Práctica 4

Interrupciones y temporización.

1. Objetivos

Uno de los objetivos de esta práctica es que el alumno aprenda el funcionamiento de las interrupciones y su manejo, y que profundice en las técnicas básicas de temporización de los microcontroladores. Como elemento periférico se utilizará el temporizador 1 del microcontrolador, además de los elementos vistos ya en la sesión anterior, como el LCD.

Todos estos conceptos se pondrán en práctica realizando un sencillo temporizador de horas minutos y segundos.

2. Introducción

En esta práctica se utilizarán los temporizadores del PIC y el LCD y se aprenderá a utilizar las interrupciones y su programación en lenguaje C. En las siguientes secciones se encuentran explicados todos los elementos necesarios para realizar con éxito esta sesión.

Interrupciones en el compilador C18

La directiva *#pragma interruplow nombre* y *#pragma interrup nombre* definen rutinas de servicio de interrupción (ISR). La primera de baja primoridad *interruplow* y la segunda de alta prioridad *#pragma interrup*. Ejemplo:

```
void foo(void);
...

#pragma interrupt foo

void foo(void)
{
/* interrupción */
}
```

El C18 no sitúa automáticamente las ISR en las posiciones de los vectores de interrupción. Hay que definir una sección de código que se sitúe en las posiciones definidas para los vectores de interrupción y dentro de dicha sección introducir un salto a la ISR correspondiente. Normalmente se pone un goto para llevar el control desde los vectores de interrupción hasta las subrutinas ISR. En el siguiente ejemplo se declara una subrutina de interrupción (denominada `high_isr)` asociada a la interrupción de alta prioridad en la posición 0x08, perteneciente al vector de interrupción de alta prioridad:

```
#include <p18cxxx.h>
void high_isr(void);

#pragma code high_vector=0x08
void interrupt_at_high_vector(void)
{
_asm GOTO high_isr _endasm
}

#pragma code                    /* vuelve a la sección de código por defecto */
#pragma interrupt high_isr
void high_isr (void)
{
/* ... */
}
```

2.1 El temporizador del PIC

El PIC18F2550 tiene cuatro temporizadores (timer0, timer1, timer2 y timer3) que se pueden configurar de muy diferentes maneras. Para la sesión de hoy se va a utilizar el timer1, que se trata de un contador/temporizador de 16 bits, que almacena en los registros *TMR1L*:*TMR1H* el valor de cuenta. *TMR1L* contiene la parte baja y *TMR1H* la parte alta de los 16 bits.

Como vamos a utilizar el reloj interno del PIC, tendremos que poner el timer1 en modo temporizador (*timer*) en vez del modo contador (*counter*). También hay que encenderlo, habilitar el oscilador y seleccionar el factor de escala de predivisor. Todo esto se programa con el registro *T1CON*. Por ejemplo haciendo *T1CON=00000001b* encendemos el timer1, el factor de escala es uno y se utiliza el reloj interno (los impulsos que incrementan el valor de cuenta proceden de la señal de reloj dividida entre 4).

REGISTRO: **T1CON: TIMER1 CONTROL REGISTER**

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **RD16:** 16-Bit bit de habilitación del modo de lectura escritura de 16 bits
1 =habilita el modo de lectura /escritura de 16 bits en una operación
0 =las lecturas y escrituras se realizan como dos operaciones de 8-bit

bit 6 **T1RUN:** Bit de estado del oscilador del Timer1
1 =El reloj interno del micro se deriva del oscilador interno del Timer1
0 =El reloj interno del micro se deriva de otra fuente

bit 5-4 **T1CKPS1:T1CKPS0:** Valor del predivisor
11 = 1:8
10 = 1:4
01 = 1:2
00 = 1:1

bit 3 **T1OSCEN:** Habilitación del oscilador interno del Timer1
1 =El oscilador interno está habilitado
0 =El oscilador interno está deshabilitado

bit 2 **$\overline{\text{T1SYNC}}$:** Sincronización del reloj externo del Timer1
Si TMR1CS = 1:
1 = No sincroniza el reloj externo con el interno
0 = Sinchronize el reloj externo con el interno
When TMR1CS = 0:
El bit se ignora. El Timer1 usa el reloj interno si TMR1CS = 0.

bit 1 **TMR1CS:** Bit de selección de la fuente de reloj para el Timer1
1 =Reloj externo entrada por el pin RC0/T1OSO/T13CKI (en los flancos de subida)
0 =reloj interno (Fosc/4)

bit 0 **TMR1ON:** Bit de habilitación del Timer1
1 =Habilita el funcionamiento del Timer1
0 =Para el Timer1

Siempre se cuenta hacia arriba de manera que cuando se pasa de 0xFFFF a 0x0000 se produce un desbordamiento que activa el bit 0 (denominado *TMR1IF*) del registro *PIR1*, lo que indica este desbordamiento (después del desbordamiento el bit *TMR1IF* se debe poner a cero por software). Si además se tiene habilitada la interrupción del timer1 mediante la activación (puesta a '1') del bit 0 (denominado *TMR1IE*) del registro *PIE1* y se tienen habilitadas las interrupciones de los periféricos y la general mediante la activación de los bits 6 y 7 (*PEIE* y *GIE*) del registro *INTCON*.

