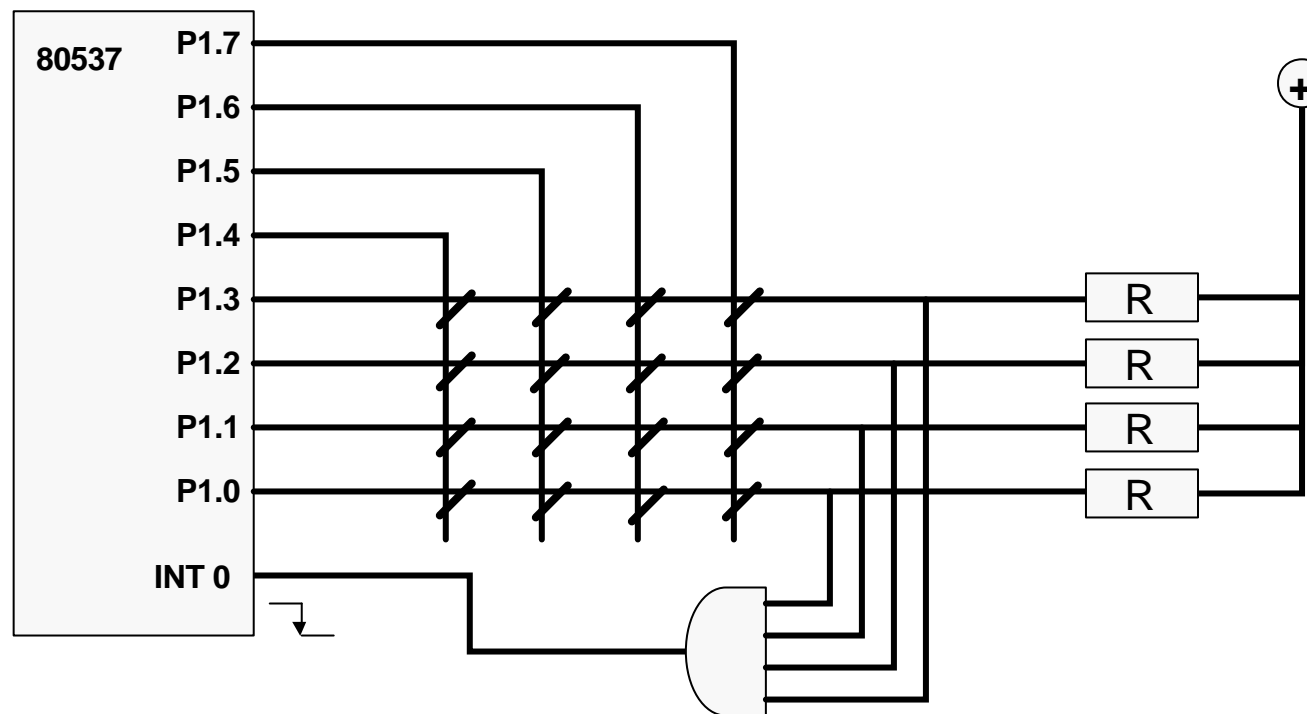
8 - Ejercicios propuestos



8 - Ejercicios propuestos

• CUESTION

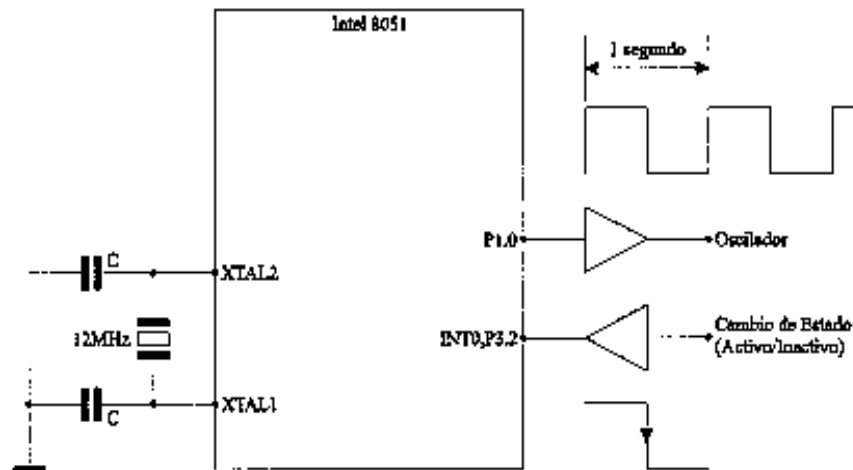
- Dado el sistema de la figura, el reconociendo de la tecla pulsada es a través del puerto P1 cuando se reciba una interrupción externa INT0, y visualizando a través del puerto P3, en formato 7 segmentos, el código de la misma (0,1,.....d,e,f).
- Indicar cómo se debería hacer un manejador de interrupción y escribir un boceto del cuerpo de programa principal.



8 - Ejercicios propuestos

• CUESTION

- Se dispone de un sistema basado en un μ C T89C51CC01 funcionando a una frecuencia de 12Mhz. En una aplicación de control, se necesita generar a través del P1.0 una señal cuadrada simétrica de periodo $\tau_t = 1$ segundo (los niveles TTL del P1.0 son adaptados por la circuitería externa). La señal debe generarse únicamente cuando el oscilador esté en Estado Activo; cuando está en Estado Inactivo el nivel del P1.0 debe ser permanentemente bajo. La circuitería externa avisa de la conmutación entre Estado Activo y Estado Inactivo mediante un flanco de bajada en la entrada de interrupción INT0 (P3.2). Para descargar el microcontrolador en lo posible, se pretende detectar el cambio entre Estado Activo y Estado Inactivo mediante el control de interrupción (no por encuesta), y realizar la temporización haciendo el mayor uso posible de los temporizadores integrados (aunque se tenga que recurrir, en parte, a rutinas de espera ocupada).
- Se pide el código ensamblador MCS-51 que resuelva la función anterior, justificando la solución adoptada mediante el uso de comentarios sobre el código fuente, el dibujo de flujogramas, y la descripción de los cálculos realizados.





