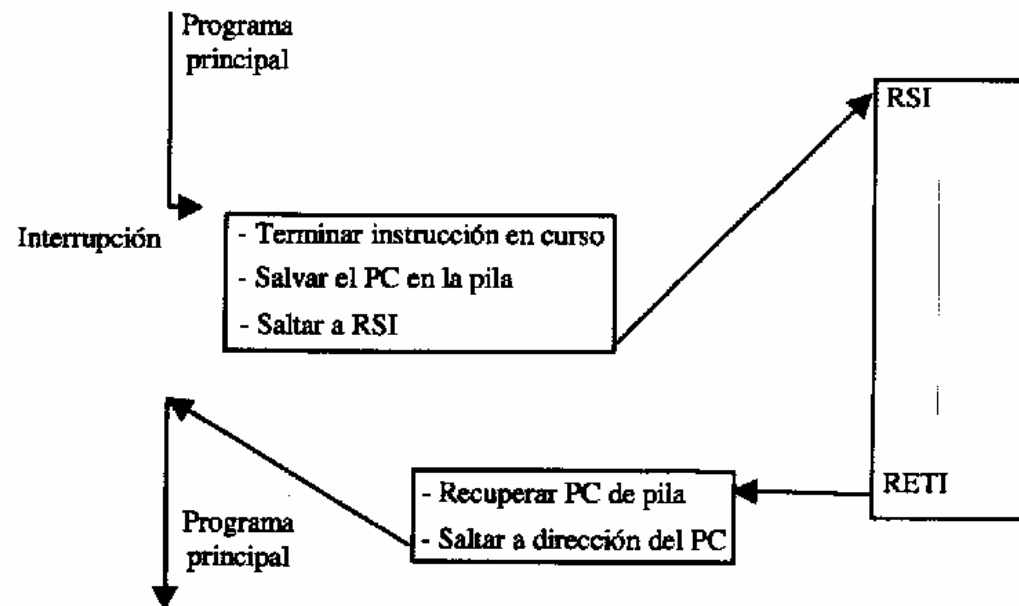


TEMA 3

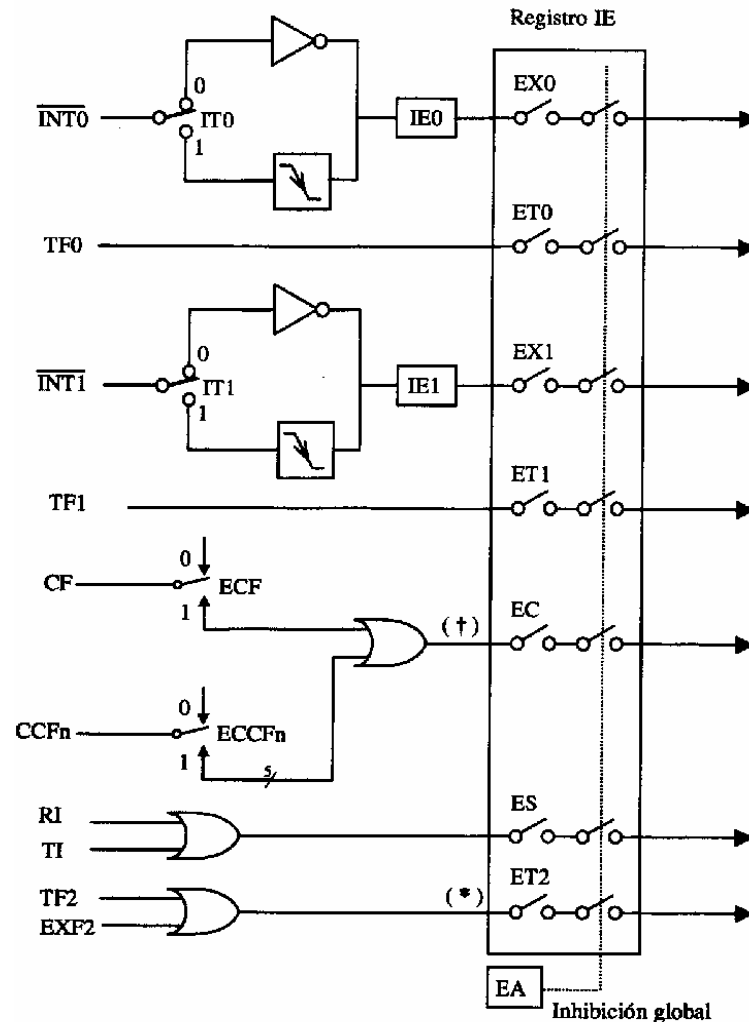
Interrupciones y Temporizadores

Proceso de Atención a la interrupción

1. Termina de ejecutar la instrucción en curso.
2. Guarda PC en la pila.
3. La CPU salta a la dirección de la RSI.
4. La rutina RSI termina con RETI.



Fuentes de Interrupción en 8051



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

* En las versiones con 3 temporizadores

† En las versiones con PCA

Registro TCON

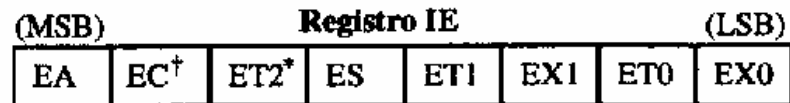
(MSB)		Registro TCON				(LSB)	
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

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.

Vectorización de las interrupciones en el 8051

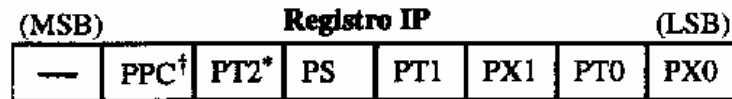
```
/******  
Vectorización de interrupciones  
*****/  
  
void rsi_int0(void) interrupt 0 {  
...  
... // atención a la interrupción situada en el vector n*8+3  
... // donde n es el número detrás de "interrupt"  
}  
  
void rsi_timer0(void) interrupt 1 {  
...  
...  
}  
  
void rsi_int1(void) interrupt 2 {  
...  
...  
}  
  
void main(void) {  
...  
... // programa principal  
...  
}
```

Registro IE



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

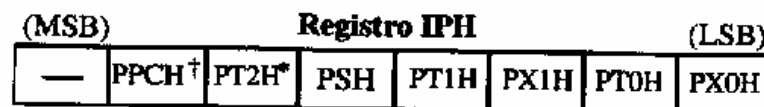
Gestión de las prioridades de las interrupciones



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

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

Registro adicional de prioridades en 8052



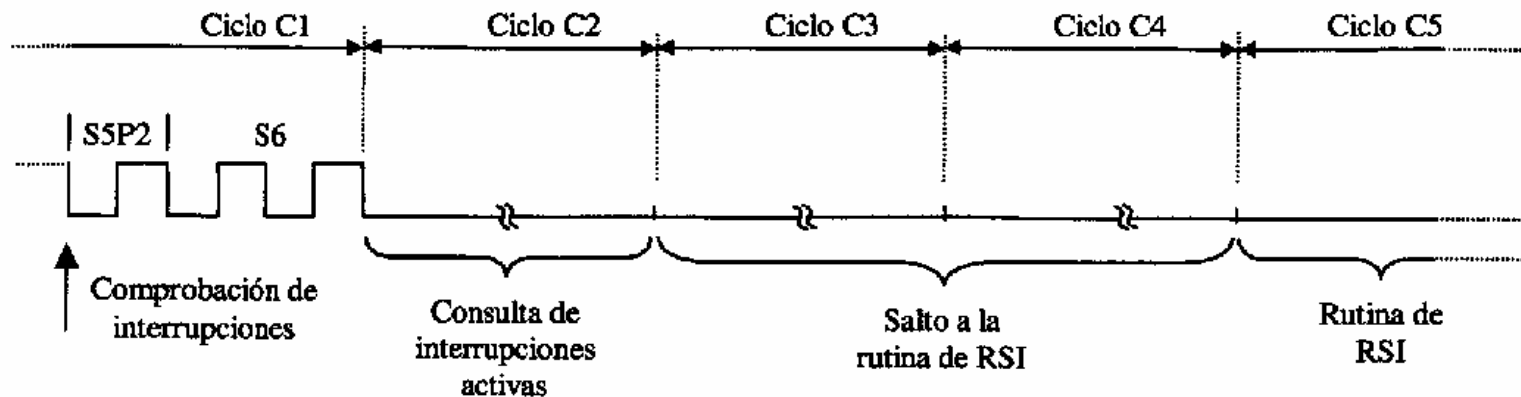
Bit	Comentario
PX0H	Bit alto de prioridad de /INT0
PT0H	Bit alto de prioridad del Timer 0
PX1H	Bit alto de prioridad de /INT1
PT1H	Bit alto de prioridad del Timer 1
PSH	Bit alto de prioridad del puerto serie
PT2H	Bit alto de prioridad del Timer 2
PPCH	Bit alto de prioridad de la PCA
-	Bit reservado

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)

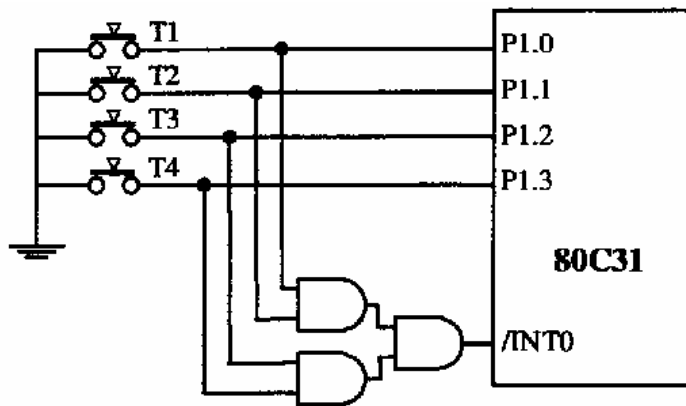
Tiempo de Respuesta

Debe esperar la interrupción si:

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.



