Universidad de Barcelona

Arthur Font Gouveia 20222613

Ángel Rubio Giménez 20222484

Programació d'Arquitectures Encastades

Pràctica 2 - Configuració de Ports

Barcelona

2020

Índice

1. Proyecto

1.1 Objetivos

Los objetivos de esta práctica han sido la configuración de los puertos GPIOs. En la placa superior (Boosterpack MK II) utilizamos los botones S1 y S2, el Joystick, y los tres LEDs RGB. Ya en la placa de interface inferior (Adaptador MSP432-Bioloid) utilizamos 8 LEDs.

1.2 Recursos utilizados

Primero configuramos los pines asociados al joystick y a los botones como entradas y los pines asociados a los LEDs como salidas.

Recursos	Px.y
Joystick Dreta	P4.5
Joystick Esquerra	P4.7
Joystick Centre	P4.1
Joystick Amunt	P5.4
Joystick Avall	P5.5
Polsador S1	P5.1
Polsador S2	P3.5
LED_R (vermell)	P2.6
LED_G (verd)	P2.4
LED_B (blau)	P5.6

Taula 1 (Px.y: Port x, pin y)

Los 8 LEDs de la placa de interface están conectados al puerto P7 (pines P7.0...P7.7).

Para configurar los pines correspondientes a nuestro hardware como GPIOs hemos configurado los registros PxSEL0 y PxSEL1 a 0.

Para seleccionar si queremos que un pin sea de entrada o salida utilizamos el registro PxDIR, si ponemos un 0 en este el pin correspondiente será de entrada y si ponemos un 1 será de salida.

Para la manipulación de los registros utilizamos los operadores lógicos para crear las máscaras que nos interesan.

Por ejemplo para encender el LED referente al P7.1 utilizamos la máscara el registro: P7OUT |= 0x01.

1.3 Problemas

Un problema de los que hemos tenido ha sido a la hora de configurar las interrupciones, porque no sabíamos que los pines tenían un valor asociado en el vector de interrupciones PxIV y asumimos que ese valor era la posición del pin. Luego miramos en la documentación la tabla de las interrupciones el valor respectivo a cada pin.

Table 10-4. PxIV Register Description

Bit	Field	Type	Reset	Description
15-5	Reserved	R	0h	Reserved. Reads return 0h
4-0	PxIV	R	0h	Port x interrupt vector value 00h = No interrupt pending 02h = Interrupt Source: Port x.0 interrupt; Interrupt Flag: PxIFG.0; Interrupt Priority: Highest 04h = Interrupt Source: Port x.1 interrupt; Interrupt Flag: PxIFG.1 06h = Interrupt Source: Port x.2 interrupt; Interrupt Flag: PxIFG.2 08h = Interrupt Source: Port x.3 interrupt; Interrupt Flag: PxIFG.3 0Ah = Interrupt Source: Port x.4 interrupt; Interrupt Flag: PxIFG.4 0Ch = Interrupt Source: Port x.5 interrupt; Interrupt Flag: PxIFG.5 0Eh = Interrupt Source: Port x.6 interrupt; Interrupt Flag: PxIFG.6 10b = Interrupt Source: Port x.7 interrupt; Interrupt Flag: PxIFG.7; Interrupt Priority: Lowest

1.4 Conclusiones

Esta práctica nos ha servido para entender cómo funciona el robot y para aprender a configurar los GPIO y sus interrupciones. También nos ha resultado muy útil para aprender a utilizar el *Technical Reference Manual*, que nos ha servido para resolver cualquier duda en cuanto al valor de los pines y para alcanzar los objetivos propuestos en esta práctica.

