Práctica 2 – Entrada y Salida en C

Jaime Arana
Manuel Ferrero
3ºA

Manejo de puertos:

```c
#include <xc.h>

int main(void)
{
    // se ponen los LEDs como salida
    TRISC = ~0x0F;

    // los LEDs son activos a nivel bajo
    LATC = ~0x0F;

    while(1)
        ;
    return 0;
}
```

En este apartado se pide encender los 4 LEDs conectados a los bits menos significativos del puerto C, siendo los bits RC0 – RC3.

En este primer apartado se ha optado por no utilizar máscaras y escribir el puerto C directamente. No obstante, para los siguientes aparatados si se va a trabajar con máscaras.

Diseño 1:

Encender LED RC2

```c
#include <xc.h>

#define BIT_LED0 0
#define BIT_LED1 1
#define BIT_LED2 2
#define BIT_LED3 3

int main(void)
{
    // LEDs como salidas
    TRISC = ~0x0F;

    // variable de ayuda ya que no se puede escribir
    // el puerto C varias veces seguidas
    puerto = 0x0000;

    //encendidos LED 2
    puerto |= (1 << BIT_LED0);
    puerto |= (1 << BIT_LED1);
    puerto &= ~(1 << BIT_LED2);
    puerto |= (1 << BIT_LED3);
    LATC = puerto;

    while(1)
        ;
    return 0;
}
```

Encender LED RC2 Y RC3

```c
#include <xc.h>

#define BIT_LED0 0
#define BIT_LED1 1
#define BIT_LED2 2
#define BIT_LED3 3

int main(void)
{
    // LEDs como salidas
    TRISC = ~0x0F;

    // variable de ayuda ya que no se puede escribir
    // el puerto C varias veces seguidas
    puerto = 0x0000;

    //encendidos LED 2 y LED 3
    puerto |= (1 << BIT_LED0);
    puerto |= (1 << BIT_LED1);
    puerto &= ~(1 << BIT_LED2);
    puerto &= ~(1 << BIT_LED3);
    LATC = puerto;

    while(1)
        ;
    return 0;
}
```

Manejo del pulsador:

```c
#include <xc.h>

#define PIN_PULSADOR 5

int main(void)
{
    int pulsador;

    // LEDs como salida
    TRISC = ~0x0F;

    // LEDs empiezan apagados
    LATC = 0x0F;

    // pulsador como entrada
    TRISB = 0xFF;

    while(1){
        // Se lee el estado del pulsador
        pulsador = ( PORTB >> PIN_PULSADOR ) & 1;
        // Cuando se pulsa el pulsador se enciende el LED 0, al dejar de pulsar se apagan
        if(pulsador == 0){
            LATC &= ~1;
        }else{
            LATC |= 1;
        }
    }

    return 0;
}
```

Detección de flancos:

```c
#include <xc.h>

#define BIT_PULSADOR 5

int main(void)
{
    int pulsador_ant, pulsador_act;

    TRISC = ~0x0F;
    LATC = 0x0F;
    TRISB |= (1 << BIT_PULSADOR);

    //leemos el estado del pulsador
    //con PORTB --> leemos el valor del puerto
    //>> BIT_PULSADOR --> leemos valor en el campo del pulsador
    pulsador_ant = (PORTB >> BIT_PULSADOR) & 1;
    int contador = 0;

    while(1){
        pulsador_act = (PORTB >> BIT_PULSADOR) & 1;

        //si el pulsador == 0 --> está pulsado
        if((pulsador_act != pulsador_ant) && (pulsador_act == 0)){
            if(contador > 15) { // vuelve a empezar
                contador = 0;
            }
            contador = contador + 1;
            //escribimos en el puerto el valor del contador para que se enciendan los LEDs
            LATC = ~contador;
        }
        pulsador_ant = pulsador_act;
    }

    return 0;
}
```

Diseño 2:

```c
#include <xc.h>

#define BIT_PULSADOR 5
#define BIT_LED0 0
#define BIT_LED1 1
#define BIT_LED2 2
#define BIT_LED3 3

int main(void)
{
    int pulsador_ant, pulsador_act;

    TRISC = ~0x0F;
    LATC = 0x0F;
    TRISB |= (1 << BIT_PULSADOR);

    //leemos el estado del pulsador
    //con PORTB --> leemos el valor del puerto
    //>> BIT_PULSADOR --> leemos valor en el campo del pulsador
    pulsador_ant = (PORTB >> BIT_PULSADOR) & 1;
    int contador = 0;

    int puerto = 0x0000;
    //encendemos solo el LED 2
    puerto &= ~(1 << BIT_LED0);
    puerto |= (1 << BIT_LED1);
    puerto |= (1 << BIT_LED2);
    puerto |= (1 << BIT_LED3);
    LATC = puerto;

    while(1){
        pulsador_act = (PORTB >> BIT_PULSADOR) & 1;

        //si el pulsador == 0 --> está pulsado
        if((pulsador_act != pulsador_ant) && (pulsador_act == 0)){
            //ponemos todos los campos a 0 --> se encienden los LEDs
            contador = contador + 1;

            if(contador < 5){
                //desplazamos un 0 a la derecha y ponemos un 0 en la posicion anterior
                LATC &= ~(1 << contador);
                //tenemos que hacer un
                asm(" NOP");
                LATC |= (1 << (contador - 1));
            }
            //si llegamos al final de la secuencia resetear al valor incial
            if(contador == 4) {
                contador = 0;
                LATC = 14;
            }
        }

        pulsador_ant = pulsador_act;
    }

    return 0;
}
```