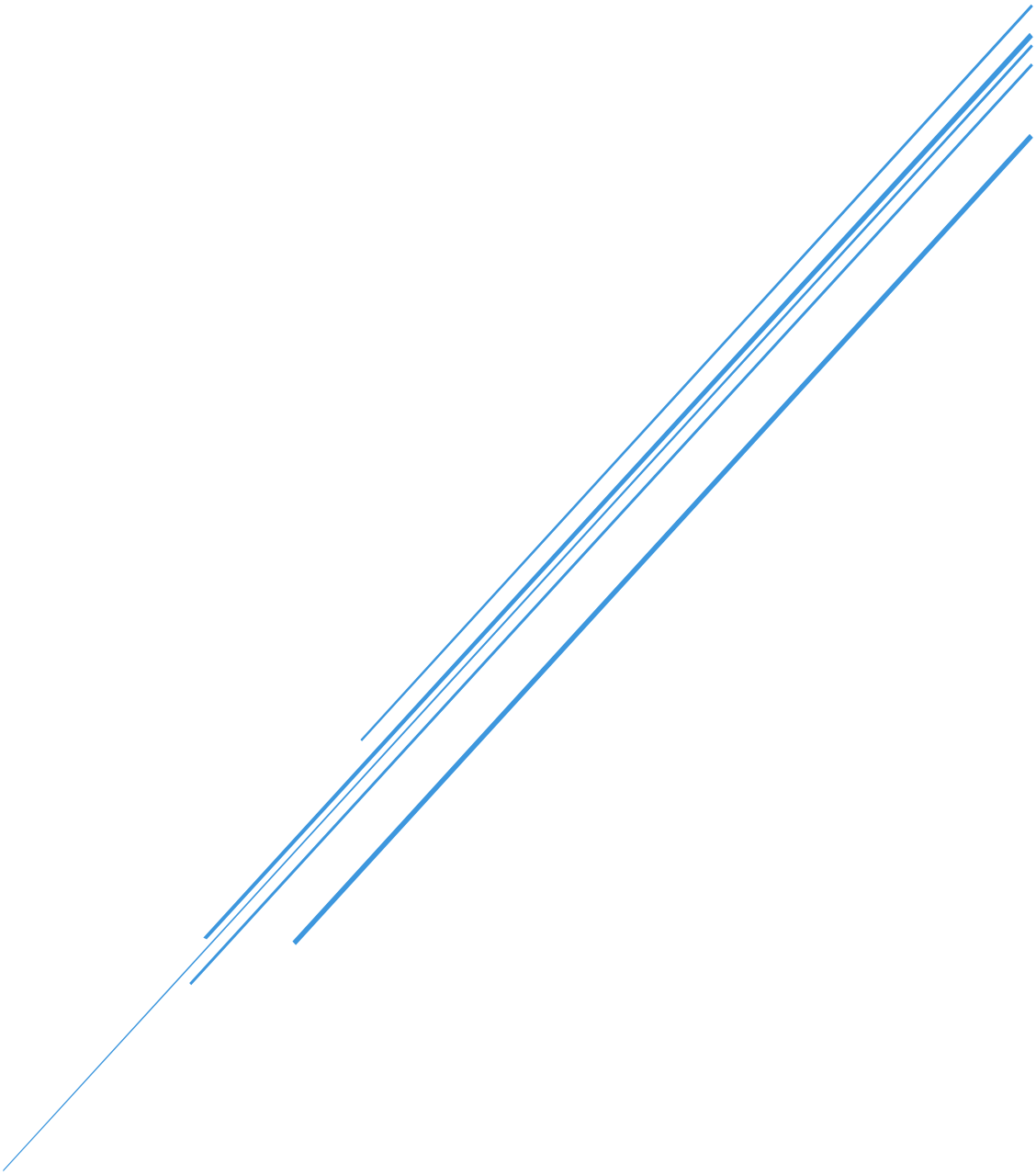
PRÁCTICA 1

Sistemas Empotrados

Luis Díaz del Río Delgado



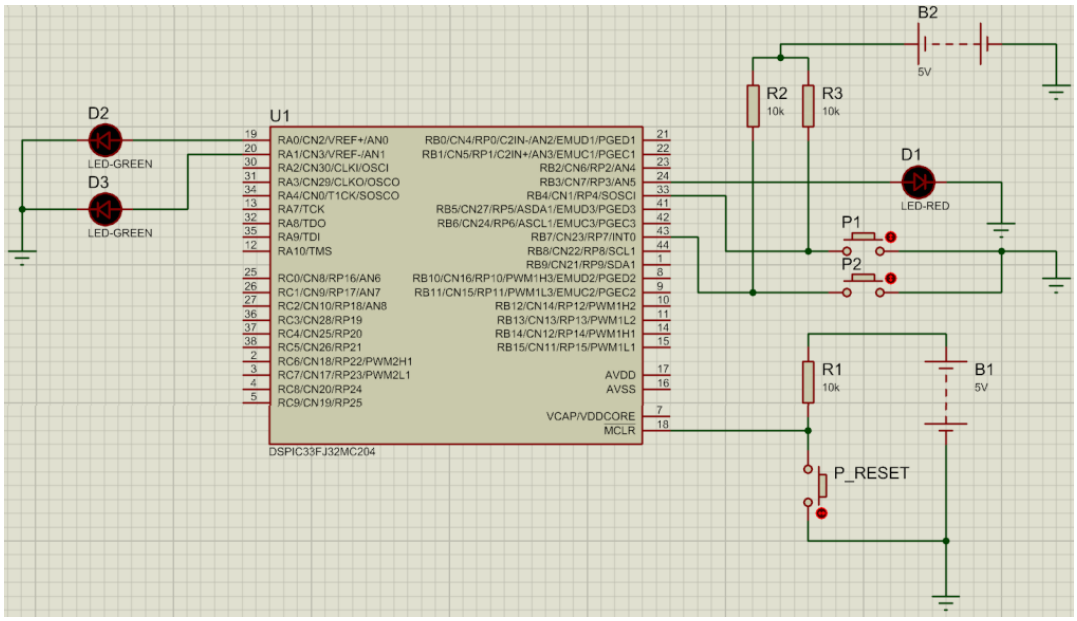
Universidad Nebrija
2021/22

Tareas

Se pide probar las siguientes tareas:

1. Lograr que el diodo LED D1 parpadee a un ritmo de 1Hz al ejecutar el programa. Considerad una frecuencia de la CPU de 4Mhz.
2. Por defecto los diodos LED D2 y D3 deben permanecer apagados. En el momento de pulsar el botón P1, los diodos D2 y D3 deben encenderse durante el tiempo que P1 permanezca pulsado. En caso contrario deben permanecer apagados.
3. Si se mantiene presionado el pulsador P2, el diodo D3 debe parpadear a un ritmo de 2Hz. En caso contrario, el diodo D3 debe permanecer apagado. A su vez, se debe cumplir en todo momento el requisito del punto 1. (Considerad una frecuencia de CPU de 4Mhz).

Además se presenta el siguiente esquemático:



Definiciones/Macros

Lo primero que hago, tras la configuración de bits vista en clase y durante la sesión de prácticas, es crear algunas macros. La finalidad es darle claridad al código.

```
#define LED_RED1 LATBbits.LATB3
#define LED_RED2 LATAbits.LATA0
#define LED_RED3 LATAbits.LATA1
#define LED_ON_state 1
#define LED_OFF_state 0
```

Teniendo en cuenta la imagen adjunta con el enunciado de la práctica, y empezando por "LED_RED1", corresponden a los LEDs D1, D2 y D3.

Por otro lado están los estados, que serán utilizados para encender y apagar los LEDs.

Funciones

En este caso, únicamente creo una función, llamada "delay_ms" y que tiene como finalidad crear un retardo en la señal de reloj.

```
void delay_ms(unsigned long time_ms){
    unsigned long i;
    for(i=0; i < time_ms*100; i++){
        asm("NOP");
    }
}
```

Para conseguir esto se utiliza un bucle "for" el cual va desde -0- a un tiempo definido por el usuario, que pasamos a la función mediante un parámetro (time_ms), multiplicado en este caso por cien.

Por cada iteración del bucle se ejecutará la instrucción en ensamblador "NOP". Esta instrucción al ser ejecutada no hace nada, por lo que conseguimos perder tantos ciclos de reloj como instrucciones "NOP" se ejecuten, creando el retardo.

Main

Para este ejercicio en concreto, se nos pide que la frecuencia de la CPU sea de 4Mhz. Nuestro cristal de cuarzo funciona a 8Mhz, así que lo primero será cambiar la frecuencia final.

```
//Fosc = Fpri*M/(N1*N2) => 8MHz*4/(2*2) = 32/4 = 8;
//Fcpu = Fosc/2 = 8MHz/2 = 4MHz;
PLLFBD = 2;
CLKDIVbits.PLLPRE = 0;
CLKDIVbits.PLLPOST = 0;
while(OSCCONbits.LOCK != 1);
```

El código sale directamente de las formularas dadas en los apuntes para configurar el PLL.

Seguidamente, elegimos si vamos a trabajar como analógico o digital. Esta línea de código viene directamente de la sesión de prácticas.

```
//Control de pin para trabajar como analógico/digital
AD1PCFGL = 0xFFFF;
```

Antes de definir el funcionamiento de cada componente, debemos elegir la direccionalidad de los pines.

```
//Control de direccinalidad de los pines de mi proyecto
TRISBbits.TRISB3 = 0;    //Pin B3 -> D1 output.
TRISBbits.TRISB4 = 1;    //Pin B4 -> P1 input.
TRISBbits.TRISB7 = 1;    //Pin B7 -> P2 input.

TRISAbits.TRISA0 = 0;    //Pin A0 -> D2 output
TRISAbits.TRISA1 = 0;    //Pin A1 -> D3 output
```

Los pines RB3, RA0 y RA1 serán pines de salida que irán conectados a los LED. Por otra parte los pines RB4 y RB7 serán pines de entrada conectados a dos botones (P1 y P2).

Para finalizar, debemos definir el comportamiento de los distintos componentes.

```
    int delayCounter = 0;
    while(1){
        if(PORTBbits.RB7 == 0){
            if(PORTBbits.RB4 == 0){
                LED_RED2 = LED_ON_state;
            }else{
                LED_RED2 = LED_OFF_state;
            }
            if(delayCounter == 2){
                LED_RED3 = !PORTAbits.RA1;
                delayCounter = 0;
            }
        }else{
            if(PORTBbits.RB4 == 0){
                LED_RED2 = LED_ON_state;
                LED_RED3 = LED_ON_state;
            }else{
                LED_RED2 = LED_OFF_state;
                LED_RED3 = LED_OFF_state;
            }
            delayCounter = 0;
        }
        LED_RED1 = !PORTBbits.RB3;
        delayCounter = delayCounter + 1 ;
        delay_ms(1000);
    }
    return 0;
}
```

Para empezar, y fuera del bucle infinito “while(1)” definimos una variable de tipo entero inicializada a 0 (delayCounter). Esta variable es un contador utilizado para el “delay” de D3.

Una vez dentro del bucle, empezamos comprobando el estado del botón conectado al pin RB7 (P2). En el caso de que el botón (P2) se mantenga pulsado, se harán dos cosas:

La primera será comprobar el estado del botón conectado al pin RB4 (P1). Esto se hace para que, si se da un caso en el que tanto el botón P2 y el botón P1 se mantengan pulsados al mismo tiempo, se cumpla cada uno de los comportamientos asociados a cada botón de forma simultánea (Los dos LED se encenderán al mismo tiempo, aunque el LED D3 cambiará de estado a una frecuencia de 2Hz). Si el botón P1 dejara de ser pulsado, pero P2 siguiera siendo pulsado, únicamente se apagaría el LED D2 mientras que el LED D3 seguiría parpadeando.

La segunda será cambiar el estado del LED D3 a la frecuencia de 2Hz como pide el enunciado. Aquí hacemos uso del contador. Como el LED D1 debe cambiar su estado constantemente desde la ejecución del programa a una frecuencia de 1Hz y la frecuencia pedida para D3 es el doble de la necesario en D1, contamos dos iteraciones del bucle, esto será equivalente a dos veces el “delay” de D1, es decir, 2Hz. Cuando el contador llega a 2, cambiamos el estado del LED y ponemos a 0 el contador.

En el cao de que el botón P2 no se esté pulsando se comprobará el estado del botón P1. Como el botón P2 no está siendo pulsado, tanto el LED D2 como el D3 se mantienen encendidos mientras el botón P1 se mantenga pulsado. En el momento en el que se deje de pulsar, los dos LEDs se quedarán apagados. También, en el caso de que P2 deje de ser pulsado, pondremos a 0 el contador para que no se quede pillado nunca. Algo que sucedió un par de veces durante las pruebas realizadas durante la realización de la práctica.

Una vez terminadas todas las comprobaciones, el LED D1 cambiará su estado contantemente, con un retardo de 1Hz (1s -> 1000ms). También se aumentará el contador utilizado para D3.

Conclusión

Estoy contento con el resultado obtenido en la práctica, así como las cosas aprendidas durante su realización. No me he encontrado con ningún impedimento ni problema durante el desarrollo de la práctica. La función “main()” completa y sin cortes y mi esquemático en Proteus se encuentras más abajo.

Código completo y mi esquemático en Proteus.

Código completo sin las opciones de configuración de bits.

```c
#include "xc.h"

#define LED_RED1 LATBbits.LATB3
#define LED_RED2 LATAbits.LATA0
#define LED_RED3 LATAbits.LATA1
#define LED_ON_state 1
#define LED_OFF_state 0

void delay_ms(unsigned long time_ms){
    unsigned long i;
    for(i=0; i < time_ms*100; i++){
        asm("NOP");
    }
}

int main(void) {
    //Fosc = Fpri*M/(N1*N2) => 8MHz*4/(2*2) = 32/4 = 8;
    //Fcpu = Fosc/2 = 8MHz/2 = 4MHz;
    PLLFBD = 2;
    CLKDIVbits.PLLPRE = 0;
    CLKDIVbits.PLLPOST = 0;
    while(OSCCONbits.LOCK != 1);

    //Control de pin para trabajar como analógico/digital
    AD1PCFGL = 0xFFFF;

    //Control de direccinalidad de los pines de mi proyecto
    TRISBbits.TRISB3 = 0;   //Pin B3 -> D1 output.
    TRISBbits.TRISB4 = 1;   //Pin B4 -> P1 input.
    TRISBbits.TRISB7 = 1;   //Pin B7 -> P2 input.

    TRISAbits.TRISA0 = 0;   //Pin A0 -> D2 output
    TRISAbits.TRISA1 = 0;   //Pin A1 -> D3 output

    int delayCounter = 0;
    while(1){
        if(PORTBbits.RB7 == 0){
            if(PORTBbits.RB4 == 0){
                LED_RED2 = LED_ON_state;
            }else{
                LED_RED2 = LED_OFF_state;
            }
            if(delayCounter == 2){
                LED_RED3 = !PORTAbits.RA1;
                delayCounter = 0;
            }
        }else{
            if(PORTBbits.RB4 == 0){
                LED_RED2 = LED_ON_state;
                LED_RED3 = LED_ON_state;
            }else{
                LED_RED2 = LED_OFF_state;
                LED_RED3 = LED_OFF_state;
            }
            delayCounter = 0;
        }
        LED_RED1 = !PORTBbits.RB3;
        delayCounter = delayCounter + 1 ;
        delay_ms(1000);
    }
    return 0;
}
```

Mi esquemático en Proteus: