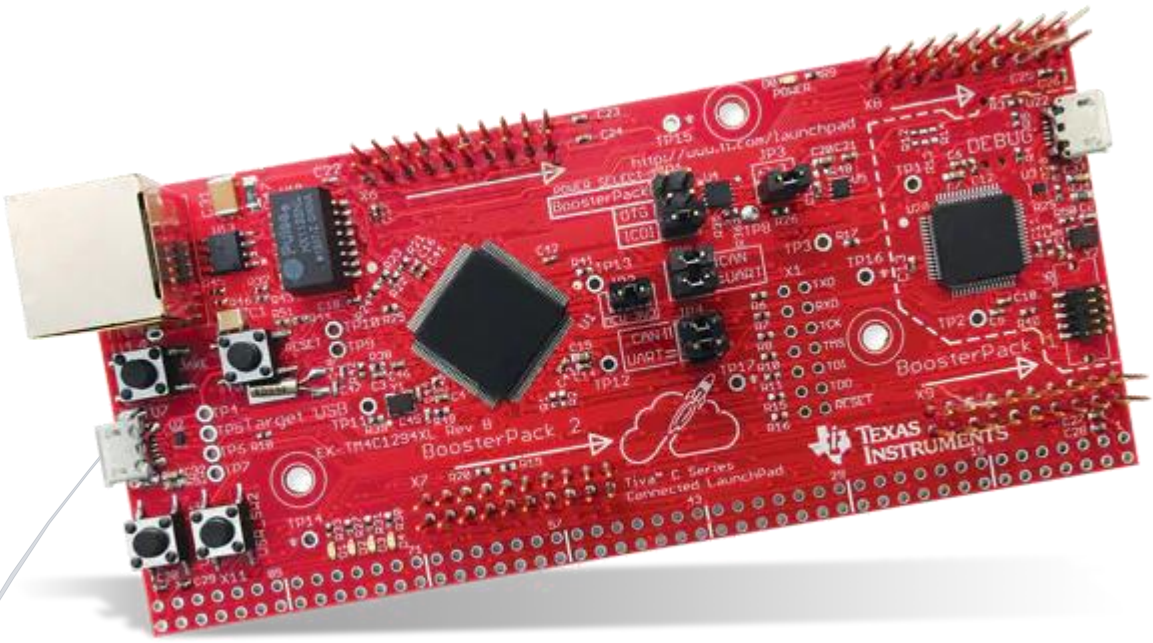
4ºGIERM

Práctica 1 SEPA

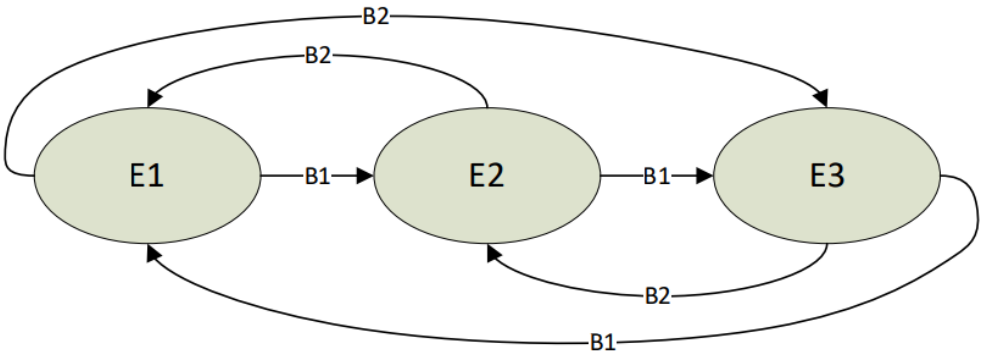
Manejo básico del ConnectedLaunchpad de Texas Instruments



Daniel Peinado Ramírez, Francisco Javier Román Escorza
SISTEMAS ELECTRÓNICOS PARA LA AUTOMATIZACIÓN

Ejercicio 1. Primer ejemplo básico:

Realizar un primer ejemplo básico, desde cero, que realice un programa en el cual se haga lo siguiente: Deberá recorrer los modos 1, 2 y 3 según se pulsen los botones B1 y B2, atendiendo al siguiente diagrama de modos de funcionamiento:



- ***En el modo 1, Empezará parpadeando los 4 leds de la placa, con un periodo de 1s y un duty cycle del 10% (0.1s encendido, 0.9s apagado).***
- ***En el modo 2 realizará la siguiente secuencia: empezando por todos los leds apagados, los irá encendiendo, esperando 1s entre uno y otro, y cuando llegue al cuarto, esperará 3s antes de volver al principio.***
- ***En el modo 3, la secuencia que se pretende de encendido y apagado será 1010-0101, con un intervalo de 500ms entre uno y otro.***

En este primer ejercicio comenzamos definiendo como variables globales los botones B1 y B2 que utilizaremos para cambiar de estado, la variable MSEC 40000, para establecer el valor de 1ms que utilizaremos para mantener encendidos o apagados los pines durante el tiempo deseado, así como MaxEst=3, que declara el estado máximo. Dentro de la función principal definimos la función estado, que es la que va a determinar si nos encontramos en el estado 1,2 o 3. En el caso que esté en el estado 3, y se intente aumentar de estado, se fuerza la variable a 1. En el caso opuesto, de que estando en el estado 1, intente disminuir, se fuerza la variable a 3.

Según nos encontremos en el estado 1, 2 o 3; con las funciones GPIOinWrite encendemos o apagamos los leds según se nos pida durante el tiempo especificado gracias a la variable MSEC definida globalmente. Cada vez que impongamos un 0 como último input de la función apagaremos los pines, mientras que escribiendo:

1. Para GPIO_PORTN_BASE:
    - GPIO_PIN_1
    - GPIO_PIN_0
2. Para GPIO_PORTF_BASE:
    - GPIO_PIN_4
    - GPIO_PIN_0

Encenderemos cada uno de los 4 pines.

Código:

```
#include <stdint.h>
#include <stdbool.h>
//Incluimos las librerías necesarias

#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()
#define MaxEst 3 //Definimos globalmente el estado máximo = 3, considerando
cada uno de los estados que se especifican en el enunciado
//Definimos tambien como variables globales los botones B1 y B2 para aumentar
y disminuir el estado, respectivamente
#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 la 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 en el eje: GPIOF, GPIOJ, GPION
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);

    //Definir tipo de pines, los botones como entradas y los leds como
salidas
    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_TYPE_STD_WPU); //Pullup en J0 y J1

    estado=1; // Variable para cada uno de los modos de funcionamientos: E1,
E2, E3.

    while(1){
        if(!(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0))){         //Si se
aprieta el botón 1
```

```
            SysCtlDelay(10*MSEC);
            while(!(GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)));
      //Debouncing...
            SysCtlDelay(10*MSEC);
            estado++; if(estado>MaxEst) estado=1;         //Incrementa el
estado. Si estado > 3, vuelve al estado 1
        }
        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=3;       //Decrementa el estado.
Si menor que uno, redirige al estado 3
        }

       if (estado == 1) //Si estamos en el estado 1:
       {
           //Encendemos todos los pines de los leds
           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);
           //Esperamos 0.1s
           SysCtlDelay(100*MSEC);
           //Apagamos todos los pines de los leds
           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);
           //Esperamos los 0.9s restantes, posteriormente se volverá al
inicio del ciclo
           SysCtlDelay(900*MSEC);

       }
       else if (estado == 2) //Si estamos en el estado 2:
       {
           //Comenzamos con todos los pines de los leds apagados
           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(1000*MSEC);
           //Encendemos progresivamente cada pin cada segundo
           GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);
           SysCtlDelay(1000*MSEC);
           GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
           SysCtlDelay(1000*MSEC);
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);
           SysCtlDelay(1000*MSEC);
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
           //Espera final de 3 segundos antes de volver a empezar el ciclo
           SysCtlDelay(3000*MSEC);
       }
       else if (estado == 3)//Si estamos en el estado 3:
       {
           //Establecemos la primera configuracion de pines (1010)
           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);
            //Esperamos 0.5s
            SysCtlDelay(500*MSEC);
            //Establecemos la segunda configuracion de pines (0101)
            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);
            //Esperamos los 0.5s restantes antes de repetir el ciclo
            SysCtlDelay(500*MSEC);

        }
        else  //En caso de error
            estado == 1;
}}
```

