ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Departamento de Eléctrica y Electrónica

Carrera de Electrónica y Automatización

SISTEMAS BASADOS EN MCU

Práctica 3.9

UTILIZACIÓN DEL MÓDULO USART DEL PIC16F877

Autor:

Iza Tipanluisa Alex Paul

Docente:

Ing. Amparo Meythaler

NRC: 4891

1) OBJETIVOS

- Identificar las características del módulo USART del PIC16F877.
- Realizar una aplicación del modo Asincrónico del USART (simplex), en lenguaje C para el PIC16F877.

2) MARCO TEORICO

MÓDULO USART

Universal Synchronous Asynchronous Receiver Transmitter. También se conoce como SCI (Serial Communications Interface).

El módulo USART de los microcontroladores PIC permite la comunicación con otros microcontroladores, memorias eeproms, PC, etc, utilizando tanto un modo asincrónico (pudiendo ser full duplex) o sincrónico (master o slave, half duplex).

Permite enviar 8 o 9 bits por transmisión y tiene posibilidad de alcanzar varias velocidades de transmisión, siendo lo más importante, que por ser una funcionalidad integrada al PIC, el programador logra desentenderse de toda la complejidad del protocolo a nivel de capa 1 y es suficiente con poner un dato en un registro del microcontrolador para enviar y leer otro registro para recibir.

Si bien existen varios protocolos relacionados con la sincronización de la comunicación USART como el RTS/CTS o el XON/XOFF, los microcontroladores PIC de gama media no los implementan, para utilizarlos hay que programarlos utilizando algún par de pines disponibles.

En el PIC 16F877, los pines que son utilizados por la funcionalidad integrada USART son el PC6/TX y el PC7/RX.

3) EQUIPOS Y MATERIALES

- PC con el paquete PIC C.
- PC con el paquete PROTEUS.

4) ACTIVIDADES

1) Trabajo Preparatorio:

a) Realizar el diagrama de flujo, la codificación e implementación correspondiente, de un programa que comunique dos microcontroladores en modo simplex (un PIC transmisor y otro PIC el receptor), utilizando el USART del PIC16F877. El transmisor tendrá un teclado y el receptor un LCD. La tecla que se pulse en el transmisor será verificada en el receptor.

1) Trabajo en el paquete que maneja lenguaje C.

- Digite el programa.
- Compile el ejercicio hasta que obtenga 0 errores (presente una captura del paquete).

RECEPTOR

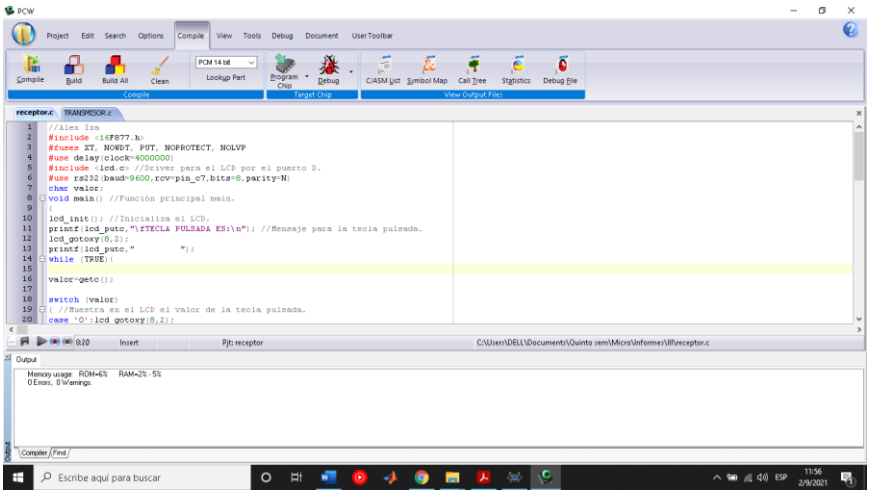

Figura 1, