UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Práctica 8: Programación en C Puertos Paralelos E/S, Puerto Serie

INTEGRANTES DE EQUIPO Y GRUPO DE TEORÍA

López Becerra Ricardo - 420053710 - GRUPO 3
Navarrete Zamora Aldo Yael - 317242409 - GRUPO 4

ASIGNATURA

Laboratorio de Microcomputadoras

GRUPO DE LABORATORIO

8

FECHA DE REALIZACIÓN

18 de mayo del 2023

FECHA DE ENTREGA

25 de mayo de 2023

1. Desarrollo

Realizar las siguientes actividades.

1. Escribir, comentar, compilar el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

void main(){
        while(1){
                output_b(0x01);
                delay_ms(1000);
                output_b(0x00);
                delay_ms(1000);
                } //while
}//main
```

Algoritmo

El código prende y apaga los leds con un retardo de 5 segundos.

Código comentado

```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20M)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

void main(){
    while(1){
      int a = input_a(); // Leemos del puerto A.
      output_b(0xFF); // Prende todos los leds
      delay_ms(5000); // Delay de 5 segundos.
      output_b(0x00); // Apaga los leds
      delay_ms(5000); // Delay de 5 segundos.

   } //while
}//main
```

2. Modificar el programa para que active y desactive todos los bits del puerto B.

Algoritmo

La modificación necesaria fue cambiar el argumento de la función output_b por 0xff para prender todos los bits.

Código comentado

```
#include <16f877.h> //Definiciones del
                    //microcontrolador
#fuses HS,NOPROTECT,
                    //Desactiva los fusible
                    //porque el bootloader
                    //los incluye
#use delay(clock=20M) //configura el reloj
//Evita escribir informacion en el bootloader
#org 0x1F00, 0x1FFF void loader16F877(void) {}

void main(){
   //ciclo infinito
    while(1){
      //output_b(0xFF); //Prende los bits
                        //del puerto b
      //delay_ms(5000); //delay de 5s
      //output_b(0x00); //apaga los leds
      //delay_ms(5000); //Delay de 5s
   } //while
}//main
```

3. Escribir, comentar, compilar el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

int var1;

void main(){
        while(1){
            var1=input_a();
            output_b(var1);
             }//while
           }//main
```

Algoritmo

Un algoritmo bastante sencillo de entender, leemos del puerto A y desplegamos en el puerto B.

Código comentado

```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20M)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

void main(){
    while(1){
      int a = input_a(); // Leer entrada del puerto A
```

```
        output_b(a); // Desplegar entrada en puerto B
    } //while
}//main
```

4. Escribir, comentar, compilar, el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

void main(){
        while(1){
                output_b(0xff); //
                printf(" Todos los bits encendidos \n\r");
                delay_ms(1000);
                output_b(0x00);
                printf(" Todos los leds apagados \n\r");
                delay_ms(1000);
            }//while
        }//main
```

Algoritmo

Este ejercicio es una combinación de los dos anteriores agregando el uso del puerto serie. Para usarlo se hace su configuración al inicio del programa. Para mandar datos se utiliza la función printf como en otras versiones de C.

Código comentado

```
#include <16f877.h>
#fuses HS,NOPROTECT,
#use delay(clock=20000000)
//Configura el puerto serial
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#org 0x1F00, 0x1FFF void loader16F877(void) {}
void main(){
    //ciclo infinito
    while(1){
        output_b(0xff); //Enciende todos los leds
        //Escribe en el puerto serial
        printf("␣Todos␣los␣bits␣encendidos␣\n\r");
        delay_ms(1000); //retardo de 1s
        output_b(0x00); //apaga los leds
        //Escribe en el puerto serial
        printf("Todos␣los␣leds␣apagados␣\n\r");
        delay_ms(1000); // retardo de 1s
    }//while
```

5. Escribir, comentar, compilar, el siguiente programa usando el ambiente del PIC C Compiler y comprobar el funcionamiento.

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#include <lcd.c>

void main() {

        lcd_init();

        while( TRUE ) {
                        lcd_gotoxy(1,1);
                        printf(lcd_putc," UNAM \n ");
                        lcd_gotoxy(1,2);
                        printf(lcd_putc," FI \n ");
                       delay_ms(300);
                       }
    }
```

Algoritmo

Este algoritmo es muy simple, configura el LCD e imprime en dos posiciones distinas FI UNAM con un retraso de 300 milisegundos.

Código comentado

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#include <lcd.c>


void main() {
   lcd_init(); // inicializar la configuracion del lcd
   while( TRUE ) {
      lcd_gotoxy(1,1); // Posicionar en la columna 1 de la fila 1
      printf(lcd_putc," UNAM \n "); //imprimir en pantalla LCD
      lcd_gotoxy(1,2);
      printf(lcd_putc," FI \n ");//imprimir en pantalla LCD
      delay_ms(300); // retraso.
    }
}
```

6. Realizar un programa empleando el compilador de C, para ejecutar las acciones mostradas en la siguiente tabla, estas son controladas a través del puerto serie; usar retardos de $\frac{1}{2}$ segundos.

DATO	ACCION Puerto B	*Ejecución*
0	Todos los bits apagados	*00000000*
1	Todos los bits encendidos	*11111111*
2	Corrimiento del bit más significativo hacia la derecha	*10000000* …….. *00000001*
3	Corrimiento del bit menos significativo hacia la izquierda	*00000001* …….. *10000000*
4	Corrimiento del bit más significativo hacia la derecha y a la izquierda	*10000000* …….. *00000001* …….. *10000000*
5	Apagar y encender todos los bits.	*00000000* *11111111*

Algoritmo

Para este ejercicio se lee del puerto serial con la función getc(), que obtiene un carácter únicamente. Para convertirlo en un entero le restamos 48 al carácter leído. Teniendo este número usamos un switch para determinar que hacer segundo la tabla. Para prender y apagar los leds simplemente escribimos FF o un 0 como corresponda. Para los corrimientos se realizaron funciones que utilizan los operadores "«" y "»"para hacer los corrimientos. En un loop verifican que no se haya desbordado el número, para después hacer el corrimiento.

Código comentado

```
#include <16F877.h>
#include <stdlib.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)
#org 0x1F00, 0x1FFF void loader16F877(void) {}

//Prototipos de funciones
void left_shift();
void right_shift();

void main() {
   int8 flag = FALSE;
   while( TRUE ) {
      //Le un caracter del puerto serie
      char input = getc();
      //Convierte el caracter a decimal
      int8 option = input - 48;
```

```
        printf("%d", option);
        //Elige el flujo correcto segun la
        //opcion
        switch(option) {
            case 0:
                //apaga todos los leds
                output_b(0x00);
                break;
            case 1:
                //prende todos los leds
                output_b(0xFF);
                break;
            case 2:
                //hace corrimiento a la izquierda
                left_shift();
                break;
            case 3:
                //hace corrimiento a la derecha
                right_shift();
                break;
             case 4:
                //hace zigzag
                left_shift();
                right_shift();
                break;
             case 5:
                //intercambia entre encendido y apagado
                if(flag){
                    output_b(0x00);
                }else{
                    output_b(0xFF);
                }
                flag = !flag;
                break;
            default:
                break;
        }
    }
}

void right_shift() {
    //mascara
    int8 x = 128;
    //mientras aun haya al menos
    //un bit se hace el corrimiento
    while(x > 1) {
        x >>= 1; //hace el corrimiento
        output_b(x); //imprime la mascara
        delay_ms(1000);
    }
```

```
}

void left_shift() {
    //mascara
    int8 x = 1;
    //Mientras aun no se desborde el
    //numero se hace el corrimiento
    while(x < 128) {
        x <<= 1; //hace el corrimiento
        output_b(x); //imprime la mascara
        delay_ms(1000);

    }
}
```

7. Realizar un programa que muestre en un Display de Cristal Líquido, la cantidad de veces que se ha presionado un interruptor, el cual esta conectado a la terminal A0. El despliegue a mostrar es:

   a) Primer línea y 5 columna; la cuenta en decimal

   b) Segunda línea y 5 columna; la cuenta en hexadecimal

Algoritmo

En este ejercicio, fue muy sencillo poder inicializar el lcd mediante código, una vez configurado, dentro el ciclo infinito leemos un valor de entrada, el cual viene del puerto A (DipSwitches), mismos de los cuales solo tomaremos el bit 5, y cuando este prendido ejecutamos la acción de incrementar el contador que se desplegará al final en el display LCD con la función `gotoxy`, la cual nos posiciona en un punto X,Y del display.

Código comentado

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#include <lcd.c>


void main() {
    lcd_init(); // Inicializamos el LCD
    int8 count = 0; // Contador
    int8 last = 0; // Ultimo valor de entrada
    while( TRUE ) {
        int8 input = input_a(); // Leemos el valor de entrada
        input &= 32;        // Nos quedamos con el bit 5
        if(last == 0 && input){ // Si el ultimo valor fue 0 y el actual es
            count++;
            last = 1;
        }
        if(input == 0){
```

```
            last = 0;
        }

        lcd_gotoxy(5,1);  // Nos posicionamos en la fila 1, columna 5
        printf(lcd_putc,"␣%d␣\n␣", count);  // Imprimimos contador
        lcd_gotoxy(5,2);  // Nos posicionamos en la fila 2, columna 5
        printf(lcd_putc,"␣%x␣\n␣", count);  // Imprimimos contador
    }
}
```

2. Conclusiones

- López Becerra Ricardo: En esta práctica aprendimos a usar varios de los periféricos de otras prácticas pero ahora en C. Pudimos apreciar como es mucho más rápido hacer los programas en este lenguaje, aunque tiene sus inconveniente, como un mayor uso de memoria. Además, aprendimos a usar el ambiente de desarrollo para programas en C.
- Navarrete Zamora Aldo Yael: Esta práctica nos deja en claro que en C es más fácil la manipulación de los datos en la PIC con el uso de los puertos paralelos de entrada y salida, con el uso de unas pocas líneas de código. El ambiente de C tiene una ligera desventaja, y esa es el precargamiento de librerías que no se les dará uso del todo, por lo que el programa resulta ser un poco más pesado en memoria.