UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Práctica 9: Programación en C Comunicación serie síncrona, I2C

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

25 de mayo del 2023

FECHA DE ENTREGA

1 de junio de 2023

1. Desarrollo

1. El objetivo del siguiente programa será para mayor comprensión de la comunicación I2C y la programación en C, por lo que se pide analizarlo y comentarlo para su reporte; observar en el circuito la conexión de A2, A1 y A0 para generar la dirección del esclavo, así como su uso en el programa.

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#use i2c(MASTER, SDA=PIN_C4, SCL=PIN_C3,SLOW, NOFORCE_SW)
int contador=0;

void escribir_i2c(){
                        i2c_start();
                        i2c_write(0x42);
                        i2c_write(contador);
                        i2c_stop();
                    }
void main()
{
        while(true)
                {
                  escribir_i2c();
                  delay_ms(500);
                  contador++;
                }
}
```

Algoritmo

El algoritmo es sencillo, es un simple contador que se incrementa de forma infinita que se muestra en el esclavo I2C que en este caso son los displays de 7 segmentos, la función `escribir_i2c()` empieza la comunicación con los esclavos y escribe en la dirección del esclavo deseado, en este caso es 0100 0001 que representado en decimal es 0x42, posteriormente, el flujo del código indica que el dato se escribe mediante la comunicación I2C en los displays el contador que irá incrementando, todo esto sucede con tiempos pausados de 500 milisegundos.

Código comentado

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
#use i2c(MASTER, SDA=PIN_C4, SCL=PIN_C3,SLOW, NOFORCE_SW)
void escribir_i2c(){
    i2c_start();      // Empieza la comunicacion y configuracion
    i2c_write(0x42); //Direcion del esclavo
    i2c_write(contador); //El dato a escribir
    i2c_stop();       // Termina la comunicacion
}
```

```
void main() {
    while(true) {
      escribir_i2c();
      delay_ms(500);
      contador++;
    }
}
```

2. Realizar la modificación al programa para que también muestre el contador en eL puerto B. Nota: Considerar que se usar un decodificador BCD-7SEG; por lo que deberá modificar el programa para ver la cuenta continua.

Algoritmo

El algoritmo es sencillo, es un simple contador que se incrementa de forma infinita que se muestra en el esclavo I2C que en este caso son los displays de 7 segmentos, la función `escribir_i2c()` empieza la comunicación con los esclavos y escribe en la dirección del esclavo deseado, en este caso es 0100 0001 que representado en decimal es 0x42, posteriormente, el flujo del código indica que el dato se escribe mediante la comunicación I2C en los displays el contador que irá incrementando, todo esto sucede con tiempos pausados de 500 milisegundos.

Código comentado

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT
//configuracion del reloj
#use delay(clock=20000000)
//COnfiguracion para poder usar el protocolo i2c
#use i2c(MASTER, SDA=PIN_C4, SCL=PIN_C3,SLOW, NOFORCE_SW)
#include <i2c_LCD.c>
int8 contador=0; //contador principal
int8 countAux = 0; //contador auxiliar
//Funcion para mandar el numero por el
protocolo i2c
void escribir_i2c(){
   i2c_start();
   i2c_write(0x42); //Direcion del esclavo
   i2c_write(contador); //El dato a escribir
   i2c_stop();
}
void main()
{  //Inicializacion de  la pantalla LCD
   lcd_init(0x4E, 16, 2);
   while(true)
   {
      //Elige la posicion de escritura
      lcd_gotoxy(0,2);
      //Imprime el contador
      printf(lcd_putc,"Contador = %d", contador);
      //Escribe el numero por i2c
```

```
        escribir_i2c();
        //Escribe el numero en el puerto b
        output_d(contador);
        delay_ms(500);
        //Aumenta los contadores
        contador++;
        countAux++;
        //Si se cuenta hasta 10, se aplica un corrimiento
        //a la suma principal
        if(countAux == 10){
            countAux = 0;
            contador += 6 ;
        }
    }
}
```

3. Realizar las modificaciones necesarias para que además de lo resuelto en el ejercicio previo, muestre el contador en un display LCD que funcionará como esclavo I2C. Consideraciones:

   a) Debe incluir la librería i2c_LCD.c a su programa; esta librería contiene el protocolo de comunicación I2C para uso del Display de Cristal Líquido LCD.

   b) La biblioteca I2C_LCD permite emplear con los mismos nombres las funciones empleadas para el LCD en formato paralelo.

   c) Es necesario incluir la función que configura la forma de comunicación I2C del LCD. a. lcd_init(DIRECCION_ESCLAVO,COLUMNAS,RENGLONES);

      1) DIRECCION_ESCLAVO: es la dirección configurada por los valores fijos de fabrica y los definidos por hardware del módulo PCF8574 que controla al LCD; ubicar en el esquemático la configuración, para obtener la dirección.

      2) COLUMNAS: es la cantidad de columnas de LCD

      3) RENGLONES: es la cantidad de filas disponibles en el LCD

      4) En la practica se usará LCD de 16x2

   d) Uso DEC BCD-7SEG

Algoritmo

Este ejercicio es realmente el mismo que el anterior, simplemente que ahora vamos a desplegar a un esclavo más además del puerto B, ahora el resultado del contador irá al display LCD, esto lo haremos con ayuda de la biblioteca `i2c_LCD.c` que contiene todas las funciones de configuración de la pantalla para la comunicación i2C, en el ciclo infinito, posicionaremos el cursor de la pantalla en la columna 0 de la fila 2, e imprimimos el mensaje 'contador = x' donde x es el valor actual del contador.

Código comentado

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

