PRÁCTICA DE PICs - SEMÁFORO

NIVEL: 1
DURACIÓN: 2 horas
OBJETIVO: Los estudiantes aprenderán a simular el funcionamiento de un semáforo con tres LEDs (rojo, amarillo y verde) que cambian de estado según un temporizador predefinido.

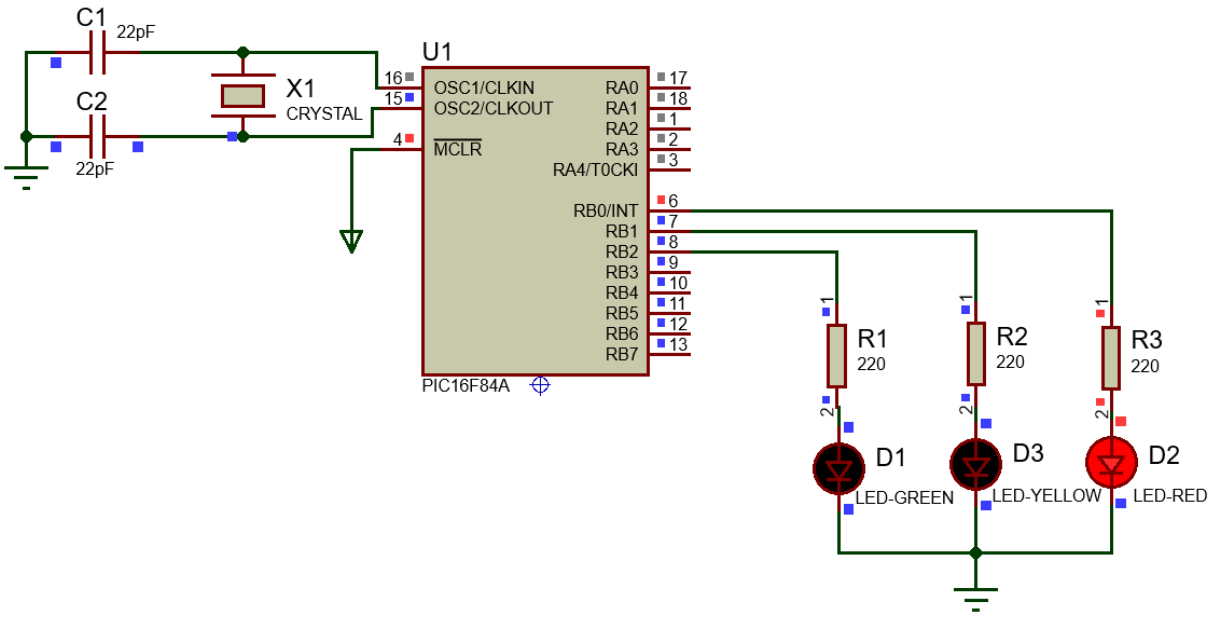
MATERIALES

- PIC16F84A
- LEDs (rojo, amarillo y verde)
- Resistores (220Ω para cada LED)
- 1 Oscilador de cristal (4MHz)
- 2 Capacitores cerámicos (22pF)
- 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 (usa 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

// Definimos los tiempos de retardo para cada LED (en milisegundos)
#define TIEMPO_VERDE 2000      // 2 segundos para LED verde
#define TIEMPO_AMARILLO 2000    // 2 segundos para LED amarillo
#define TIEMPO_ROJO 2000        // 2 segundos para LED rojo

void main(void) {
    // Configuración de puertos
    TRISB = 0x00;               // Configura todo PORTB como salida
    PORTB = 0x00;               // Apaga todos los LEDs inicialmente

    while (1) {
        // Encender LED verde
        PORTBbits.RB2 = 1;      // LED verde en RB2
        __delay_ms(TIEMPO_VERDE);
        PORTBbits.RB2 = 0;      // Apaga LED verde

        // Encender LED amarillo
        PORTBbits.RB1 = 1;      // LED amarillo en RB1
        __delay_ms(TIEMPO_AMARILLO);
        PORTBbits.RB1 = 0;      // Apaga LED amarillo