Ejemplo 1: Conexión de teclas

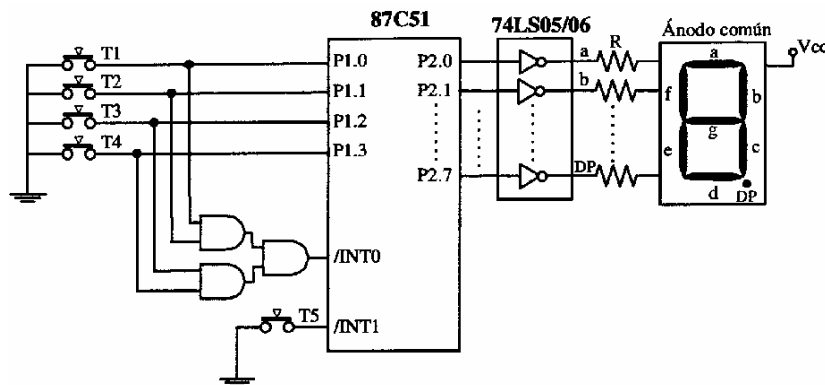


```
#include <reg51.h>
/*****
Rutina de vectorización (ejemplo 1)
*****/
//Definición de entradas
sbit Tecla_1=P1^0;
sbit Tecla_2=P1^1;
sbit Tecla_3=P1^2;
sbit Tecla_4=P1^3;
//Variables globales
unsigned char tecla;

// 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.
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) {
    }
}
```

Ejemplo 2: Conexión teclas y dígito 7 seg.



```
#include <reg51.h>
unsigned char tecla;

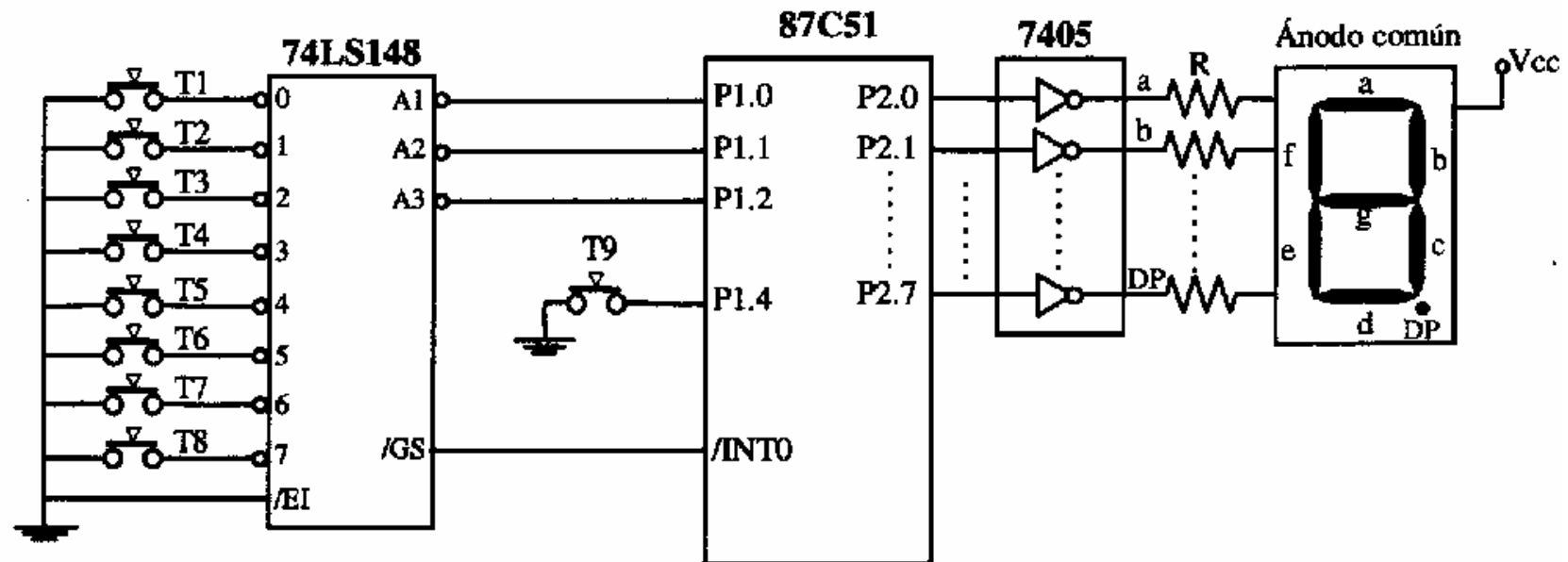
// Rutina de RSI de /INT0
void RSI_Int0(void) interrupt 0 {
    if ((P1|0xF0)==0xFE) tecla=0x01;
    else if ((P1|0xF0)==0xFD) tecla=0x02;
    else if ((P1|0xF0)==0xFB) tecla=0x03;
    else if ((P1|0xF0)==0xF7) tecla=0x04;
    // se han pulsado varias teclas
    tecla=0x05;
    IE0=0;
}

// 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.
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 dígito
    }
}
```

Ejemplo 3. Conexión teclas mediante 74LS148



```
#include <reg51.h>
/***** Rutina de vectorización (ejemplo 3) *****/
sbit Tecla_T9=P1^4;
unsigned char tecla;

//tabla que convierte de código 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
}

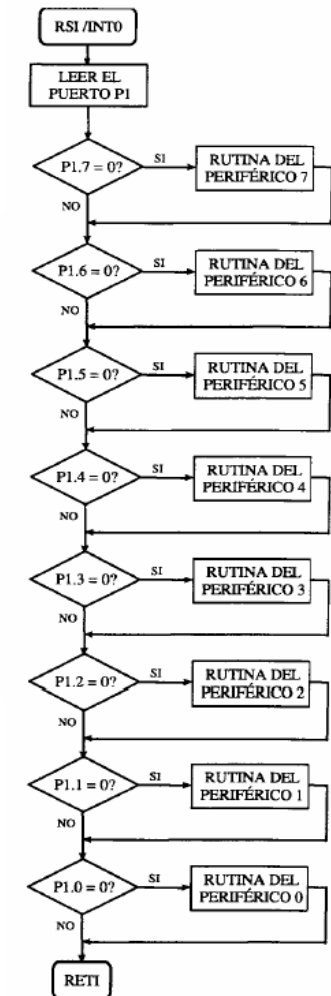
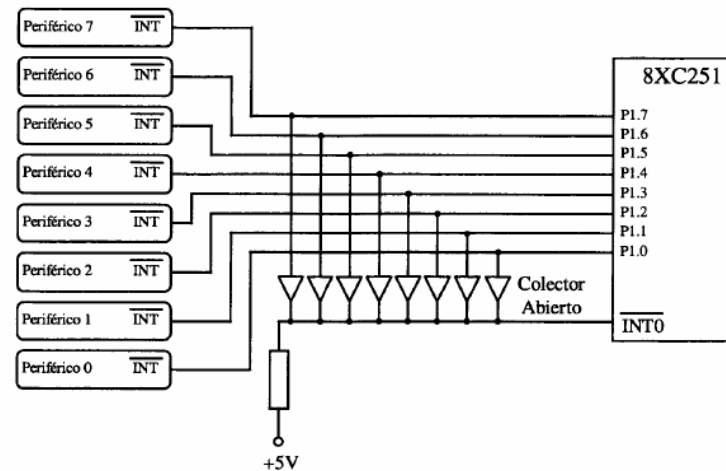
/***** Rutina de inicio (Habilitación de interrup /INT0 por nivel) *****/
void inicio(void) {
    PX0=1; //Prioridad alta para
    EX0=1; //Activa bit de habilitación de /INT0
    EA=1; //Activa el bit de habilitación general
}

/***** Programa principal *****/
void main(void) {
    inicio();
    while(1) {
        if (Tecla_T9) P2=tecla;
        else tecla=0x00;
    }
}
```

Ejemplo 4: Control de multiples fuentes

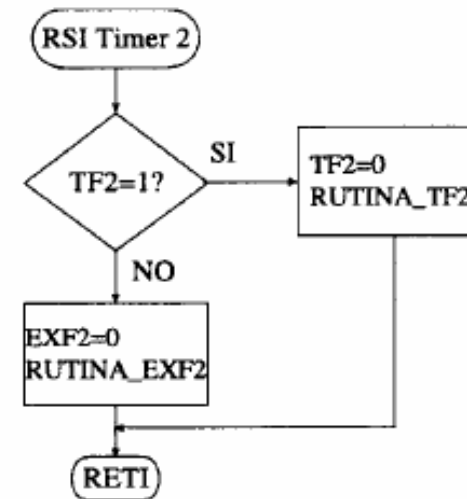
```
#include <reg51.h>
sbit PERI1=P1^0;
sbit PERI2=P1^1;
sbit PERI3=P1^2;
sbit PERI4=P1^3;
sbit PERI5=P1^4;
sbit PERI6=P1^5;
sbit PERI7=P1^6;
/*****
  RUTINA DE SERVICIO A LA INTERRUPCION INTO
  *****/
void RSI_Int0(void) interrupt 0 {
  // Si P1.7 (bit MSB) es cero salta a la
  // rutina de atención 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
}
```



Interrupción del Timer 2

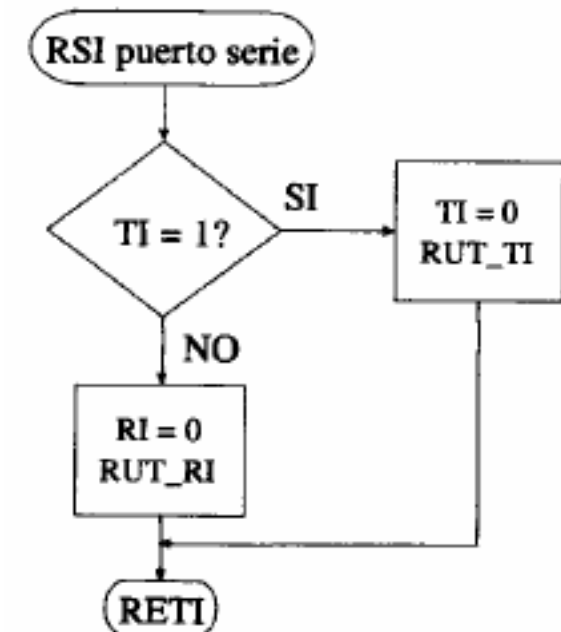
```
#include <reg52.h>
/*****
  RUTINA DE SERVICIO A LA INTERRUPCION TIMER 2
  *****/
void RSI_Timer2(void) interrupt 5 {
  //Dirección de RSI del Timer 2 es 0x2B(8*5+3)
  if (TF2) {
    TF2=0;
    // Tratamiento de TF2
  }
  else {
    EXF2=0;
    // Tratamiento EXF2
  }
}
```



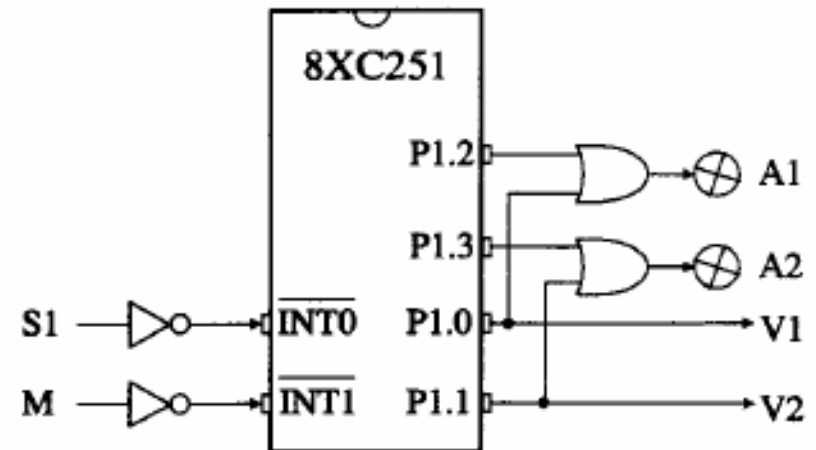
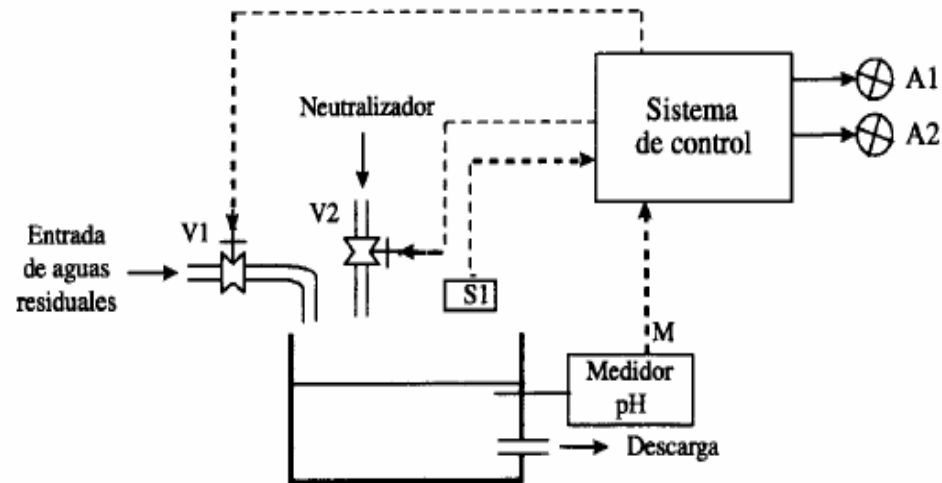
- TF2 bit de rebosamiento del timer 2, EXF2 bit que indica que se ha producido un cambio de nivel en la patilla T2EX, P1.1 del 8052/8032 siempre que este activo EXEN2.

Interrupción del puerto serie

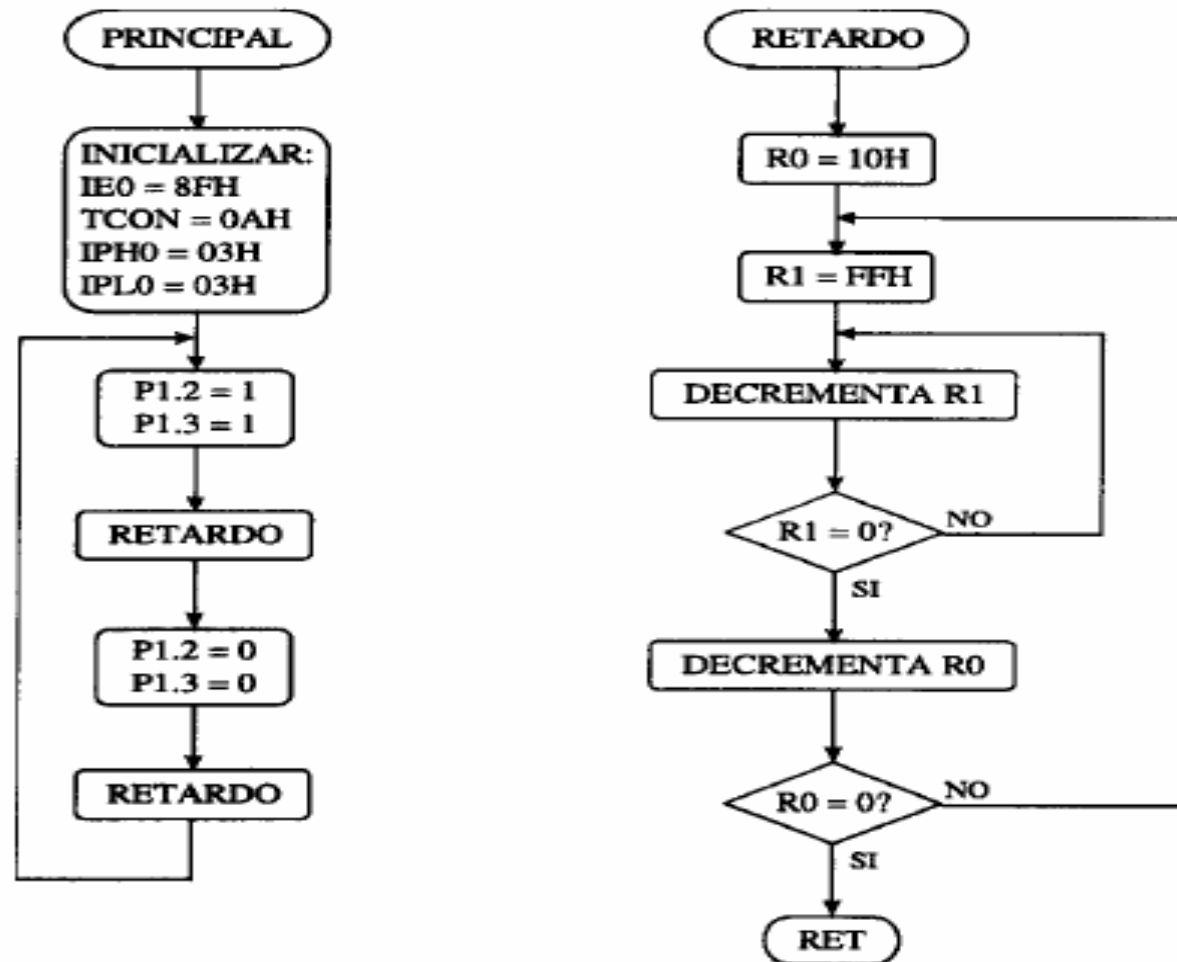
```
#include <reg51.h>
/*****
***
RUTINA DE SERVICIO A LA INTERRUPCION DEL PUERTO
SERIE
*****/
void RSI_Serie(void) interrupt 4 {
//La dirección de comienzo de la
//RSI del puerto serie es 0x23(8*4+3)
if (TI) {
    TI=0;
    // envio nuevo carácter
}
else {
    RI=0;
    // recepcion nuevo carácter
}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



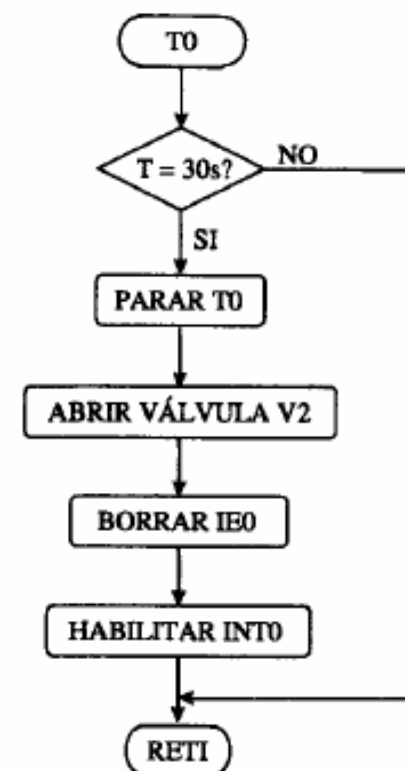
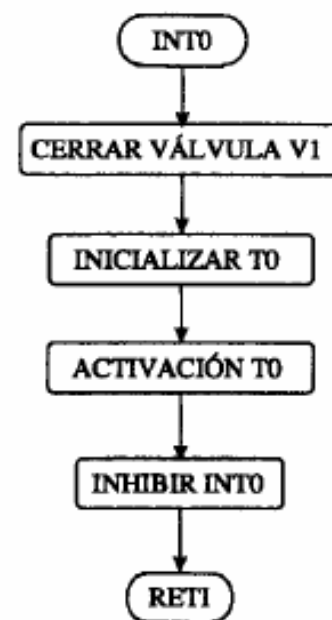
Ejemplo 5: Control ph depósito.



Control de ph

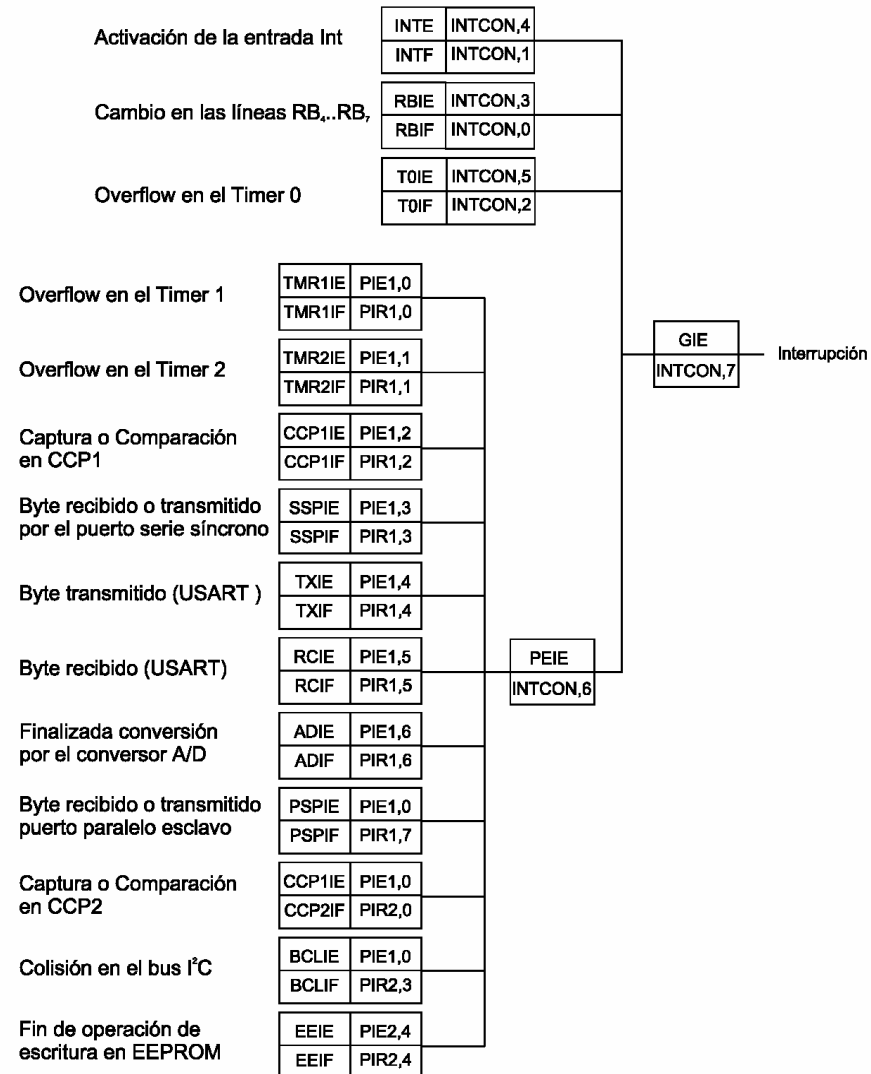


Control de ph



