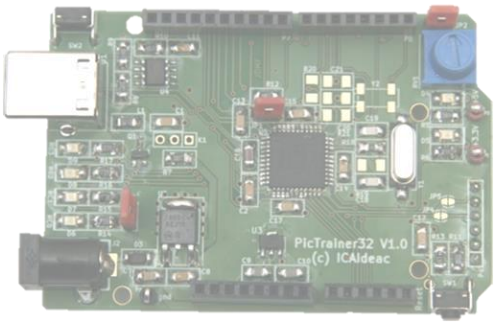
Microprocesadores

Práctica 2 – Entrada y salida en C

Francisco Martín Martínez



Jorge Calvar, Fernando Santana
5 de febrero de 2021

Contenido

Introducción

En esta práctica se usará la tarjeta montada en la práctica 1 con el objetivo de familiarizarse con los puertos de entrada y salida digital para interactuar con los LEDs y el pulsador de la placa.

Manejo de puertos

En primer lugar, se ha diseñado un programa que enciende los cuatro LEDs conectado en el puerto C (RC3 a RC0). Para ello simplemente se han definidos estos bits como salida y se ha configurado su valor a 0, ya que los LEDs están activos a nivel bajo.

```
/*
 * File:   main2.c
 * Author: Fernando y Jorge
 *
 * Created on 05 February 2021, 11:00
 */

#include <xc.h>

int main(void) {

    TRISC = 0xFFF0; // 4 LEDs as output 11111111110000
    LATC &= ~0xF; // LEDs on

    while (1);

    return 0;
}
```

Diseño 1

En este apartado, se ha modificado el programa anterior para:

1. Encender sólo el LED conectado a RC2.

Solo se han configurado los últimos cuatro bits de LATC para que tomen el valor 1011 en vez de 0000.

```
/*
 * File:   main3_1.c
 * Author: Fernando y Jorge
 *
 * Created on 05 February 2021, 11:02
 */

# include <xc.h>

int main(void)
{
        TRISC = 0xFFF0; // 4 LEDs as output
        LATC &= ~0x4; // RC3 1111111011
        LATC |= 0xB;
```

```
        while (1);

        return 0;
}
```

2. Encender sólo los LEDS RC3 y RC2.

En este caso se han configurado LATC para que sus últimos cuatro bits tomen el valor 0011.

```
/*
 * File:   main3_2.c
 * Author: Fernando y Jorge
 *
 * Created on 05 February 2021, 11:03
 */

#include <xc.h>

int main(void)
{
        TRISC = 0xFFF0; // 4 LEDs as output
        LATC &= ~0xC; // RC3,RC2  111110011
        LATC |= 0x3;

        while (1);

        return 0;
}
```

Manejo del pulsador

El pulsador está conectado al pin 5 del puerto B (RB5).

Hemos configurado el programa realizando las modificaciones que se mencionan en el enunciado:

- Se han definido los pines RC3 a RC0 como salidas, el resto del puerto como entradas.
- Se han apagado inicialmente todos los LEDs de la placa.
- Se han definido los pines del puerto B como entradas.
- Se ha explicado en el comentario la funcionalidad el programa.

También hemos modificado el programa para que se enciendan todos los LEDs en vez uno solo al presionar el pulsador.

```
/*
 * File:   main4.c
 * Author: Fernando y Jorge
 *
 * Created on 05 February 2021, 11:04
 */

#include <xc.h>

#define PIN_PULSADOR 5
```

```
int main(void)
{
        int pulsador;

        TRISC = 0xFFF0; // 4 LEDs as output
        LATC |= 0xF; // LEDs as OFF

        TRISB = 0xFFFF; // Button is input

        while (1) {

                // We read button state
                pulsador = (PORTB >> PIN_PULSADOR) & 1;

        // We turn LEDs on when button is pressed and off when not
                if(pulsador == 0){
                        LATC &= ~0xF;
                }else{
                        LATC |= 0xF;
                }

        }
}
```

Detección de flancos

En el siguiente programa, se ha creado un detector de flancos a partir del código propuesto en el enunciado. Este programa es increíblemente útil porque es típico querer realizar una única acción al pulsar el botón y no estar realizándola indefinidamente durante la duración de la pulsación.

Por ejemplo, en este caso cada vez que se presiona el pulsador incrementa un contador que se muestra en binario en los leds. Por tanto, solo debe haber una unidad de incremento entre cada pulsación. Para ello, es necesaria la detección del flanco.

```
/*
 * File:   main5.c
 * Author: Fernando y Jorge
 *
 * Created on 05 February 2021, 11:05
 */

#include <xc.h>

#define PIN_PULSADOR 5

int main ( void )
{
        int pulsador_ant, pulsador_act;

        TRISC = 0xFFF0; // 4 LEDs as output
        LATC |= 0xF; // LEDs as OFF

        TRISB = 0xFFFF; // Button RB5 is input 1111111111011111

        pulsador_ant = (PORTB >> PIN_PULSADOR) & 1;
```

```
        //1111111111X10000  -> 000011111111111X -> 000000000000000X

        while (1) {

                // We read buttons state
                pulsador_act = (PORTB >> PIN_PULSADOR) & 1;

                if((pulsador_act != pulsador_ant) && (pulsador_act==0))
{

            // Falling edge in button, which means it has just been
pressed.

            // We will show the number of times button has been
pressed with the LEDs
            // When all LEDS are on, we will turn all off.
            if((LATC & 0xF) == 0x0){
                LATC |= 0xF;
            }else{
                LATC--;
            }
                }

                pulsador_ant = pulsador_act;

        }

}
```

Diseño 2

Por último, se ha diseñado un programa que enciende uno de los cuatro LED y cada vez que se presiona el pulsador el LED encendido cambia siguiendo un orden de rotación. Para ello, simplemente se ha creado un entero contador que marca el número de posiciones que se desplaza un 1. Esta variable se ha negado, ya que los LEDs son activos a nivel bajo.

```
/*
 * File:   main6.c
 * Author: Fernando y Jorge
 *
 * Created on 05 February 2021, 12:39
 */

#include <xc.h>

#define PIN_PULSADOR 5

int main ( void )
{
        int pulsador_ant, pulsador_act;
    int contador = 0;

        TRISC = 0xFFF0 ; // 4 LEDs as output
        LATC |= 0xE ; // LEDs are OFF except first one
    LATC &= ~1; // We set first LED ON
```

```
        TRISB = 0xFFFF; // Button RB5 is input 1111111111011111

        pulsador_ant = (PORTB >> PIN_PULSADOR) & 1;

        //1111111111X10000  -> 000011111111111X -> 000000000000000X

        while (1) {

                // We read buttons state
                pulsador_act = (PORTB >> PIN_PULSADOR) & 1;

                if((pulsador_act != pulsador_ant) && (pulsador_act==0))
{

            // Falling edge in button, which means it has just been
pressed.
            // Each time the button is pressed the one LED ON will
change in order

            if(contador == 3)
                contador = 0;
            else
                contador++;

            LATC = (~(1 << contador)) & (0xF);

                }

                pulsador_ant = pulsador_act;

        }

}
```

Conclusión

En esta práctica hemos aprendido a:

- Manejar los puertos de entrada y salida digital del PIC32.
- Realizar programas sencillos que interactúen con el hardware.
- Depurar utilizando el debugger para encontrar sus fallos.
- Comprobar que los distintos componentes se encuentran bien soldados utilizando el multímetro para comprobar continuidad

El obstáculo principal que nos hemos encontrado en la práctica es que, al ser la primera, no sabíamos si los errores se debían al software o al hardware. Hemos cometido un pequeño error de código y antes de descubrirlo hemos estado revisado todas las conexiones de la PCB con un multímetro pensando que se trataba de un problema de soldadura. Sin embargo, solo era una letra mal escrita en el código. Esto nos ha enseñado la importancia de la depuración.