Ejercicio 2

En el segundo ejercicio, implementamos los estados de forma análoga al primero. Sin embargo, la transición ahora la realizamos mediante interrupciones, lo cual resulta en una transición entre estados más fluida y precisa. Esto, en la práctica se refleja en que no debemos buscar pulsar los botones al final de los estados, si no que se cambiarán tantos estados como número de pulsaciones se hayan realizado antes de terminar el estado.

Para implementarlo, desechamos la lectura directa de la pulsación del botón e implementamos una función de la rutina de interrupción que permitirá el correcto y deseado funcionamiento del programa.

Código:

```
#include <stdint.h>
#include <stdbool.h>
//Incluimos las librerías necesarias

#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()
#define MaxEst 3//Definimos globalmente el estado máximo = 3, considerando
cada uno de los estados que se especifican en el enunciado
//Definimos tambien como variables globales los botones B1 y B2 para aumentar
y disminuir el estado, respectivamente
#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))
```

```
int estado; //En este caso, definimos la variable estado globalmente, fuera de la función principal,
//debido a que la usaremos dentro de la función de la interrupción

//Rutina que modificará el estado al pulsar el interruptor por interrupción
void rutina_interrupcion(void)
{
    if(B1_ON)
    {
        while(B1_ON); //Mientras se pulsa el botón 1
        SysCtlDelay(20*MSEC);
        estado++;
        if(estado>MaxEstado) estado=1; //Incrementa el estado. Si es mayor que 3, redirigimos al estado 1.
        GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_0); //Borra la interrupción pendiente
    }
    if(B2_ON)
      {
          while(B2_ON); //Mientras se pulsa el botón 2
          SysCtlDelay(20*MSEC);
          estado--;
          if(estado<1) estado=3;      //Decrementa el estado. Si menor a uno, redirigimos al estado 3.
          GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_1); //Borra la interrupción pendiente
      }

}

uint32_t reloj=0;

int main(void)
{

    //Fijar velocidad a 120MHz
    reloj=SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ | SYSCTL_OSC_MAIN | SYSCTL_USE_PLL | SYSCTL_CFG_VCO_480), 120000000);

    //Habilitar los periféricos implicados: GPIOF, 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_TYPE_STD_WPU); //Pullup en J0 y J1

    estado=1; // Variable para los modos de funcionamientos E1, E2, E3.
```

```
//  SysCtlPeripheralClockGating(true);                      //Habilitar el
apagado selectivo de periféricos (no lo hemos utilizado)
    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 == 1)
       {
           //encendemos todos los pines de los leds
           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);
           //esperamos 0.1s
           SysCtlDelay(100*MSEC);
           //apagamos todos los pines de los leds
           GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1*0);
           GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0*0);
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4*0);
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0*0);
           //esperamos los 0.9s restantes
           SysCtlDelay(900*MSEC);
       }
       else if (estado == 2)
       {
           //comenzamos con todos los pines de los leds apagados
           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(1000*MSEC);
           //comenzamos a encender cada pin cada segundo
           GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, GPIO_PIN_1);
           SysCtlDelay(1000*MSEC);
           GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0, GPIO_PIN_0);
           SysCtlDelay(1000*MSEC);
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_PIN_4);
           SysCtlDelay(1000*MSEC);
           GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
           SysCtlDelay(3000*MSEC);
       }
       else if (estado == 3)
       {
           //establecemos la primera configuracion de pines (1010)
           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);
           //esperamos 0.5s
           SysCtlDelay(500*MSEC);
           //establecemos la segunda configuracion de pines (0101)
           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);
            //esperamos los 0.5s restantes
            SysCtlDelay(500*MSEC);
        }
        else //En caso de error
            estado == 1;

         }

}
```