```
#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 rebasamientos_T0, rebasamientos_T1;
//Vector de interrupción de /INT0
void RSI_Int0(void) interrupt 0 {
    VALV_V1=0; //Se cierra la válvula V1
    TMOD|=0x01; //Se programa el Timer 0 en modo 1
    TH0=0x3C; //Se pone Timer 0 para temporizar 0.5 s
    TL0=0x0AF;
    rebasamientos_T0=0; //contador de rebasamientos del Timer 0
    TR0=1; //Puesta en marcha del Timer 0
    EX0=0; // Se inhibe la interrupción /INT0
}
//Vector de interrupción del Timer 0
void RSI_Timer0(void) interrupt 1 {
    rebasamientos_T0++; //Incrementa el contador de rebasamientos
    //Si núm menor o igual que 59 continua
    if (rebasamientos_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;
    rebasamientos_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 {
    rebasamientos_T1++; //Incrementa el contador de rebasamientos
    //Si menor o igual que 9 continua
    if (rebasamientos_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 /INT1 por nivel
    IP=0x03; //programación niveles de prioridad
    //Secuencia de parpadeo de los indicadores luminosos A1 y A2
    while(1) {
        LED_A1=LED_A2=1;
        delay(50);
        LED_A1=LED_A2=0;
        delay(50);
    }
}
```

Interrupciones PIC16



Registro INTCON

GIE	PEIE	TOIE	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

Registro PIE1 y PIE2

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 ² C	BCLIF	PIR2	3
EEIE	PIE2	4	Fin de operación de escritura en la EEPROM	EEIF	PIR2	4

Proceso de atención a interrupciones

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).

Rutina de Servicio de interrupción

```
#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();
}
```

Ejemplo 6: Conexión de teclas y dígito

```
#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)
}

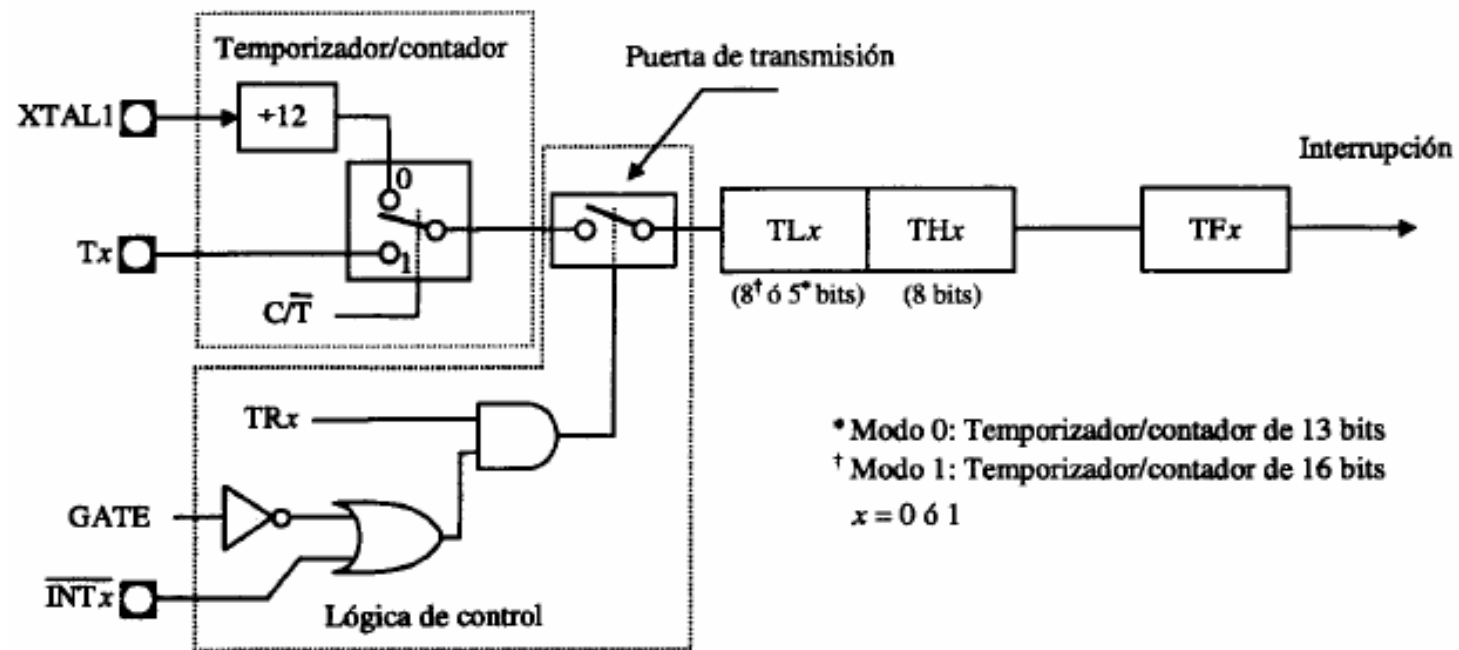
unsigned char
Siete_seg[5]={0x3F,0x06,0x5B,0x4F,0x66,0x00};

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

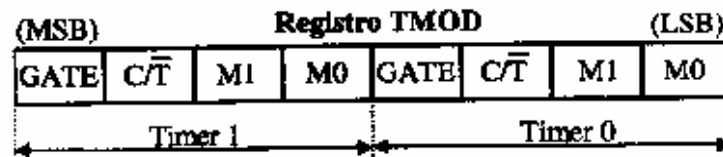
Temporizadores y Contadores

- Tres temporizadores/contadores para MCS-51.
- Tres temporizadores/contadores y un Watch Dog para PIC.

MCS-51:

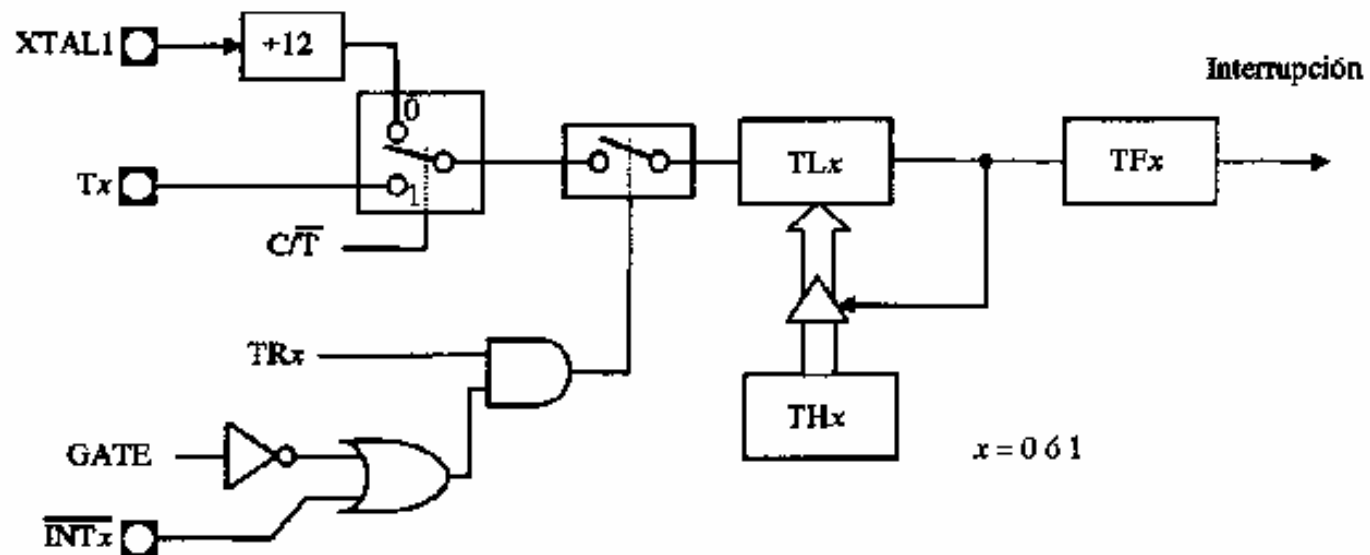


Registro TMOD

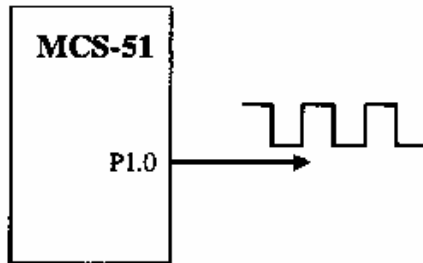


Bit	Comentario
GATE	GATE a 0 lógico hace que el Timer se gobierne mediante TRx, con TRx a 1 lógico se pone en marcha el Timers y con TRx a 0 lógico se detiene (x=0 ó 1). GATE a 1 lógico, junto con TRx a 1, hace que el Timer se gobierne por hardware, mediante el estado lógico de la entrada /INTx
C/T	Selecciona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