```
#use i2c(MASTER, SDA=PIN_C4, SCL=PIN_C3,SLOW, NOFORCE_SW)
#include <i2c_LCD.c> // Agregar la biblioteca que nos permite utilizar el
int8 contador=0; // Variable global del contador
int8 countAux = 0;// Variable temporal contadora
void escribir_i2c(){
   i2c_start();
   i2c_write(0x42); //Direcion del esclavo
   i2c_write(contador); //El dato a escribir
   i2c_stop();
}
void main() {
   lcd_init(0x4E, 16, 2); // Inicializamos el LCD con la direccion del es
   while(true) {
      lcd_gotoxy(0,2); // Nos posicionamos en la fila 2, columna 0
      printf(lcd_putc,"Contador = %d", contador); // Imprime contador
      escribir_i2c(); // Funcion
      output_d(contador); // Salida al PORTB del contador
      delay_ms(500); // DELAY
      contador++;
      countAux++;
      if(countAux == 10){ // Variable auxiliar, si llega a 10, se le aum
      countAux = 0;     // Se reinicia el contador auxiliar
      contador += 6 ; // Se aumenta al contador
      }

    }
}
```

4. Realizar un programa de tal forma que obtenga la lectura de la entrada generada por otro dispositivo esclavo y los muestre en los tres periféricos usados en la actividad 3.

Algoritmo

Para este caso no se pudo aplicar el mismo algoritmo que en los ejercicios anteriores porque no sería conveniente contar hasta el número recibido con en único objetivo de generar los corrimientos para desplegar correctamente el número en el display de 7 segmentos. En este caso, se hizo una conversión directa usando la fórmula siguiente:

- n: Número de dígitos en un número expresado en formato decimal.
- p: Posición de un dígito contando de derecha a izquierda empezando por 0. Por ejemplo, para 50, 5 tiene $p = 1$
- d_p: Dígito en la la posición p. Por ejemplo, para 60 y para $p = 1$ entonces $d = 6$.
- r: Número a mandar al puerto D.

$$r = d_0(16)^0 + d_1(16)^1 + d_2(16)^2 + \cdots + d_n(16)^n$$

Mandando el número original a la pantalla lcd y r a los displays de 7 segmentos se pudo observar el mismo valor en ambos.

Código comentado

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT
#use delay(clock=20000000)
//COnfiguracion del protocolo i2c
#use i2c(MASTER, SDA=PIN_C4, SCL=PIN_C3,SLOW, NOFORCE_SW)
//Bibliteca para el uso de la pantalla LCD
#include <i2c_LCD.c>
//Funcion para obtener la potencia de un numero
int8 power(int8 base, int8 p){
   int res = 1;
   for(int i = 0; i < p; i++){
      res *= base;
   }
   return res;
}
//Funcion para escribir con el protocolo 12c
//dato: calor a escribir por i2c
void escribir_i2c(int8 dato){
   i2c_start();
   i2c_write(0x42); //Direcion del esclavo
   i2c_write(dato); //El dato a escribir
   i2c_stop();
}
//funcion que convierte un numero decimal de tal
//manera que se muestre correctamente en un decodificador
//para numeros hexadecimales
int8 dec2Hex(int8 dato){
   int8 res = 0;
   int8 pos = 0;
   //Mientras el numero aun tenga digitos este se multiplicara
   //por la potencia de 16 correspondiente.
   while(dato > 0){
      int8 digito = dato % 10;
      dato /= 10;
      res += power(16, pos) * digito;
      pos++;
   }
   return res;
}
//Funcion que lee un por el protocolo i2c
int8 leer_i2c(){
   int8 dato;
   i2c_start();
   i2c_write(0x45); //Direcion del esclavo
   dato = i2c_read(); //El dato a escribir
   i2c_read(0);//cierre de comunciacion
   i2c_stop();
   return dato;
}
void main()
```

```
{
   //configuracion de la pantalla lcd
   lcd_init(0x4E, 16, 2);

   while(true)
   {
      //limpia la pantalla lcd
      printf(lcd_putc,"          ");
      //lee un dato con el protocolo
      //i2c
      int8 dato = leer_i2c();
      //Se mueve a la posicion correcta
      //de la pantalla
      lcd_gotoxy(0,2);
      //Imprime el valor en la pantalla lcd
      printf(lcd_putc,"Dato = %d", dato);
      //Manda al display de 7 segmentos el valor
      //convertido
      escribir_i2c(dec2Hex(dato));
      //Manda al otro display de 7 segmentos el valor
      //convertido
      output_d(dec2Hex(dato));
      delay_ms(500);
    }
}
```

2. Conclusiones

- López Becerra Ricardo: En esta práctica aprendimos a controlar periféricos con el protocolo i2c el cual es muy utilizado en aplicaciones de la vida real por su versatilidad y relativa facilidad de uso una vez configurado. Una ventaja de este periférico es que varios esclavos se pueden conectar al mismo maestro por el mismo puerto, por lo que no hay que hacer múltiples conexiones. Por otro lado, de igual manera que en la práctica anterior se pudo ver como la programación en C es mucho más conveniente.
- Navarrete Zamora Aldo Yael: La práctica concretamente se enfoca en demostarnos cómo es que sucede la comunicación entre masters y esclavos a través de i2C, esta comunicación resulta ser conveniente cuando necesita haber un cambio de control entre distintos componentes dentro de un sistema para que pueda ser concretada la comunicación serial síncrona. Además, así como en las prácticas pasadas, resulta ser más conveniente y fácil manejar el entorno de desarrollo en ambiente C para las prácticas.