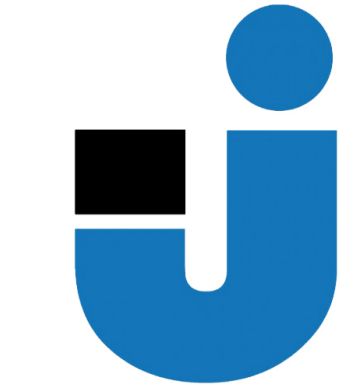

Universidad Nacional
ARTURO JAURETCHE

Programación en Tiempo Real

Trabajo Práctico nº 3

Programación de microcontroladores

Profesor
Diego Encinas

Autores
Cristian Sanabria
Gerónimo Bazán
Emiliano Salvatori

5 de noviembre de 2020

Índice

1. Introducción

En el siguiente informe se detalla lo realizado como parte del laboratorio de la materia **Programación en Tiempo Real** para la **Comisión nº 1**. El presente trabajo se basa en lo solicitado para la Práctica nº 3 sobre **Programación de microcontroladores** realizando una simulación bajo el entorno de simulación ISIS (Proteus) y el compilador de lenguaje C PICC para microcontroladores.

2. Cuestionario

2.1. Ejercicio nº 1

Utilice el entorno ISIS para abrir el diseño del circuito que se encuentra en la carpeta *PIC Doorbell*. En este esquema se observa un microcontrolador PIC 16F84; dos pulsadores conectados a los pines RA0 y RA1 como entradas; dos leds conectados a los pines RB4 y RB5 como salidas.

Configure el entorno y realice un programa C que permita encender y apagar los leds de manera alternada cada 250 milisegundos utilizando la función ¿delay_ms¿. Cuando inicia el programa ambos leds deben estar encendidos. Luego de presionar alguno de los pulsadores deben comenzar a titilar.

2.2. Ejercicio nº 2

Rehaga el ejercicio anterior conservando la misma funcionalidad pero modificando la implementación. Utilice el módulo del *Timer0* con el prescaler para controlar el tiempo de espera y la interrupción para controlar el estado de los leds.

2.3. Ejercicio nº 3

Utilice el entorno ISIS para abrir el diseño del circuito que se encuentra en la carpeta *ADC Example*. En este esquema se observa un microcontrolador PIC 16F877; una resistencia variable (que simula un transductor) conectada a través del pin AN0; dos Latchs 74LS373 (multiplexados) cuyas entradas comparten el puerto B y que son controlados a través de los pines RD6 y RD7 del puerto D; cuatro displays de 7 segmentos hexadecimales conectados a las salidas de los latchs.

Configure el entorno y realice un programa en C que permita capturar valores analógicos realizando "polling" y muestre los valores digitales en los displays de 7 segmentos. Tenga en cuenta que:

- Debe configurar el pin AN0 como entrada analógica y como tensión de referencia el mismo del microcontrolador (VCC).
- Los latchs (74LS373) funcionan de forma multiplexada por lo que requieren la activación/desactivación en el pin LE para que los datos de entrada queden en la salida.

Mostrar el valor decimal en los display de 7 segmentos. Para esto es necesario convertir el valor digital de decimal a BCD de manera que el valor a enviar a cada display quede entre 0 y 9.

3. Resoluciones

3.1. Ejercicio nº 1

A continuació se puede visualizar el código requerido para el Ejercicio nº 1:

```
#include <htc.h>
#define _XTAL_FREQ 1000000  //1MHz

void main() {
        //Configuracion del puerto B de 8 bits;habilita las salidas de los LED
        TRISB = 0b11001111;
        //Configuracion del puerto A de 5 bits;habilita como entradas los pulsadores
        TRISA = 0b00011;
        //enciende LED salida RB4 al comenzar el programa
        RB4 = 1;
        //enciende LED salida RB5 al comenzar el programa
        RB5 = 1;

        while (1) {

                //Se habilita el parpadeo de luces si se aprieta cualquier pulsador
                if (RA0==0 || RA1==0) {

                        //Se ejecuta el parpadeo de LEDS

                        //Prendo LED R3
                        RB4=1;

                        //Se mantiene encendido
                        __delay_ms(250);

                        //Se apaga LED R3
                        RB4=0;
                        //Se prende LED R4
                        RB5=1;
                        //Se mantiene R4 prendido
                        __delay_ms(250);


                        //Se apaga LED R4
                        RB5=0;

                }
                else {
                        //Se mantienen encendido los LED
                        RB5=1;
                        RB4=1;
```

```
                }
        }
}
```

A continuació se puede visualizar el código requerido para el Ejercicio nº 2:

3.2. Ejercicio nº 2

```
#include <htc.h>

#define _XTAL_FREQ 1000000 //1Mhz

void interrupt timer(void) {

        // Invierte bits, necesario para encender y apagar los LEDS
        PORTB = PORTB ^ 0b00110000;

        //El timer cuenta de 12 a 256
        TMR0 = 12;

        //Flag de estado de interrupción por desborde del Timer0
        T0IF = 0;
}

void main () {

        //Config las salidas de los LED
        TRISB = 0b11001111;

        //Config las entradas los pulsadores
        TRISA = 0b00011;

        //Enciende los LED al comenzar el programa
        RB5 = 1;
        RB4 = 1;

        PSA = 0; //bit del prescaler

        OPTION_REG = (OPTION_REG & 0b01111110) | 0b111;

        T0CS = 0; // bit de seleccion de clock

        //Registro de control de interrupciones
        INTCON = 0b10100000;

        //Desactivo las Interrupciones
        GIE = 0;

        while (1) {

                //si cualquiera de los pulsadores estan activos
                if (RA0==0 || RA1==0) {
                        //se activa la interrupcion
```

```
                        GIE = 1;
                } else {
                        //dejo los LED encendidos
                        RB5=1;
                        RB4=1;
                }
        }
}
```

3.3. Ejercicio nº 3

A continuació se puede visualizar el código requerido para el Ejercicio nº 3:

```
#include <stdio.h>
#include <htc.h>

void main() {

        //Configuracion del modulo

        //ADCS:00 -> 1 ciclo de reloj de conversion cada 2 reloj de sistema.
        ADCON0 = 0b01000001;

        //establezco ADFM:1 de conversion
        ADCON1 = 0b10001110;

        //establezco al bit 0 como entrada RA0/AN0
        TRISA = 0b00000001;

        // 0 = sin interrupciones
        ADIE = 0;

        //Configuracion de las salidas

        //Se establecen todos los bits de B como salida de los latch 74LS373
        TRISB = 0b00000000;

        //Se establecen los bits 7 y 6 de D como salida(asociados a los latch)
        TRISD = 0b00000000;

        //Inicio de la Conversion
        GO=1;

        while(1) {

                //si no no termina la conversion
                if(GO==0) {

                        //obtengo el valor convertido
                        unsigned short valor = ADRESH << 8 | ADRESL;

                        //muestro el valor obtenido en la salida
                        RD6=1;
```

```
41                              RD7=0;
42                              PORTB = (valor & 0xff);
43
44                              RD6=0;
45                              RD7=1;
46
47                              //Se resetea la variable para
                                    volver a empezar
48                              GO=1;
49                      }
50              }
51
52      }
```