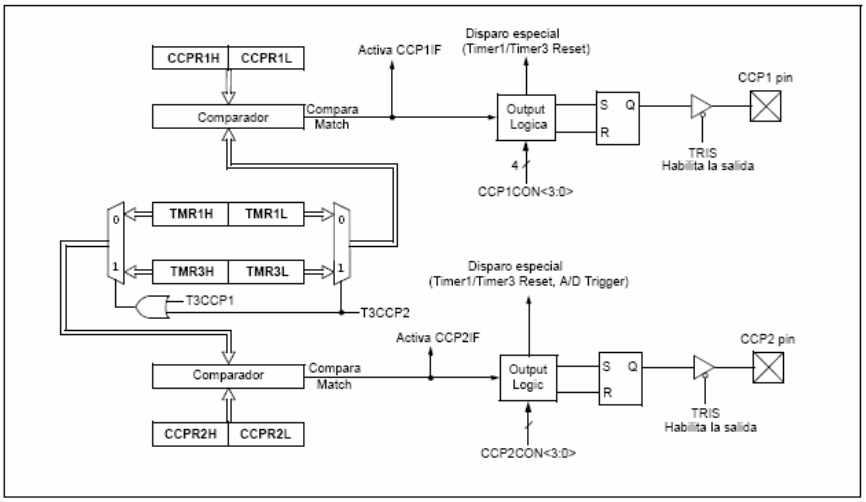
Utilizar las interrupciones sin niveles de prioridad, ya que no es necesario distinguir entre varias fuentes de interrupción. Para ello el bit 7 (*IPEN*) del registro *RCON* debe ser '0'. En este caso todas las interrupciones tienen la misma prioridad y todas utilizan la dirección del vector de interrupción 00008h.

La rutina de interrupción que trate la interrupción del timer1 debe comprobar que efectivamente la interrupción ha sido provocada por este timer1 mirando el bit *TMR1IF* indicado anteriormente. Este bit debe de ponerse a cero en la propia rutina de interrupción, por software para indicar que ya ha sido tratada. En esta rutina deberán también inicializarse los valores de *TMR1L* y *TMR1H*.

2.2 Módulo CCP (Captura, Comparación y PWM) en modo comparación

En el modo de comparación, el módulo CCP1 compara el valor de sus registros CCPR1H:CCPR1L de forma continuada con el valor de cuenta del TMR1 o TMR3. Cuando el valor del registro CCPR1H:CCPR1L coincide con el del temporizador se produce un evento. El tipo de evento que se produce se configura en el registro de control CCP1CON.

FIGURA: COMPARACIÓN



En esta práctica se utilizará el denominado modo de comparación con disparo especial, mediante el cual el módulo CCP1 cuando se produce el evento pone a 0 de nuevo el Timer y activa el bit CCP1IF que solicita una interrupción. La interrupción finalmente se produce si está habilitada mediante el bit CCP1IE=1.

El timer seleccionado, ya sea el TMR1 o el TMR3, en este modo debe estar configurado como temporizador síncrono, nunca en modo asíncrono.

REGISTRO: CCPxCON: CCP CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—(1)	—(1)	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **No implementado**: Se lee como '0'

bit 5-4 **DCxB1:DCxB0**: Bits del ciclo activo de la señal PWM.
Capture mode:
Unused.
Compare mode:
Unused.
PWM mode:
Estos bits son los LSbs (bit 1 y bit 0) de la señal 10-bit PWM para e ciclo activo. Los 8 bits MSBs (DCx9:DCx2) están en el registro CCPRxL.

bit 3-0 **CCPxM3:CCPxM0**: Modo de funcionamiento del módulo CCP
0000 = Captura/Comparación/PWM deshabilitado (resets el módulo CCP)
0001 = Reservado
0010 = Comparación, invierte la salida (CCPIF bit se activa)
0011 = Reserved
0100 = Captura, en flanco de bajada
0101 = Captura, en flanco de subida
0110 = Captura, en el 4th flanco de subida
0111 = Captura,en el 16th flanco de subida
1000 = Comparación: inicializa el pin CCP a cero; cuando se da la igualdad, activa el pin CCP (CCPIF se activa la inerrupción)
1001 = Comparación: inicializa el pin CCP a uno; cuando se da la igualdad, desactiva el pin CCP (CCPIF se activa la inerrupción)
1010 =Comparación: genera interrupción por software cuando se da la igualdad (CCPIF se activa,)
1011 = Comparación: evento especial, reset del timer y comienza una conversión A/D si es módulo es el CCP2 (CCPIF se activa la inerrupción)
11xx = PWM

Para generar las interrupciones de cuenta, en la práctica se va a utilizar el módulo CCP1 combinado con el timer1. El temporizador timer1 se debe configurar como temporizador síncrono de 16 bits pero no se debe

activar su interrupción de desbordamiento, simplemente se configura y se pone a correr. Por su parte el módulo de CCP1 se configura en modo comparación con evento especial, (lo bits del registro CCP1CON serán CCP1M3:CCp1M0:1011). Además hay que calcular el valor de fin de cuenta que debemos cargar en CCPR1H:CCPR1L para que cuando el timer1 llegue hasta este valor haya pasado un periodo de tiempo de 50ms. Cuando se produce la igualdad se dispara el evento especial que resetea el valor del timer1 y solicita la interrupción asociada al módulo CCP1 mediante el bit CCP1IF. Para que se produzca la interrupción debe estar habilitada mediante la activación del bit CCP1IE=1.

3. Trabajos a realizar

En todas las sesiones de laboratorio se pide la realización de unos programas o tareas. **Es imprescindible que se tengan preparados, o al menos esbozados, los programas al principio de cada sesión**. Al inicio de la sesión el profesor comprueba de que efectivamente los programas han sido preparados y se pone la nota correspondiente. Tener los programas preparados al inicio de la sesión es muy importante, de lo contrario, es muy difícil que en una misma sesión se puedan terminar los programas habiendo probado su funcionamiento.

Programa: Cronómetro de cuenta de minutos, segundos y décimas

Se pide realizar un programa que cuente los minutos, los segundos y las décimas de segundo que lleva encendido el microcontrolador mediante interrupciones generadas por el módulo CCP1 en modo comparación con evento especial. El timer asociado será el temporizador 1 configurado como temporizador síncrono de 16 bits. El módulo CCP1 se debe programar para contar periodos de 50 milisegundos. Hay que calcular el valor de final de cuenta (X) necesario para que cuando el timer1 cuente desde 0x0000->X haya transcurrido un periodo de 50 ms. (Recordar que el cristal utilizado el las placas de prácticas tiene una frecuencia de 4MHz).

Las interrupciones generadas por el módulo CCP1 servirán para llevar la cuenta del tiempo transcurrido en 3 variables que almacenarán los valores de los minutos, segundos y décimas. Estas variables se incrementarán dentro de la subrutina de procesamiento de las interrupciones generadas por el módulo CCP1. La cuenta se iniciará a cero y deberá mostrarse en todo momento en formato decimal por el LCD mediante 6 dígitos situados en las primeras posiciones de la primera línea con formato: “mm:ss:dd”. El funcionamiento del LCD se explicó en la práctica anterior y las funciones a utilizar para su manejo son las mismas que se vieron en dicha práctica.

Es importante tener en cuenta que el valor de cuenta se debe mostrar en el LCD como 6 carácteres ASCII. Es por tanto necesario mostrar en el LCD el valor de cada variable como dos dígitos decimales con formato ASCII. Para ver siempre los valores sobre las mismas posiciones del LCD se puede situar el cursor al principio del LCD siempre que se vaya a rescribir el valor de cuenta.

Es importante que los conmutadores de la placa EduMIC que sirven para elegir Analógico o Digital estén en su posición central (LIBRE) de esta manera los periféricos no interfieren con la operación sobre los bits RA0, RA1 y RA2 que son los que utiliza el LCD.

Recordar que antes de programar el microcontrolador con el software de programación EduMIC, la palabra de configuración config1 hay que modificarla para que su valor sea **config1= ‘0C00’.**

Apartado opcional: Cronómetro de cuenta atrás.

Se propone realizar una modificación en el programa anterior para implementar un cronómetro de cuenta atrás que sirva como alarma para saber cuando han pasado una serie de minutos. El cronómetro se inicializará después de un reset mediante las funciones para la lectura por teclado vistas en la práctica anterior, introduciendo el número de minutos que debe contar el sistema. Después de inicializarse el cronómetro sigue la secuencia de cuenta atrás y muestra su valor por el LCD con formato:”mm:ss:dd” hasta llegar a cero, en ese instante muestra por el LCD un mensaje de Alarma.