        // Encender LED rojo
        PORTBbits.RB0 = 1;      // LED rojo en RB0
        __delay_ms(TIEMPO_ROJO);
        PORTBbits.RB0 = 0;      // Apaga LED rojo
    }
}
```

Figura 1. Source Files (main)

Explicación

- Se definen constantes para los tiempos que cada LED permanecerá encendido:

- TIEMPO_VERDE para el LED verde (2 segundos),
- TIEMPO_AMARILLO para el LED amarillo (2 segundos),
- TIEMPO_ROJO para el LED rojo (5 segundos).

- Configuración del Puerto B:

- TRISB = 0x00; configura todos los pines de PORTB como salidas.
- PORTB = 0x00; asegura que todos los LEDs comiencen apagados.

- Bucle Principal (while(1)):

- LED verde: Se enciende el LED verde en RB2 y se mantiene encendido durante el tiempo definido en TIEMPO_VERDE. Luego, se apaga.
- LED amarillo: A continuación, se enciende el LED amarillo en RB1 durante TIEMPO_AMARILLO, y luego se apaga.
- LED rojo: Finalmente, se enciende el LED rojo en RB0 durante TIEMPO_ROJO y luego se apaga.

2. ANÁLISIS DE RESULTADOS

- ¿Por qué se usa __delay_ms() en lugar de un temporizador interno?

La función __delay_ms() permite implementar retardos de forma simple en código sin configurar registros adicionales, ideal para un proyecto básico como este. Sin embargo, si se desea mayor precisión o liberar al microcontrolador para realizar otras tareas mientras se controla el tiempo, sería mejor emplear un temporizador interno, como el Timer0.

- ¿Cómo afecta la frecuencia del oscilador a los retardos de __delay_ms()?

La frecuencia del oscilador afecta directamente a __delay_ms(), ya que el tiempo de retardo se calcula en función de esta frecuencia. En este proyecto, definimos la frecuencia a 4 MHz. Si la frecuencia cambia, los retardos también cambiarán a menos que se ajuste la constante _XTAL_FREQ.

- ¿Qué sucede si los LEDs no están conectados en los pines correctos?

Si los LEDs no están en los pines especificados en el código (RB0 para rojo, RB1 para amarillo, y RB2 para verde), no se encenderán en el orden correcto del semáforo. Por lo tanto, es crucial conectar los LEDs en los pines indicados en el programa.

- ¿Es posible agregar más estados al semáforo (por ejemplo, LED parpadeando en amarillo)?

Sí, es posible agregar estados adicionales modificando el código. Se podría, por ejemplo, hacer parpadear el LED amarillo al agregar ciclos adicionales de encendido/apagado con retardos cortos dentro de su sección del código.

- ¿Por qué PORTB se configura como salida?

PORTB se configura como salida para que el microcontrolador pueda enviar señales de encendido y apagado a los LEDs. Cada LED necesita un pin de salida para recibir la señal de 5V (alta) o 0V (baja) según el estado deseado.

- ¿Qué sucede si se utilizan resistencias de diferente valor para los LEDs?

El valor de las resistencias afecta la intensidad del brillo de cada LED. Si las resistencias son mayores, el brillo será menor, y si son menores, el LED brillará más. La resistencia de 220Ω es una elección común para proteger el LED sin reducir demasiado su brillo, pero valores diferentes podrían cambiar su apariencia.

3. CONCLUSIONES

- Este proyecto permite entender cómo se puede controlar el tiempo en microcontroladores usando retardos simples. Aunque se utilizaron funciones de retardo (__delay_ms()), un temporizador interno podría haber brindado mayor precisión y flexibilidad en el control del semáforo.

- El uso de LEDs en una secuencia predefinida ayuda a familiarizarse con la lógica secuencial en programación, lo cual es esencial en sistemas de control básicos como los semáforos.

- Configurar los pines de salida y su asociación con los LEDs permite comprender cómo el PIC maneja los puertos y cómo estos se pueden utilizar para controlar dispositivos externos.

- Este código se puede expandir para simular diferentes tipos de semáforos (por ejemplo, semáforos para peatones o modos de parpadeo). Esto demuestra la adaptabilidad de los microcontroladores y cómo una lógica simple puede extenderse para aplicaciones más complejas.