PRÁCTICA DE PICs - INTERRUPCIONES CON BOTÓN

NIVEL: 1
DURACIÓN: 2 horas
OBJETIVO: Los estudiantes aprenderán a cambiar el estado del LED en función de las interrupciones de un botón.

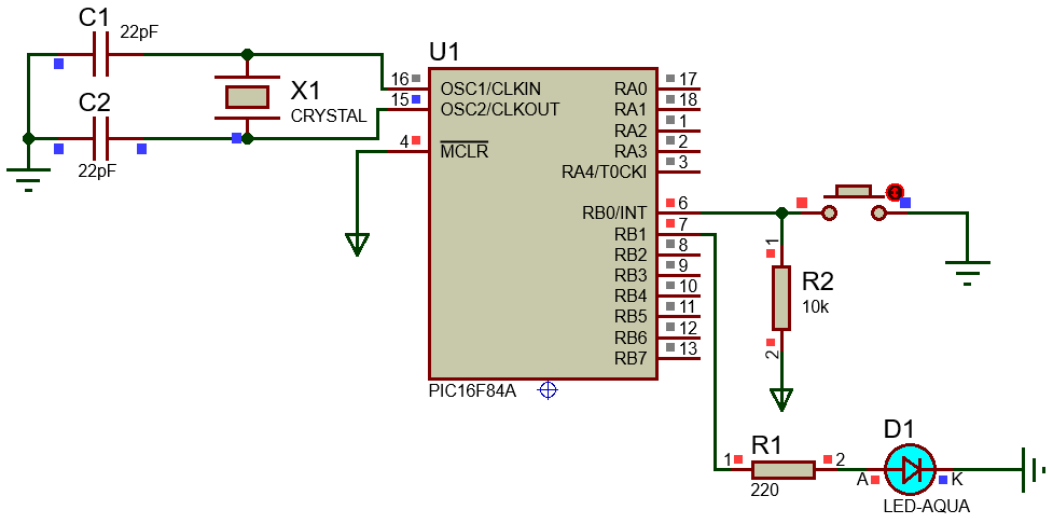
MATERIALES

- Un microcontrolador PIC (como el PIC16F877A)
- LED
- Resistor para el LED (220Ω suele ser suficiente)
- Botón (push-button)
- Resistor pull-down para el botón (10kΩ)
- Cristal de reloj 4MHz
- 2 Capacitores 22 pF
- Fuente de alimentación
- Cables y protoboard
- Proteus
- Mplab X IDE

1. DESARROLLO DE LA PRÁCTICA

1.1 SIMULACIÓN EN PROTEUS

1.1.1 MONTAJE



2.2 PROGRAMACIÓN EN MPLAB

```
#include <xc.h>

// Configuración de fusibles y frecuencia del oscilador
#pragma config FOSC = XT        // Oscilador XT (utiliza cristal externo)
#pragma config WDTE = OFF       // Watchdog Timer apagado
#pragma config PWRTE = ON       // Power-up Timer habilitado
#pragma config CP = OFF         // Protección de código apagada
#define _XTAL_FREQ 4000000      // Frecuencia del oscilador a 4 MHz

// Variables globales
volatile int ledState = 0;// Estado del LED

void __interrupt() isr() {
    if (INTF) { // Verifica si la interrupción fue generada por RB0
        ledState = !ledState;// Cambia el estado del LED
        INTF = 0;// Limpia la bandera de interrupción externa
    }
}

void main(void) {
    // Configuración de puertos
    TRISB0 = 1;// RB0 como entrada (botón)
    TRISB1 = 0;// RB1 como salida (LED)

    // Configuración de interrupciones
    INTE = 1;// Habilita interrupción externa en RB0
    GIE = 1; // Habilita interrupciones globales
    OPTION_REGbits.INTEDG = 1;// Interrupción en flanco ascendente

    while (1) {
        RB1 = ledState;// Actualiza el estado del LED
    }
}
```

Figura 1. Source Files (main)

Explicación

- FOSC: Configura el oscilador del PIC para usar un cristal externo (XT) a 4 MHz, adecuado para el PIC16F84A.
- WDTE: Deshabilita el Watchdog Timer para evitar reinicios automáticos no deseados.
- PWRTE: Habilita el Power-up Timer, que introduce un pequeño retraso en el encendido para estabilizar el voltaje.
- CP: Deshabilita la protección de código, permitiendo que el código sea reprogramado o leído.
- _XTAL_FREQ: Define la frecuencia de oscilación del microcontrolador a 4 MHz, importante para los retardos.
- ledState: Es una variable de tipo int que se utiliza para almacenar el estado del LED. volatile indica que la variable puede cambiar inesperadamente (en este caso, debido a la interrupción), asegurando que siempre se lea directamente de la memoria.

- Función de Interrupción (isr)
    - Esta función es la rutina de interrupción que se ejecuta cuando el pin RB0 detecta un flanco ascendente (cuando el botón se presiona).
    - INTF: Se verifica la bandera de interrupción externa (INTF) para confirmar si fue generada por el pin RB0. Si es cierto, se cambia el estado del LED (ledState = !ledState).
    - INTF = 0: Limpia la bandera de interrupción, permitiendo detectar el próximo cambio de estado.
- main
    - TRISB0 y TRISB1: Configura el pin RB0 como entrada para el botón y el pin RB1 como salida para el LED.
    - INTE y GIE: Habilita la interrupción externa en RB0 y permite el uso de interrupciones globales.
    - OPTION_REGbits.INTEDG: Define la interrupción en flanco ascendente, es decir, detecta el momento en que el botón se presiona.
    - Bucle Principal (while(1)): Mantiene el valor de ledState reflejado en el pin RB1, permitiendo que el LED encienda o apague en función del último cambio.

2. ANÁLISIS DE RESULTADOS

- ¿Por qué usamos el registro INTF en la función de interrupción?

INTF es la bandera de interrupción externa que indica si el pin RB0 detectó un evento de flanco ascendente. Al verificar INTF, el microcontrolador sabe si la interrupción fue generada por el botón.

- ¿Qué sucede cuando ledState cambia de valor?

Cuando ledState cambia (de 0 a 1 o de 1 a 0), el estado de RB1 cambia, alternando el encendido o apagado del LED.

- ¿Qué pasaría si no se limpia la bandera INTF en la interrupción?

Si no se limpia INTF, el sistema interpretará que la interrupción no ha sido atendida, lo que puede causar que el código quede atrapado en una interrupción continua, bloqueando el programa.

- ¿Qué función cumple el pull-down resistor en el botón?

El resistor pull-down asegura que el pin RB0 tenga un valor lógico bajo cuando el botón no está presionado, evitando lecturas incorrectas debido al ruido eléctrico.

- ¿Por qué se usa volatile en la variable ledState?

volatile asegura que el compilador no optimice el acceso a ledState, permitiendo que la variable se lea directamente de la memoria cada vez que se utiliza, lo cual es importante ya que puede cambiar por la interrupción.

3. CONCLUSIONES

a. Este proyecto demuestra el uso de interrupciones externas en el PIC16F84A, permitiendo que el sistema responda rápidamente a cambios externos sin depender de un bucle continuo para monitorear el botón.

b. Al utilizar una variable ledState, el microcontrolador puede almacenar el estado del LED de forma eficiente y alternarlo sin necesidad de instrucciones complejas.

c. El uso de un resistor pull-down en el botón evita lecturas incorrectas, mejorando la estabilidad y precisión en la detección de la señal de entrada.