Temporizador en Modo 2



Ejemplo 7: Generación de una señal periódica



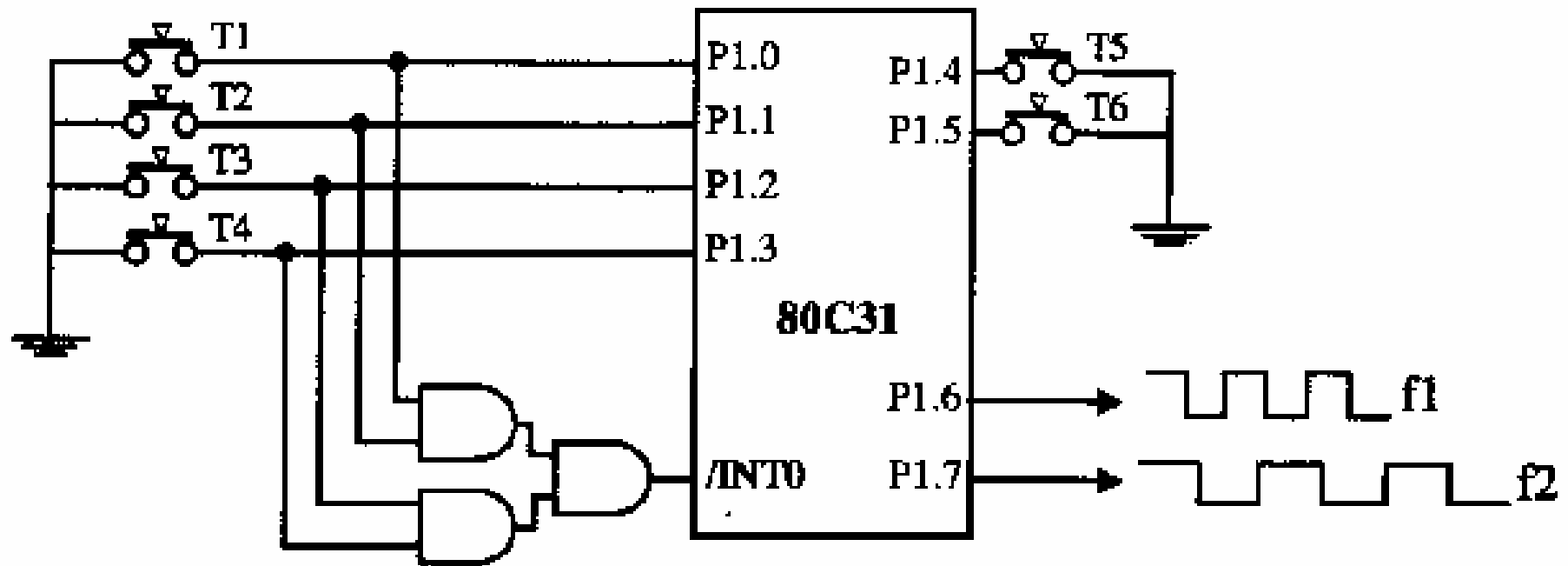
```
#include <reg51.h>
//Generación de una señal periódica
//Definición salida
sbit SALIDA=P1^0;
//Rutina de servicio del Timer0
void RSI_Timer0(void) interrupt 1 {
    SALIDA=~SALIDA;    //Complementa P1.0. Pasa 1a0, y 0a1
                       //El bit TFO se borra automáticamente RETI
}
// Rutina 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;    //M1=1 y M0=0 (Modo 2),
                  // GATE=0 y C/T=0 del Timer 0
    TL0=156;    //Pone valor 156 en TL0 (256-100)
    TH0=156;    //Pone valor de recarga en TH0 (256-100)
    TR0=1;     //Pone en marcha el contador
}
// Rutina Principal
void main(void) {

    inicio();
    while (1) {

    }

}
```

Ejemplo 8: Generador de diversas frecuencias



```

#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
}

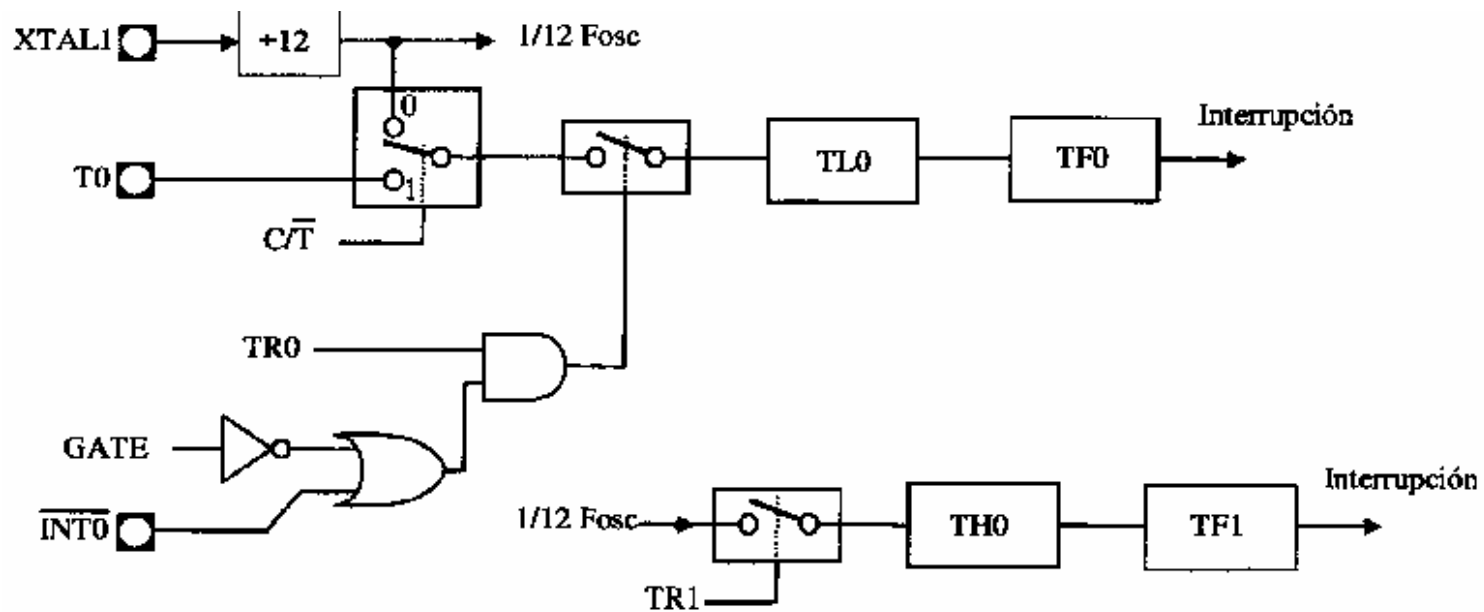
// Rutina de servicio del Timer1
void RSI_Timer1(void) interrupt 3 {
    SAL_2=~SAL_2; //Complementa P1.7
    //El bit TF1 se borra automáticamente
}

//Rutina de Inicio
void inicio(void) {
    IT0=1; //INT0 activa por flanco descendente
    PT0=1; //Asigna prioridad alta al Timer0
    P1=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
}

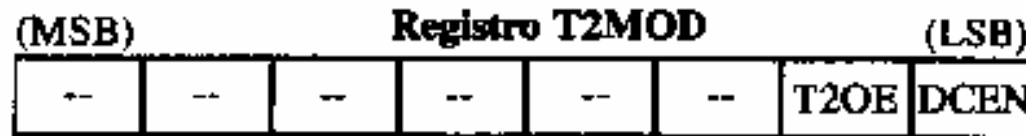
// Rutina Principal
void main(void) {
    inicio();
    while (1) {
        if (!Tecla_T5) TH1=236; //((256-20) valor de recarga para 25kHz
        else if (!Tecla_T6) TH1=246; //((256-10) valor recarga para 50kHz
        TH0=6; //por defecto 2kHz en Timer 0
        TH1=6; //por defecto 2kHz en Timer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

```


Modo 3. Varios contadores.



Control del Timer 2. T2MOD.



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

Control del Timer 2. T2CON.