2. Código comentado

```
/*****************************
 *
 * Practica_02_PAE Programació de Ports
 * i pràctica de les instruccions de control de flux:
 * "do ... while", "switch ... case", "if" i "for"
 * UB, 02/2017.
 *****************************/

#include <msp432p401r.h>
#include <stdio.h>
#include <stdint.h>
#include "lib_PAE2.h" //Libreria grafica + configuracion reloj MSP432

#define Button_S1 1
#define Button_S2 2
#define Jstick_Left 3
#define Jstick_Right 4
#define Jstick_Up 5
#define Jstick_Down 6
#define Jstick_Center 7

char saludo[16] = " PRACTICA 2 PAE";//max 15 caracteres visibles
char cadena[16];//Una línea entera con 15 caracteres visibles + uno
oculto de terminación de cadena (codigo ASCII 0)
char limite[16];
char borrado[] = "               "; //una línea entera de 15 espacios
en blanco
uint8_t linea = 0;
```

```
uint8_t estado=0;

uint8_t estado_anterior = 8;

uint32_t retraso = 500000;


/**************************************************************************
*****

 * INICIALIZACIÓN DEL CONTROLADOR DE INTERRUPCIONES (NVIC).

 *

 * Sin datos de entrada

 *

 * Sin datos de salida

 *


**************************************************************************
****/

void init_interrupciones(){

    // Configuración al estilo MSP430 "clasico":

    // --> Enable Port 4 interrupt on the NVIC.

    // Segun el Datasheet (Tabla "6-39. NVIC Interrupts", apartado
"6.7.2 Device-Level User Interrupts"),

    // la interrupción del puerto 4 es la User ISR numero 38.

    // Segun el Technical Reference Manual, apartado "2.4.3 NVIC
Registers",

    // hay 2 registros de habilitacion ISER0 y ISER1, cada uno para 32
interrupciones (0..31, y 32..63, resp.),

    // accesibles mediante la estructura NVIC->ISER[x], con x = 0 o x
= 1.

    // Asimismo, hay 2 registros para deshabilitarlas: ICERx, y dos
registros para limpiarlas: ICPRx.


    //Int. port 3 = 37 corresponde al bit 5 del segundo registro
ISER1:
```

```
    NVIC->ICPR[1] |= BIT5; //Primero, me aseguro de que no quede
ninguna interrupción residual pendiente para este puerto,

    NVIC->ISER[1] |= BIT5; //y habilito las interrupciones del puerto

    //Int. port 4 = 38 corresponde al bit 6 del segundo registro
ISERx:

    NVIC->ICPR[1] |= BIT6; //Primero, me aseguro de que no quede
ninguna interrupción residual pendiente para este puerto,

    NVIC->ISER[1] |= BIT6; //y habilito las interrupciones del puerto

    //Int. port 5 = 39 corresponde al bit 7 del segundo registro
ISERx:

    NVIC->ICPR[1] |= BIT7; //Primero, me aseguro de que no quede
ninguna interrupcion residual pendiente para este puerto,

    NVIC->ISER[1] |= BIT7; //y habilito las interrupciones del puerto


    __enable_interrupt(); //Habilitamos las interrupciones a nivel
global del micro.

}


/**********************************************************************
*****

 * INICIALIZACIÓN DE LA PANTALLA LCD.

 *

 * Sin datos de entrada

 *

 * Sin datos de salida

 *

**********************************************************************
****/

void init_LCD(void)

{

    halLcdInit(); //Inicializar y configurar la pantallita
```

```
    halLcdClearScreenBkg(); //Borrar la pantalla, rellenando con el
color de fondo
}

/**********************************************************************
*****
 * BORRAR LINEA
 *
 * Datos de entrada: Linea, indica la linea a borrar
 *
 * Sin datos de salida
 *
**********************************************************************
****/
void borrar(uint8_t Linea)
{
    halLcdPrintLine(borrado, Linea, NORMAL_TEXT); //escribimos una
línea en blanco
}

/**********************************************************************
*****
 * ESCRIBIR LINEA
 *
 * Datos de entrada: Linea, indica la linea del LCD donde escribir
 *                   String, la cadena de caracteres que vamos a
escribir
 *
 * Sin datos de salida
 *
```

```
*********************************************************************
****/
void escribir(char String[], uint8_t Linea)


{
    halLcdPrintLine(String, Linea, NORMAL_TEXT); //Enviamos la String
al LCD, sobreescribiendo la Línea indicada.
}


/*********************************************************************
*****
 * INICIALIZACIÓN DE LOS BOTONES & LEDS DEL BOOSTERPACK MK II.
 *
 * Sin datos de entrada
 *
 * Sin datos de salida
 *

*********************************************************************
****/
void init_botons(void)
{
    //Configuramos botones y leds
    //**************************


    //Leds RGB del MK II:
      P2DIR |= 0x50;  //Pines P2.4 (G), 2.6 (R) como salidas Led (RGB)
      P5DIR |= 0x40;  //Pin P5.6 (B)como salida Led (RGB)
      P2OUT &= 0xAF;  //Inicializamos Led RGB a 0 (apagados)
      P5OUT &= ~0x40; //Inicializamos Led RGB a 0 (apagados)
```

```
    //Boton S1 del MK II:
      P5SEL0 &= ~0x02;   //Pin P5.1 como I/O digital,
      P5SEL1 &= ~0x02;   //Pin P5.1 como I/O digital,
      P5DIR &= ~0x02; //Pin P5.1 como entrada
      P5IES &= ~0x02;   // con transicion L->H
      P5IE |= 0x02;     //Interrupciones activadas en P5.1,
      P5IFG = 0;    //Limpiamos todos los flags de las interrupciones
del puerto 5
      //P5REN: Ya hay una resistencia de pullup en la placa MK II


    //Boton S2 del MK II:
      P3SEL0 &= ~0x20;   //Pin P3.5 como I/O digital,
      P3SEL1 &= ~0x20;   //Pin P3.5 como I/O digital,
      P3DIR &= ~0x20; //Pin P3.5 como entrada
      P3IES &= ~0x20;   // con transicion L->H
      P3IE |= 0x20;   //Interrupciones activadas en P3.5
      P3IFG = 0;  //Limpiamos todos los flags de las interrupciones
del puerto 3
      //P3REN: Ya hay una resistencia de pullup en la placa MK II


    //Configuramos los GPIOs del joystick del MK II:
      P4DIR &= ~(BIT1 + BIT5 + BIT7);   //Pines P4.1, 4.5 y 4.7 como
entrades,
      P4SEL0 &= ~(BIT1 + BIT5 + BIT7);  //Pines P4.1, 4.5 y 4.7 como
I/O digitales,
      P4SEL1 &= ~(BIT1 + BIT5 + BIT7);
      P4REN |= BIT1 + BIT5 + BIT7;  //con resistencia activada
      P4OUT |= BIT1 + BIT5 + BIT7;  // de pull-up
      P4IE |= BIT1 + BIT5 + BIT7;   //Interrupciones activadas en
P4.1, 4.5 y 4.7,
      P4IES &= ~(BIT1 + BIT5 + BIT7);   //las interrupciones se
generaran con transicion L->H
```

```
      P4IFG = 0;    //Limpiamos todos los flags de las interrupciones
del puerto 4


      P5DIR &= ~(BIT4 + BIT5);  //Pines P5.4 y 5.5 como entrades,

      P5SEL0 &= ~(BIT4 + BIT5); //Pines P5.4 y 5.5 como I/O digitales,

      P5SEL1 &= ~(BIT4 + BIT5);

      P5IE |= BIT4 + BIT5;  //Interrupciones activadas en 5.4 y 5.5,

      P5IES &= ~(BIT4 + BIT5);  //las interrupciones se generarán con
transición L->H

      P5IFG = 0;    //Limpiamos todos los flags de las interrupciones
del puerto 4

    // - Ya hay una resistencia de pullup en la placa MK II

}


/**********************************************************************
*****

 * DELAY - A CONFIGURAR POR EL ALUMNO - con bucle while

 *

 * Datos de entrada: Tiempo de retraso. 1 segundo equivale a un
retraso de 1000000 (aprox)

 *

 * Sin datos de salida

 *


**********************************************************************
****/

void delay_t (uint32_t temps)

{

   volatile uint32_t i;


   /*************************

     * A RELLENAR POR EL ALUMNO
```

```
     *************************/
   i = 0; // Inicializar la variable i que utilizamos como contador
   do {
       i++; // Incrementar el contador
   } while(i < temps); // Salir del bucle cuando el contador llegue al
tiempo

}

/**********************************************************************
********
 * CONFIGURACIÓN DEL PUERTO 7. A REALIZAR POR EL ALUMNO
 *
 * Sin datos de entrada
 *
 * Sin datos de salida
 *

**********************************************************************
******/
void config_P7_LEDS (void)
{

    //A RELLENAR POR EL ALUMNO
    P7SEL0 = 0x00;
    P7SEL1 = 0x00; //Configurar todos los pines del puerto 7 como GPIO
    P7DIR |= 0xFF; //Configurar todos los pines del puerto 7 como
salida
    P7OUT &= ~(0xFF); //Apagar los LEDs
}
```

```
void main(void)
{

    WDTCTL = WDTPW+WDTHOLD;          // Paramos el watchdog timer

    //Inicializaciones:
    init_ucs_24MHz();       //Ajustes del clock (Unified Clock System)
    init_botons();          //Configuramos botones y leds
    config_P7_LEDS();
    init_interrupciones();  //Configurar y activar las interrupciones
de los botones
    init_LCD();             // Inicializamos la pantalla

    halLcdPrintLine(saludo,linea, INVERT_TEXT); //escribimos saludo en
la primera línea
    linea++;                    //Aumentamos el valor de línea y con
ello pasamos a la línea siguiente
    int i = 0;
    int tiempo = 100000;
    //Bucle principal (infinito):
    do
    {

    if (estado_anterior != estado)          // Dependiendo del valor
del estado se encenderá un LED u otro.
    {
        sprintf(cadena,"Estado %02d", estado);  // Guardamos en cadena
la siguiente frase: Estado "valor del estado",
