

Práctica 5. Interrupciones

(Duración: 1 sesión)

Introducción

En esta práctica nos familiarizaremos con las interrupciones del microcontrolador.

Objetivos

Al concluir la práctica, el alumno deberá ser capaz de:


- Configurar los periféricos para generar interrupciones al microcontrolador.
- Escribir rutinas de tratamiento de interrupciones.
- Saber trabajar con variables compartidas entre interrupciones y programa principal evitando problemas de incoherencia de datos.

Trabajo previo

Antes de ir al laboratorio ha de:

- Leer detenidamente la práctica, anotando las dudas que le surjan para preguntárselas al profesor.
- Escribir los programas solicitados en las secciones 2, 3 y 4.

1. Crear un proyecto

En primer lugar ha de crear un proyecto para esta práctica, siguiendo los pasos indicados en el apartado 2 de la primera práctica. No olvide trabajar en la carpeta  D: ►Micros►Grupo_XX, creada en la práctica 1.

Al igual que en la práctica 3, es necesario configurar la tarjeta para que funcione con el oscilador de cuarzo externo. Como recordará basta con incluir en el proyecto el archivo `Pic32Ini.c` disponible en Moodle.

Por otro lado asegúrese de que los *jumpers* JP1 y JP3 están conectados, ya que en caso contrario no funcionarán ni el pulsador ni los LEDs.

2. Temporizador con interrupciones

Una de las ventajas de las interrupciones es que es fácil realizar dos tareas a la vez. Partiremos del programa que enciende el LED RC0 cuando se pulsa el pulsador RB5 realizado en la práctica 2, que, por si no se lo sabe ya de memoria de tanto usarlo, se incluye a continuación para mayor comodidad:

```
1  #include <xc.h>
2  #define PIN_PULSADOR 5
3
4  int main(void)
5  {
6      int pulsador;
7
8      TRISC = XX; // A completar por el alumno (trabajo previo Práctica 2)
9      LATC  = XX; // A completar por el alumno (trabajo previo Práctica 2)
10     TRISB = XX; // A completar por el alumno (trabajo previo Práctica 2)
11
12     while(1){
13         // Se lee el estado del pulsador
14         pulsador = (PORTB>>PIN_PULSADOR) & 1;
15         if(pulsador == 0){
16             LATC &= ~1;
17         }else{
18             LATC |= 1;
19         }
20     }
21 }
```

Una vez compilado el programa para ver que funciona correctamente, se modificará para que el LED RC3 se encienda y se apague con un periodo de 2 s, en paralelo con el funcionamiento del pulsador y el LED RC0. Para ello ha de configurar el temporizador 2 para que genere interrupciones cada segundo. La prioridad de las interrupciones la pondremos a 2 y la subprioridad a 0. Una vez configurado el temporizador, se escribirá la rutina de atención a la interrupción, la cual se encargará de encender y apagar el LED. El prototipo de dicha rutina será el siguiente:

```
void __attribute__((vector(_TIMER_2_VECTOR), interrupt(IPL2SOFT), nomips16))
    InterruptionTimer2(void)
```

Tenga en cuenta que LATC es una variable compartida entre el programa principal y la interrupción, por lo que tendrá que tomar las medidas necesarias para evitar problemas de incoherencia de datos.

2.1. Ejercicio

Elimine momentáneamente las instrucciones que haya incluido para evitar la incoherencia de datos y vea si el programa presenta algún funcionamiento erróneo. Indique en la memoria de la práctica lo que haya observado.

3. Interrupciones múltiples

Modifique el programa anterior para que el LED RC2 parpadee con un periodo de 1 s, manteniendo el LED RC3 parpadeando con un periodo de 2 s. Para ello se configurará el temporizador 3 para que genere interrupciones cada 500 ms y se le dará prioridad 4 y subprioridad 0 a su interrupción.

Tenga en cuenta que ahora funcionan las dos interrupciones a la vez, por lo que pueden aparecer problemas de incoherencia de datos en el manejo del LATC.

4. Juego de velocidad

Para terminar la práctica se va a realizar un programa que permita averiguar lo rápido que es un jugador pulsando repetidamente un pulsador. Para ello se partirá del programa realizado en la sección 2. Se creará una variable global compartida entre el programa principal y la interrupción. La interrupción la pondrá a cero cada vez que se ejecute y el programa principal la incrementará cada vez que se pulse el pulsador (necesitará detectar el flanco). Además, si se detecta que la variable supera cinco pulsaciones,¹ se encenderá el LED RC1. Dicho LED permanecerá encendido hasta que se dé un reset al microcontrolador (opcionalmente puede apagarlo transcurridos 4 segundos para volver a empezar el juego).

¹Para poder cambiar fácilmente el número de pulsaciones para ganar el juego, es recomendable usar un `#define` para ello.