UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE MECANICA ELÉCTRICA
MICROCONTROLADORES

INTEGRANTES:

No.	NOMBRE	CARNÉ
1	Cindy Melissa Gatica Arriola	201709692
2	Rodrigo Sosa Aquino	202012337
3	Stheeven Adonías, Coc Chán	201700519
4		
5		

PRACTICA No. 1
CONTROL DE ENTRADAS Y SALIDAS

OBJETIVOS

- Que el estudiante se familiarice con el entorno Code Composer Studio.
- Implementar el conocimiento adquirido en una aplicación práctica.

CONOCIMIENTOS PREVIOS

- Configuración inicial y relojes.
- Habilitación de entradas y salidas
- Debug.

FORMA DE ENTREGA

- La práctica se entrega una semana después de su publicación.
- Subir los archivos a la plataforma en formato PDF.
- Agregar programa como carpeta comprimida, con el nombre Práctica_01_Micro.

DESCRIPCIÓN

La práctica consiste en habilitar todo el puerto B (B0-B7) como salida. Creando un efecto cascada, cada led debe estar encendido 1 segundo antes de saltar al siguiente. Esto deberá activarse mediante un pulsador externo, cualquier pin que no sea PF0 ni PF4, cuando el pulsador se suelte, deberá permanecer el último led activo encendido, al presionarlo nuevamente continuar a partir de donde se quedó.

En formato PDF subir el código utilizado. La semana siguiente a la publicación deberán presentar el programa.

Código

```
// Predefinidas
#include <stdint.h>
#include <stdbool.h>

// Agregadas microcontrolador
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/tm4c123gh6pm.h"

// Librerías extras
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"

// Las variables se declaran abajo de las librerías
int valor = 0;
int ultimo = 1;  // Inicia con el primer LED (PB0)
int estado_cascada = 0; // Para mantener el estado actual de la cascada
bool boton_presionado_anterior = false;

int main(void)
{
  SysCtlClockSet(SYSCTL_OSC_MAIN | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ | SYSCTL_SYSDIV_2_5); // Definir el CLK
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);  // Habilitar el puerto B
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);  // Habilitar el puerto A

  GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_0 | GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3 | GPIO_PIN_4 |
GPIO_PIN_5 | GPIO_PIN_6 | GPIO_PIN_7); // Definir los pines de salida PB0 a PB7
  GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_2); // Definir pin de entrada PA2 (botón)
  GPIOPadConfigSet(GPIO_PORTA_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU); //
Configurar PA2 con pull-up

  while(1) {
    valor = GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_2); // Leer el botón

    if (valor == 0) { // Si el botón está presionado (valor es 0 por pull-up)
      if (!boton_presionado_anterior) { // Si el botón estaba previamente suelto, continuar desde el último LED
        boton_presionado_anterior = true; // Ahora el botón está presionado
      }

      // Mientras el botón está presionado, la secuencia continúa
      switch (estado_cascada) {
        case 0:
          GPIOPinWrite(GPIO_PORTB_BASE, 0xFF, 1); // Encender PB0
          ultimo = 1;  // Guardar último LED encendido
```

```
            break;
        case 1:
            GPIOPinWrite(GPIO_PORTB_BASE, 0xFF, 2); // Encender PB1
            ultimo = 2;
            break;
        case 2:
            GPIOPinWrite(GPIO_PORTB_BASE, 0xFF, 4); // Encender PB2
            ultimo = 4;
            break;
        case 3:
            GPIOPinWrite(GPIO_PORTB_BASE, 0xFF, 8); // Encender PB3
            ultimo = 8;
            break;
        case 4:
            GPIOPinWrite(GPIO_PORTB_BASE, 0xFF, 16); // Encender PB4
            ultimo = 16;
            break;
        case 5:
            GPIOPinWrite(GPIO_PORTB_BASE, 0xFF, 32); // Encender PB5
            ultimo = 32;
            break;
        case 6:
            GPIOPinWrite(GPIO_PORTB_BASE, 0xFF, 64); // Encender PB6
            ultimo = 64;
            break;
        case 7:
            GPIOPinWrite(GPIO_PORTB_BASE, 0xFF, 128); // Encender PB7
            ultimo = 128;
            break;
      }

      // Avanzar al siguiente estado de la cascada
      estado_cascada = (estado_cascada + 1) % 8; // Se reinicia después del último LED

      // Retardo para efecto de cascada
      SysCtlDelay(80000000 / 3);
    } else {
      // Si el botón no está presionado, mantener el último LED encendido
      GPIOPinWrite(GPIO_PORTB_BASE, 0xFF, ultimo);

      // Marcar que el botón no está presionado para que pueda continuar la secuencia al ser presionado
      boton_presionado_anterior = false;
    }
  }
}
```