                                                //con formato decimal,
2 cifras, rellenando con 0 a la izquierda.
        escribir(cadena,linea); // Escribimos la cadena al LCD
```

```
        estado_anterior = estado; // Actualizamos el valor de
estado_anterior, para que no esté siempre escribiendo.



        /*********************************************************+
            A RELLENAR POR EL ALUMNO BLOQUE switch ... case
        Para gestionar las acciones:
        Boton S1, estado = 1
        Boton S2, estado = 2
        Joystick left, estado = 3
        Joystick right, estado = 4
        Joystick up, estado = 5
        Joystick down, estado = 6
        Joystick center, estado = 7
        *********************************************************/
        switch(estado){
                    case Button_S1:
                        P2OUT |= 0x50;
                        P5OUT |= 0x40;//Encender los tres RGB
                        break;
                    case Button_S2:
                        P2OUT &= ~(0x50);
                        P5OUT &= ~(0x40);//Apagar los tres RGB
                        break;
                    case Jstick_Left:
                        P2OUT |= 0x50;
                        P5OUT |= 0x40;//Encender los tres RGB
                        for(i = 7; i>=0; i--){
                            P7OUT = 0x00;
```

```
                            if(i == 0){
                                P7OUT |= 0x01;
                            }else if(i == 1){
                                P7OUT |= 0x02;
                            }else if(i == 2){
                                P7OUT |= 0x04;
                            }else if(i == 3){
                                P7OUT |= 0x08;
                            }else if(i == 4){
                                P7OUT |= 0x10;
                            }else if(i == 5){
                                P7OUT |= 0x20;
                            }else if(i == 6){
                                P7OUT |= 0x40;
                            }else if(i == 7){
                                P7OUT |= 0x80;
                            }
                            delay_t(tiempo);//Encender los LEDs de
derecha a izquierda
                        }
                        P7OUT = 0x00;
                        break;
                    case Jstick_Right:
                        P2OUT |= 0x50;
                        P5OUT &= ~(0x40);//Apagar el LED B i encender
el R y G
                        for(i = 0; i<8; i++){
                            P7OUT = 0x00;
                            if(i == 0){
                                P7OUT |= 0x01;
```

```
                        }else if(i == 1){
                            P7OUT |= 0x02;
                        }else if(i == 2){
                            P7OUT |= 0x04;
                        }else if(i == 3){
                            P7OUT |= 0x08;
                        }else if(i == 4){
                            P7OUT |= 0x10;
                        }else if(i == 5){
                            P7OUT |= 0x20;
                        }else if(i == 6){
                            P7OUT |= 0x40;
                        }else if(i == 7){
                            P7OUT |= 0x80;
                        }
                        delay_t(tiempo);//Encender los LEDs de
izquierda a derecha
                    }
                    P7OUT = 0x00;
                    break;
                case Jstick_Up:
                    P2OUT |= 0x40;
                    P2OUT &= ~(0x20);
                    P5OUT |= 0x40;  //Apagar el LED G i encender
el R y B
                    tiempo += 10000;//Aumenta el tiempo que están
encendidos los LEDs
                    borrar(linea+2);
                    break;
                case Jstick_Down:
                    P2OUT |= 0x20;
```

```
                        P2OUT &= ~(0x40);
                        P5OUT |= 0x40;//Apagar el LED R i encender el
G y B
                        if (tiempo >= 30000){ // Definimos un tiempo
mínimo para que el tiempo no sea negativo
                            tiempo -= 10000; //Disminuye el tiempo que
están encendidos los LEDs
                        }else{
                            sprintf(limite,"Temps limit"); //Escribir
en la cadena
                            escribir(limite, linea+2); // Escribimos
un aviso en la pantalla LCD que hemos llegado al tiempo límite
                        }
                        break;