8 - Ejercicios propuestos

- CUESTION

- Disponemos de un contador binario de 8 bits que tiene una salida OVF, de forma que $OVF=1$ cuando las salidas del contador pasan de 1111 1111 a 0000 0000. Cada vez que se active OVF se deberá producir una interrupción INT0 en un microcontrolador Atmel T89C51CC01.
- Se pide:
 - Dar un esquema de interconexión de la salida OVF del contador y el uC.
 - Indicar los registros que se deberán programar para que se reconozca la interrupción
 - Dar un resumen de cómo sería un manejador de esta interrupción y del vector correspondiente.





9 - Bibliografía





9 - Bibliografía

- <http://www.atmel.com>
- Introducción a los Microcontroladores; Hardware, Software y Aplicaciones; 8x52, 8x51. José Adolfo González Vázquez. McGraw-Hill
- 8XC51/80C31: 8-bit CMOS microcontroller families. Intel Preliminary specification. 1997 May 30
- The 8051 family of microcontrollers. Richard H. Barnett. Prentice Hall, 1995





10 - Apéndice





10 - Apéndice

– INTERRUPTIÓN DEL TIMER-0 y TIMER-1

- Generadas por los bits TCON.TF0; TCON.TF1, se ponen a “1” cuando sus registros asociados rebosan.
- TCON.TF0 y TCON.TF1 son puestos a “0” automáticamente por el hardware cuando se accede al manejador asociado.

– INTERRUPTIONES PUERTOS SERIE 0 y 1

- Se generan por la operación OR de los bits:
 - S0CON.RI0 or S0CON.TI0
 - S0CON.RI1 or S0CON.TI1
- Al ejecutar el manejador no se ponen a “0” los flags. Debe hacerse por SW.
- El manejador deberá verificar si la interrupción se ha generado por transmisión o recepción de un carácter por el puerto serie correspondiente.