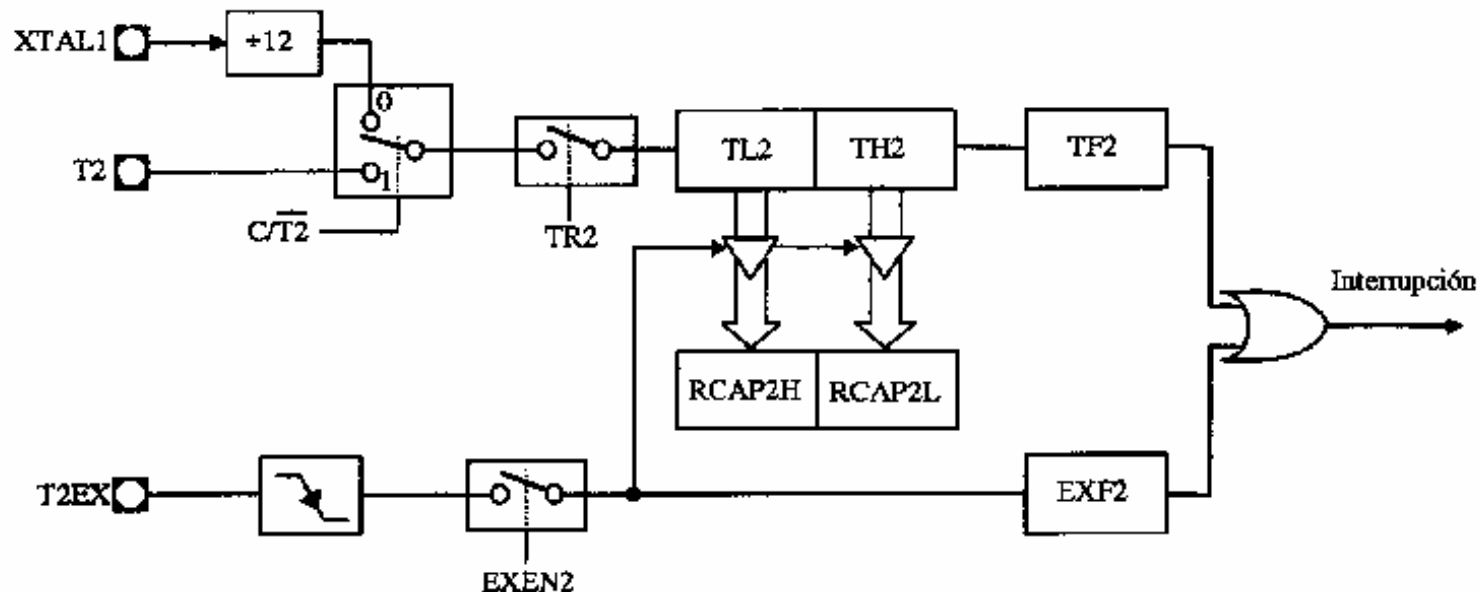


Bit	Comentario
TF2	Bit de desbordamiento. TF2=1 al producirse un desbordamiento. Este bit no se activa si RCLK=1 o TCLK=1, Debe borrarse por software.
EXF2	Bit de entrada externa. EXF2 se pone a 2 lógico al producirse un flanco descendente en el terminal T2EX, siempre y cuando EXEN2 esté habilitado
RCLK	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 específicas 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 captura al flanco negativo en T2EX, si EXEN2=1. Con CP/RL2=0 se produce recarga al flanco negativo en T2EX, si EXEN2=1. Si RCLK=1 o TCLK=1 se ignora CP/RL2 y se fuerza recarga del Timer 2 al producirse un desbordamiento en su valor

Modos de funcionamiento del Timer 2

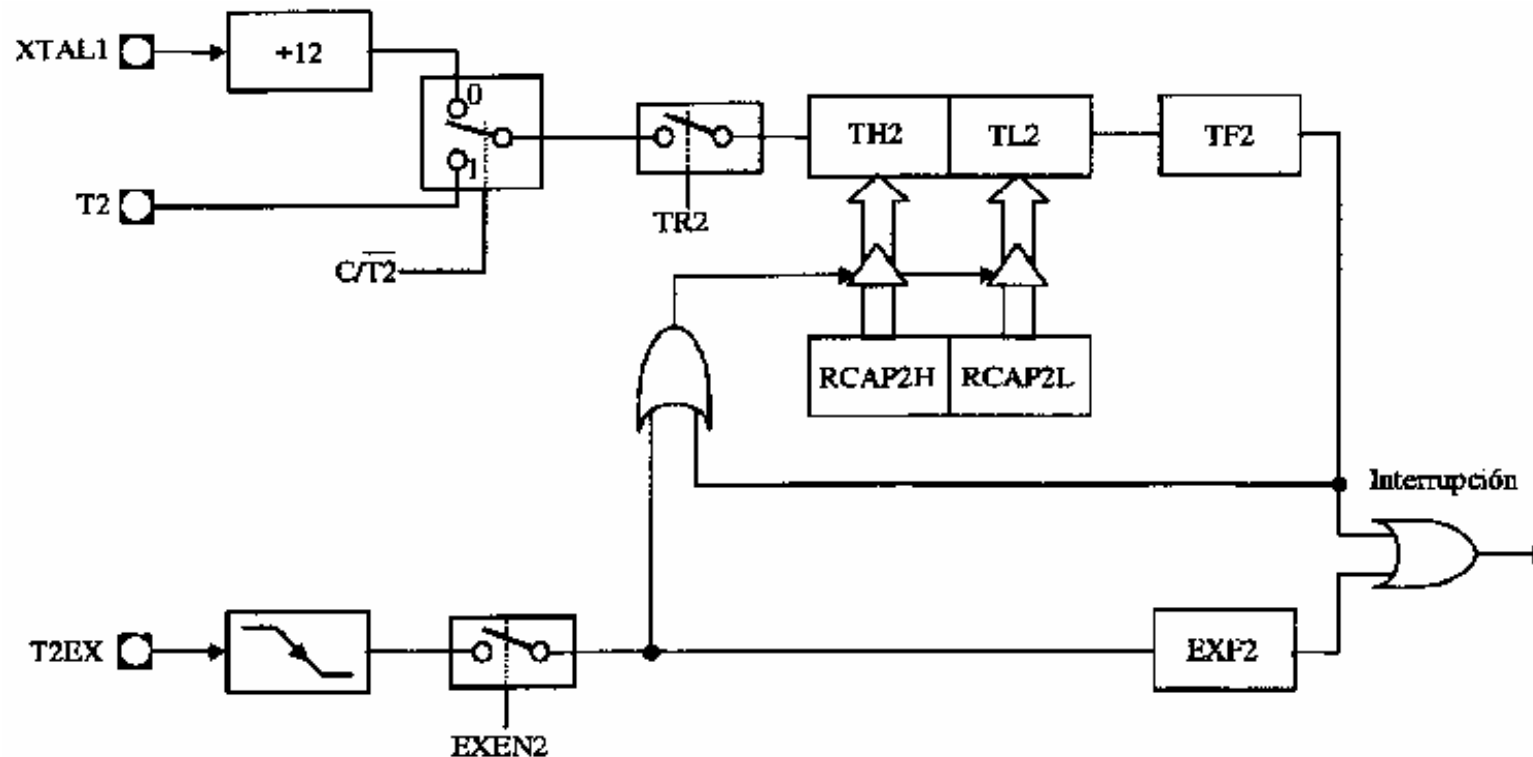
Modo	RCLK o TCLK	CP/RL2	T2OE
Modo autorrecarga	0	0	0
Modo captura	0	1	0
Modo Baud Rate	1	X	X
Modo Clock-out	X	0	1

Modo Captura.



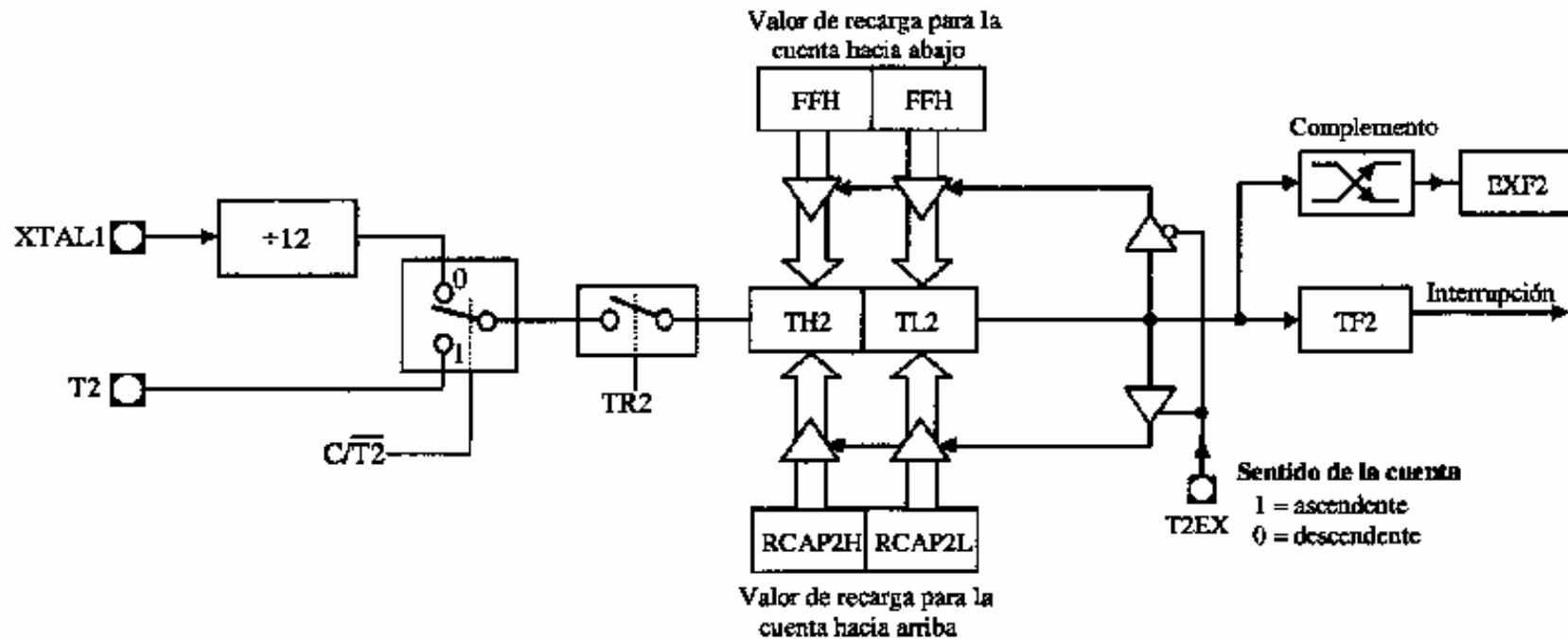
Modos de funcionamiento del Timer 2

Modo Auto-Recarga. Contador Ascendente (DCEN=0).



Modos de funcionamiento del Timer 2

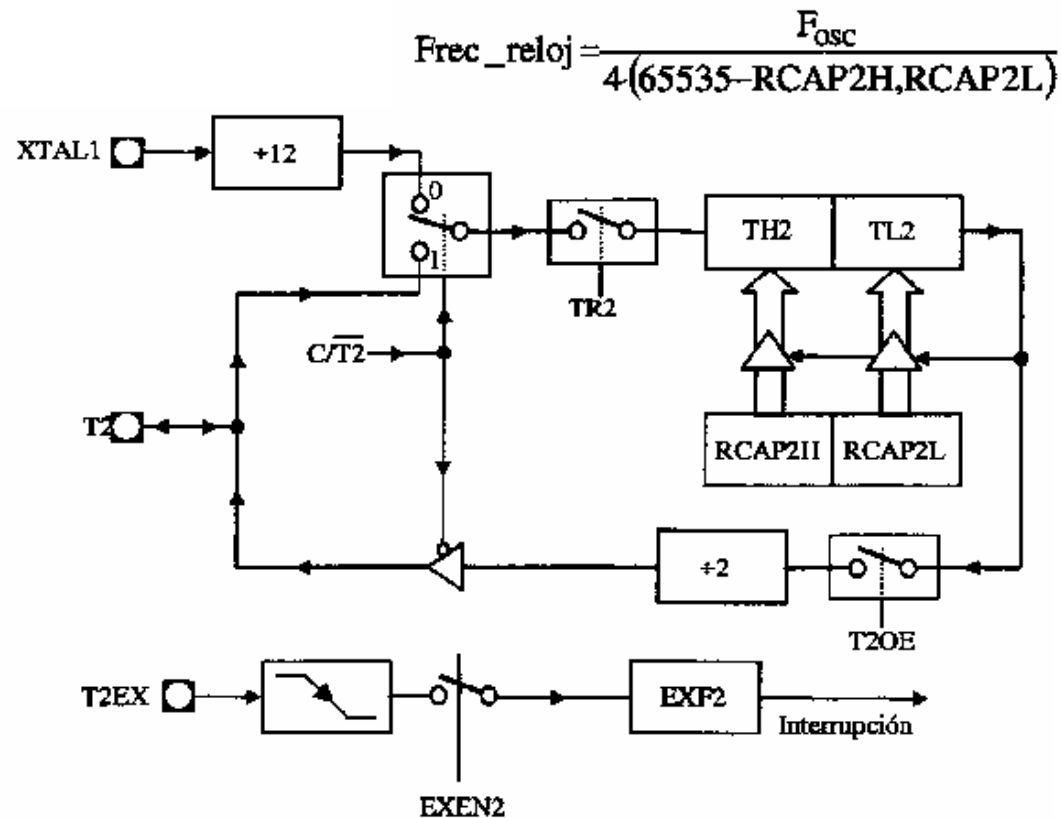
Modo Auto-Recarga. Contador Descendente (DCEN=1).



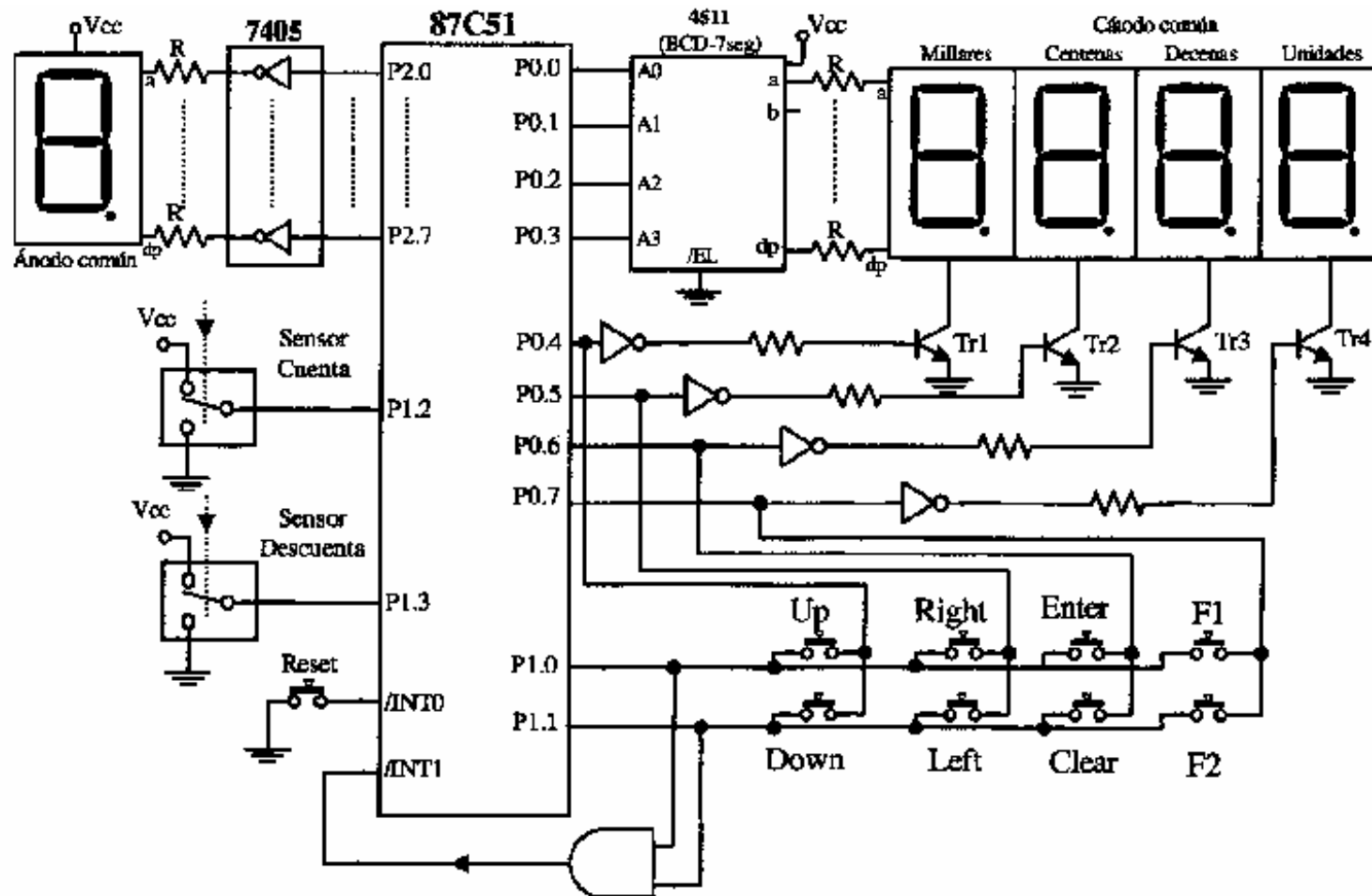
Modos de funcionamiento del Timer 2

Modo Generador de Baudios.

Modo Clock-Out.



Ejemplo 9: Refresco y teclado matricial




```

#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 dígito 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 /INT1
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
}
/*****
Rutina principal
*****/
void main(void) {
inicio();
while (1) {
if (!B_CUENTA) Cuenta();
else if (!B_DESCU) Descuenta();
P2=tecla; //Carga dígito en P2, para mostrar dígito
}
}

```

Temporizadores PIC16. Timer 0.

- Registro TMR para leer/escribir cuenta.
- Bits de control en registro INTCON. T0IE y T0IF.
- Señal externa o interna seleccionable con OPTION(5). T0CS.
- Selección flanco de subida o bajada con OPTION(4). T0SE.
- Selección entre preescaler o Watch Dog con bit OPTION(3). PSA.
- Preescaler seleccionable con PS2,PS1 y PS0

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

Temporizadores PIC16. Timer 1.

- Registro TMR1H y TMR1L para leer/escribir cuenta.
- Bits de control en registro INTCON. TMR1E y TMR1F.
- Señal externa o interna seleccionable con TMR1CS.
- Señal externa puede provenir de un oscilador con T1OSCEN.
- Bit de activación del timer TMR1ON.
- Preescaler activo con los bits T1CKPS1 y T1CKPS0.

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

Temporizadores PIC16. Timer 2.

- Registro TMR2 para leer/escribir cuenta.
- Bits de control TMR2E y TMR2F.
- Señal interna.
- Bit de activación del timer TMR2ON.
- Preescaler controlado con T2CKPS1 y T2CKPS0.

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

- Postscaler controlado con T2OUTPS3 a T2OUTPS0.

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

Ejemplo 10: Reloj por interrupciones

```
#include <pic.h>
#include "serial.h"

unsigned char segundos,minutos,horas;
unsigned int msec;
static bit bad_intr;

// El timer del pic se incrementa cada pulso de reloj/4, o sea a 4MHz
// se incrementa cada microsegundo, luego para contar
// 1000 microsegundos, se produciría la interrupción cada 256 usec.
// Lo que no nos sirve, porque cometemos errores aunque usemos
// una variable para almacenar los usec.

// Otra posibilidad es emplear el preescaler del TIMER0 para
// dividir por 4, de esta forma al alcanzar el overflow,
// se habrán producido 256*4 ticks, o sea 1024 usec que
// aproximadamente
// suponen un mseg. Esta es la forma más sencilla de hacerlo.

/* Función de atención a la interrupción*/
static void interrupt
ISR(void)
    // Aquí esta la función de interrupción
    // El nombre puede ser cualquiera.
{
    if(!T0IF)    // Hubo un overflow del timer?
        bad_intr = 1;    // NO! Es de otro tipo! la ignoramos
    else {
        msec++; //suma a la cuenta
        if (msec>=1000) {msec=0;segundos++;}
        if (segundos >= 60) { segundos=0;minutos++; }
        if (minutos >= 60) { minutos=0;horas++; }
        if (horas >= 24) horas=0;
        T0IF = 0;    // Limpia bandera de interrupción,
                    // listo para la siguiente
    }
}

void main(void) {
    /* Inicializa puerto serie */
    serial_setup();

    /* Inicializa reloj */
    msec=segundos=minutos=horas=0;

    /* Inicializa timer */
    PSA = 0;    // Preescaler asignado a Timer 0
    PS2 = 0;
    PS1 = 0;    // PS2-PS0: 001 -> división por cuatro
    PS0 = 1;    // Programamos preescaler Timer
                // (Ver data sheet en anexo)
    T0CS = 0;    // Timer incrementa con pulso de reloj
    T0IE = 1;    // Habilita interrupción en overflow TMR0
    GIE = 1;    // Habilitación global de interrupciones
    while (1) {
        putch((horas/10)+0x30); //decena de horas
        putch((horas%10)+0x30); //unidad de horas
        putch(':');
        putch((minutos/10)+0x30); //decena de minutos
        putch((minutos%10)+0x30); //unidad de minutos
        putch(':');
        putch((segundos/10)+0x30); //decena de segundos
        putch((segundos%10)+0x30); //unidad de segundos
        putch('\r');
        //vuelve cursor a inicio
    }
}
```

Ejemplos: Pulsadores y LEDs

```
include <pic.h>

#define PUL1 RB4
#define PUL2 RB5
#define PUL3 RB6
#define LED1 RA5
#define LED2 RA3
#define LED3 RA2

void main(void) {

ADCON1=0x7; //desactiva entradas analógicas
OPTION=0x7F; //Activa pull-ups
TRISA=0x13; //Salidas a LEDS

while (1) {
    LED1=PUL1;
    LED2=PUL2;
    LED3=PUL3;
}

}
```

```
#include <pic.h>
#include "teclado.h"

#define LED1 RA5
#define LED2 RA3
#define LED3 RA2

void main(void) {
    unsigned char tmp;

    ADCON1=0x7; //desactiva entradas analógicas
    OPTION=0x7F; //Activa pull-ups
    TRISA=0x13; //Salidas a LEDS
    TRISB=0xF0; //Entradas a Filas
    PORTB=0xF0; //Columnas a cero

    LED1=LED2=LED3=1; //apagados al principio

    while (1) {
        tmp=lee_teclado(); //lee ascii de teclado
        if (tmp<'8') {
            //si es código entre 0 y 7
            tmp=(tmp-0x30)&0x7; //se extraen tres bits
            LED1=! (tmp&0x04);
            LED2=! (tmp&0x02);
            LED3=! (tmp&0x01);
        }
    }
}
```

Ejemplos: Adivinar un número

```
#include <pic.h>
#include "teclado.h"
#include "main.h"
#include "delay.h"

#define LED1 RA5 //ROJO
#define LED2 RA3 //AMARILLO
#define LED3 RA2 //VERDE

void main(void) {
    unsigned char dec, uni, num, entrado;

    ADCON1=0x7; //desactiva entradas analogicas
    OPTION=0x7F; //Activa pull-ups
    TRISA=0x13; //Salidas a LEDs
    TRISB=0xF0; //Entradas a Filas
    PORTB=0xF0; //Columnas a cero
    LED1=LED2=LED3=1; //apagados al principio

    TMR0=0; //Pone a cero timer0
    TOCS=0; //Arranca la cuenta
    while (!lee_teclado()); //espera pulsacion tecla
    num=TMR0; //obtiene numero aleatorio
    DelayMs(50); //elimina rebote
    if (num>99) num=num/10; //si es mayor de 99 lo reduce

    while (1) {
        while (!(dec=lee_teclado())); //lee ascii de decena
        DelayMs(50);
        while (!(uni=lee_teclado())); //lee ascii de unidad
        DelayMs(50);
        entrado=(dec-0x30)*10+(uni-0x30);

        if (num>entrado) {
            //numero es menor, enciendo verde
            LED3=0;
            DelayS(1); //espera un segundo
            LED3=1;
        }
        else if (num<entrado) {
            //numero es mayor, enciendo amarillo
            LED2=0;
            DelayS(1);
            LED2=1;
        }
        else {
            //numero es igual, enciendo rojo
            LED1=0;
            while(1); //fin
        }
    }
}
```


Ejemplos: Control de Acceso (I)

```
#include <pic.h>
#include "teclado.h"
#include "main.h"
#include "delay.h"
#include "lcd.h"

#define LED1 RA5 //ROJO

bit abierta; //estado de la puerta
unsigned const char clave[]="Entra clave:";
unsigned const char incorr[]="Clave incorrecta.";
unsigned const char corr[]="Clave correcta.";

void main(void) {
    unsigned char i,tmp;

    ADCON1=0x7; //desactiva entradas analogicas
    OPTION=0x7F; //Activa pull-ups
    TRISA=0x13; //Salidas a LEDs
    TRISB=0xF0; //Entradas a Filas
    PORTB=0xF0; //Estado teclado
    TRISC=0xC0; //lineas del LCD
    LED1=1; //inicia la puerta a cerrada
    abierta=0;
    lcd_init();
    DelayMs(150);
    lcd_clear();
    if (EEPROM_READ(0)!='C') { //micro nuevo
        lcd_puts(clave);
        for (i=0;i<4;i++) {
            //espera a pulsar tecla
            while(!(tmp=lee_teclado()));
            //espera a soltar tecla
            while(lee_teclado());
            lcd_putch(tmp);
            EEPROM_WRITE(i+1,tmp);
        }
        EEPROM_WRITE(0,'C');
    }
    while (1) {
        lcd_clear();
        lcd_puts("(A) Apertura\0");
        lcd_goto(0x40);
        lcd_puts("(C) Cambio clave");
        //espera a pulsar tecla
        while(!(tmp=lee_teclado()));
        //espera a soltar tecla
        while(lee_teclado());
        DelayMs(150);
        if (tmp=='A') pide_clave();
        if (tmp=='C') cambia_clave();
    }
}
```

Ejemplos: Control de Acceso (II)

```
void pide_clave(void) {
    unsigned char error,i,tmp;

    lcd_clear();
    lcd_puts(clave);
    for (i=0;i<4;i++) {
        //espera a pulsar tecla
        while(!(tmp=lee_teclado()));
        //espera a soltar tecla
        while(lee_teclado());
        lcd_putch(tmp);
        if (EEPROM_READ(i+1)!=tmp) error=1;
    }
    if (error) {
        lcd_goto(0x40);
        lcd_puts(incorr);
        DelayS(1);
    }
    else {
        if (!abierta) {
            LED1=0; //Abre la puerta
            abierta=1;
        }
        else {
            LED1=1; //Abre la puerta
            abierta=0;
        }
        lcd_goto(0x40);
        lcd_puts(corr);
        DelayS(1);
    }
}
```

```
void cambia_clave(void) {
    unsigned char error,tmp,i;

    lcd_clear();
    lcd_puts(clave);
    for (i=0;i<4;i++) {
        //espera a pulsar tecla
        while(!(tmp=lee_teclado()));
        //espera a soltar tecla
        while(lee_teclado());
        lcd_putch(tmp);
        if (EEPROM_READ(i+1)!=tmp) error=1;
    }
    if (error) {
        lcd_goto(0x40);
        lcd_puts(incorr);
        DelayS(1);
    }
    else {
        lcd_goto(0x40);
        lcd_puts(corr);
        DelayS(1);
        lcd_clear();
        lcd_puts(clave);
        for (i=0;i<4;i++) {
            //espera a pulsar tecla
            while(!(tmp=lee_teclado()));
            //espera a soltar tecla
            while(lee_teclado());
            lcd_putch(tmp);
            // escribe el caracter en memoria
            EEPROM_WRITE(i+1,tmp);
        }
    }
}
```

Ejemplos: Frecuencimetro

```
#include <pic.h>
#include "teclado.h"
#include "main.h"
#include "delay.h"
#include "lcd.h"

#define HI(x)    ((x)>>8)
#define LO(x)    ((x)&0x00FF)

unsigned char eventos,frecuencia;

void interrupt isr(void) {
if (TMR1IF) {
    TMR1ON=0; //Timer1 apagado
    eventos++;
    if (eventos>3) {
        frecuencia=TMR0;
        eventos=0;
        TMR0=0;
    }
    TMR1H=HI(65535-50000); //Timer 1 50ms=50000us
    TMR1L=LO(65535-50000) ;
    TMR1ON=1; //Timer1 habilitado
    TMR1IF=0; //Limpio bandera int.
}
}
```

```
void main(void) {

ADCON1=0x7; //desactiva entradas analogicas
TRISA=0x13; //Salidas a LEDS
TRISC=0xC0; //lineas del LCD

T1CKPS0=0; //Preescaler 1:1
T1CKPS1=0;
T1OSCEN=0; //Oscilador deshabilitado
TMR1CS=0; //Temporizador interno(OSC/4)
//El TMR1 se programa para interrumpir cada 50ms
TMR1H=HI(65535-50000); //Timer 1 50ms=50000us
TMR1L=LO(65535-50000) ;
TMR1ON=1; //Timer1 habilitado
TMR1IE=1; //Habilito int. TMR1
PEIE=1; //Habilito int. perifericos
GIE=1; //Habilito general

T0CS=1; // Cuenta eventos externos
T0SE=1; // Cuenta en flanco de bajada
PSA=1; // Preescaler desactivado
frecuencia=0;

lcd_init();
lcd_clear();
lcd_puts("Frec: xx Hz");
while(1) {
    lcd_goto(6);
    lcd_putch(((frecuencia<<2)/10)+0x30);
    lcd_putch(((frecuencia<<2)%10)+0x30);
    while(eventos); //refresca cada 250ms
}
}
```