                    case Jstick_Center:
                        P2OUT ^= 0x50;
                        P5OUT ^= 0x40;//Invierte el estado de los LEDs
RGB
                        break;
                    }
    }
    sprintf(limite,"Temps: %02d", tiempo); // Escribir en la cadena
    escribir(limite,linea+1); // Se muestra en la pantalla LCD el
tiempo actual que tarda los LEDs en encenderse

    }while(1); //Condición para que el bucle sea infinito
}



/*********************************************************************
*****
 * RUTINAS DE GESTIÓN DE LOS BOTONES:
```

```
 * Mediante estas rutinas, se detectará qué botón se ha pulsado
 *
 * Sin Datos de entrada
 *
 * Sin datos de salida
 *
 * Actualizar el valor de la variable global estado
 *
*************************************************************************
****/


//ISR para las interrupciones del puerto 3:
void PORT3_IRQHandler(void){//interrupcion del pulsador S2
    uint8_t flag = P3IV; //guardamos el vector de interrupciones. De
paso, al acceder a este vector, se limpia automáticamente.
    P3IE &= 0xDF;  //interrupciones del botón S2 en port 3
desactivadas
    estado_anterior=0;
    switch(flag){
        case 0x0C:
            estado = Button_S2;
            break;
    }//Guardamos el estado de la interrupción


    /*********************************************************+
        A RELLENAR POR EL ALUMNO
    Para gestionar los estados:
    Boton S1, estado = 1
    Boton S2, estado = 2
    Joystick left, estado = 3
```

```
    Joystick right, estado = 4
    Joystick up, estado = 5
    Joystick down, estado = 6
    Joystick center, estado = 7
    ************************************************************/



    P3IE |= 0x20;   //interrupciones S2 en port 3 reactivadas
}

//ISR para las interrupciones del puerto 4:
void PORT4_IRQHandler(void){  //interrupción de los botones. Actualiza
el valor de la variable global estado.
    uint8_t flag = P4IV; //guardamos el vector de interrupciones. De
paso, al acceder a este vector, se limpia automáticamente.
    P4IE &= 0x5D;   //interrupciones Joystick en port 4 desactivadas
    estado_anterior=0;
    switch(flag){
            case 0x04:
                estado = Jstick_Center;
                break;
            case 0x0C:
                estado = Jstick_Right;
                break;
            case 0x10:
                estado = Jstick_Left;
                break;
        }//Guardamos el estado de la interrupción

    /************************************************************+
```

```
        A RELLENAR POR EL ALUMNO BLOQUE switch ... case
    Para gestionar los estados:
    Boton S1, estado = 1
    Boton S2, estado = 2
    Joystick left, estado = 3
    Joystick right, estado = 4
    Joystick up, estado = 5
    Joystick down, estado = 6
    Joystick center, estado = 7
    ************************************************************/




    /*********************************************
     * HASTA AQUI BLOQUE CASE
     *********************************************/

    P4IE |= 0xA2;   //interrupciones Joystick en port 4 reactivadas
}

//ISR para las interrupciones del puerto 5:
void PORT5_IRQHandler(void){  //interrupción de los botones. Actualiza
el valor de la variable global estado.
    uint8_t flag = P5IV; //guardamos el vector de interrupciones. De
paso, al acceder a este vector, se limpia automáticamente.
    P5IE &= 0xCD;   //interrupciones Joystick y S1 en port 5
desactivadas
    estado_anterior=0;
    switch(flag){
            case 0x04:
                estado = Button_S1;
```

```
                break;
            case 0x0C:
                estado = Jstick_Down;
                break;
            case 0x0A:
                estado = Jstick_Up;
                break;
            }//Guardamos el estado de la interrupción

    /*********************************************************+
        A RELLENAR POR EL ALUMNO BLOQUE switch ... case
    Para gestionar los estados:
    Boton S1, estado = 1
    Boton S2, estado = 2
    Joystick left, estado = 3
    Joystick right, estado = 4
    Joystick up, estado = 5
    Joystick down, estado = 6
    Joystick center, estado = 7
    *********************************************************/




    /*********************************************
     * HASTA AQUI BLOQUE CASE
     *********************************************/


    P5IE |= 0x32;   //interrupciones Joystick y S1 en port 5
reactivadas
}
```

3. Diagrama de Flujo