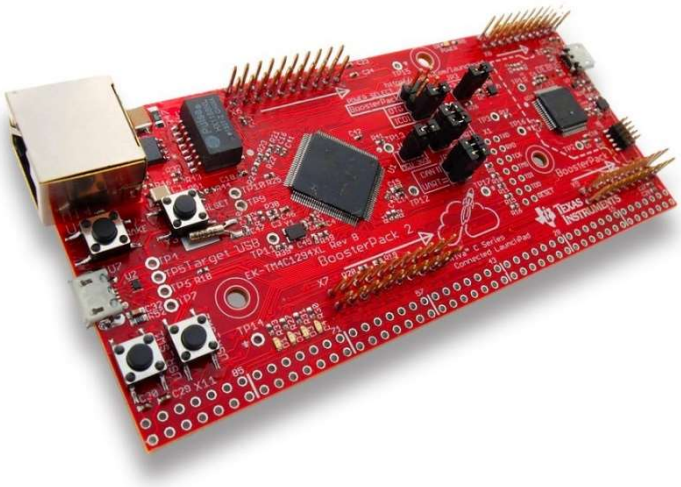


SISTEMAS ELECTRÓNICOS PARA AUTOMATIZACIÓN

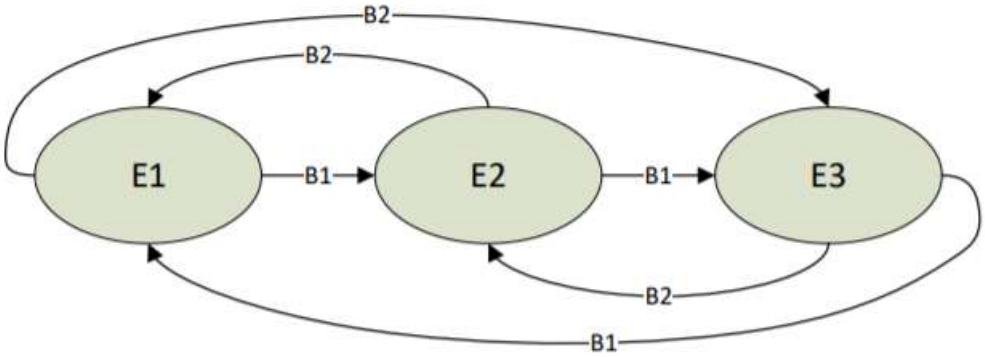
Práctica 1

Álvaro Calvo Matos
Damián Jesús Pérez Morales

Índice

1. INTRODUCCIÓN

En la primera práctica de la asignatura, se pretende familiarizarse con el microcontrolador "TIVA TM4C1294NCPDT" y con la librería "DriverLib", resolviendo un par de ejercicios con máquinas de estados para trabajar con los LEDs y botones implementados en el propio microcontrolador. En el primero de ellos, se hace pasar entre los estados pulsando los botones sin rutinas de interrupción y sin timers; y, en el segundo de ellos, se recurre a las rutinas de interrupción de los botones. La máquina de estados que se implementa es la que se exige en el enunciado de la práctica:



- E1: Se encienden los 4 leds de la placa durante 0'1s y se apagan 0'9s, o sea, un 10% de duty cycle.
- E2: Se realiza una secuencia de encendido de LEDs, empezando por el que está situado más a la izquierda hasta el último de ellos, permaneciendo en el tránsito los leds encendidos y con un período de 1s entre el encendido de LED y LED y, cuando los cuatro LEDs estén encendidos, se apagarán durante 3s.
- E3: Se encenderán de forma permutada el primer y el tercer LED durante 0'5s y, seguidamente, el segundo y cuarto LED durante otros 0'5s.

2. EJERCICIO 1

En el ejercicio 1, se hace pasar entre los diferentes estados mencionados en la introducción de la memoria sin emplear rutinas de interrupción, por lo que se tendrán problemas a la hora de permutar entre estados, ya que sería necesario tener el botón pulsado justo al finalizar las instrucciones a ejecutar de cada estado. El código empleado es el siguiente:

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

#include "driverlib2.h"
/*********************************************************
 * Primer ejemplo de manejo de pines de e/s, usando el HW de la placa
 * Los pines se definen para usar los leds y botones:
 *      LEDS: F0, F4, N0, N1
 *      BOTONES: J0, J1
 * Cuando se pulsa (y se suelta)un botón, cambia de estado,
 * entre los definidos en la matriz LED. El primer botón incrementa el estado
 * y el segundo lo decrementa. Al llegar al final, se satura.
 *********************************************************/


#define MSEC 40000 //Valor para 1ms con SysCtlDelay() cuando el reloj está
configurado a 120MHz
#define MaxEst 3    //Variable auxiliar para establecer el numero de estados


//Definiciones para facilitar la lectura de los pulsadores
#define B1_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)
#define B1_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0))
#define B2_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1)
#define B2_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1))


uint32_t reloj=0;


int main(void)
{
    int estado;
    //Fijar velocidad del reloj a 120MHz
    reloj = SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN |
SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480), 120000000);

    //Habilitar los periféricos implicados: GPIO F, J, N
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);
      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
```

```
            //Definir tipo de pines
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 |GPIO_PIN_4); //F0 y F4:
salidas
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0 |GPIO_PIN_1); //N0 y N1:
salidas
    GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);   //J0 y J1:
entradas

GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_STRENGTH_2MA,GPIO_PIN_T
YPE_STD_WPU); //Pullup en J0 y J1

    estado = 0; //Inicializacion de la variable de estado

    while(1){
        //Mediante una estructura switch-case ejecutamos la rutina
        //correspondiente al estado en el que nos encontremos
        switch(estado)
        {
        case 0:
            //Encendemos todos los leds y esperamos 100 ms
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
            SysCtlDelay(100*MSEC);

            //Apagamos todos los leds durante 900 ms
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, 0);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);
            SysCtlDelay(900*MSEC);
            break;
        case 1:
            //Encendemos el primer led y esperamos 1 segundos
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);
            SysCtlDelay(1000*MSEC);
            //Encendemos el siguiente led y esperamos 1 segundos
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
            SysCtlDelay(1000*MSEC);
            //Encendemos el siguiente led y esperamos 1 segundos
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);
            SysCtlDelay(1000*MSEC);
            //Encendemos el ultimo led y esperamos 1 segundos
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
            SysCtlDelay(1000*MSEC);

            //Apagamos los 4 leds y esperamos 3 segundos
             GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, 0);
             GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0);
             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, 0);
             GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);
```

```
      //Encendemos los leds en las posiciones pares durante 500 ms
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, 0);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
            SysCtlDelay(500*MSEC);
            break;
        default: break;
        }

        //Al salir terminar la rutina del estado actual comprobamos si hay algun
boton
        //pulsado y lo gestionamos de ser así actualizando consecuentemente el
estado
        if(!(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0))){     //Si se aprieta el
boton 1
            SysCtlDelay(10*MSEC);
            while(!(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)));  //Debouncing...
            SysCtlDelay(10*MSEC);
            estado++; if(estado==MaxEst) estado=0;       //Incrementa el estado. Si
máximo, vuelve a cero
        }
        if( !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1))){        //Si se aprieta el
botón 2
            SysCtlDelay(10*MSEC);
            while( !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1))); //Debouncing..
            SysCtlDelay(10*MSEC);
            estado--; if(estado==-1) estado=MaxEst - 1;      //Decrementa el
estado. Si menor que cero, salta a Max.
        }

    }
}
```

3. EJERCICIO 2

Este segundo ejercicio es muy similar al primero, pero con la mejora de las rutinas de interrupción para cada botón y se solucionaría el problema mencionado en este primer ejercicio, pero con la pega de que no se cambia de estado justo al pulsar el botón, sino al finalizar las instrucciones del estado. El código implementado es el siguiente:

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

#include "driverlib2.h"
/*********************************************************
 * Este código es una modificación de la "Variante del EJ2, pero
 * manejando los pines por interrupción".
 *
 * Ahora se mejora el comportamiento respecto al apartado anterior
 * gracias al uso de interrupciones para gestionar la pulsación de
 * los botones en el instante en que el usuario las lleva a cabo
 *********************************************************/


#define MSEC 40000 //Valor para 1ms con SysCtlDelay()
#define MaxEst 3    //Variable auxiliar para establecer el numero de estados


//Definiciones para facilitar la lectura de los pulsadores
#define B1_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)
#define B1_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0))
#define B2_OFF GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1)
#define B2_ON !(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1))


uint32_t reloj=0;
int estado;


/*******************************************************
 * Rutina de interrupción del puerto J.
 * Leo los pines, y en cada caso hago la función necesaria,
 * (incrementar o decrementar el estado)
 * Establezco 20ms de debouncing en cada caso
 * Y debo borrar el flag de interrupción al final.
 *******************************************************/
```

```
void rutina_interrupcion(void)
{
    if(B1_ON)
{
        //Esperamos a que se suelte el boton
        while(B1_ON);
        //Debouncing
        SysCtlDelay(20*MSEC);
        //Actualizamos el estado
        estado++;
        if(estado==MaxEst) estado=0;
        //Borramos el flag de interrupsión
        GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_0);
    }
    if(B2_ON)
      {
        //Esperamos a que se suelte el boton
          while(B2_ON);
          //Debouncing
          SysCtlDelay(20*MSEC);
          //Actualizamos el estado
          estado--; if(estado==-1) estado=MaxEst - 1;
          //Borramos el flag de interrupsión
          GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_1);
      }

}


int main(void)
{
    int estado_ant;
    //Fijar velocidad a 120MHz
    reloj=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN |
SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480), 120000000);

    //Habilitar los periféricos implicados: GPIO F, J, N
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

    //Definir tipo de pines
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 |GPIO_PIN_4); //F0 y F4:
salidas
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0 |GPIO_PIN_1); //N0 y N1:
salidas
    GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);   //J0 y J1:
entradas

GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0|GPIO_PIN_1,GPIO_STRENGTH_2MA,GPIO_PI
N_TYPE_STD_WPU); //Pullup en J0 y J1

    //Inicializacion de las variables de estado.
    estado=0;
    estado_ant=0;
```

```
    GPIOIntEnable(GPIO_PORTJ_BASE, GPIO_PIN_0|GPIO_PIN_1);  // Habilitar pines
de interrupción J0, J1
   GPIOIntRegister(GPIO_PORTJ_BASE, rutina_interrupcion);  //Registrar (definir)
la rutina de interrupción
    IntEnable(INT_GPIOJ);                                    //Habilitar
interrupción del pto J
    IntMasterEnable();                                       // Habilitar
globalmente las ints

    while(1){
        if(estado != estado_ant)  //Para no estar continuamente accediendo a los
puertos...
        {
            estado_ant=estado;
        }
        //Mediante una estructura switch-case ejecutamos la rutina
        //correspondiente al estado en el que nos encontremos
        switch(estado)
        {
        case 0:
            //Encendemos todos los leds y esperamos 100 ms
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
            SysCtlDelay(100*MSEC);

            //Apagamos todos los leds durante 900 ms
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, 0);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);
            SysCtlDelay(900*MSEC);
            break;
        case 1:
            //Encendemos el primer led y esperamos 1 segundos
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);
            SysCtlDelay(1000*MSEC);
            //Encendemos el siguiente led y esperamos 1 segundos
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
            SysCtlDelay(1000*MSEC);
            //Encendemos el siguiente led y esperamos 1 segundos
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);
            SysCtlDelay(1000*MSEC);
            //Encendemos el ultimo led y esperamos 1 segundos
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
            SysCtlDelay(1000*MSEC);

            //Apagamos los 4 leds y esperamos 3 segundos
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, 0);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);
            SysCtlDelay(3000*MSEC);
             break;
```

```
        case 2:
            //Encendemos los leds en las posiciones impares durante 500 ms
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0);
            SysCtlDelay(500*MSEC);

            //Encendemos los leds en las posiciones pares durante 500 ms
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, 0);
            GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, 0);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
            SysCtlDelay(500*MSEC);
            break;
        default: break;
        }
    }
}
```

4. COMENTARIOS ACERCA DE LA PRÁCTICA

En cuanto a la realización de la práctica, no se ha tenido ningún problema. Resultó bastante sencilla al tener los ejemplos de clase de apoyo y fue bastante trivial ajustarlo a las especificaciones requeridas para la práctica, ya que eran ejercicios bastante similares a los propuestos.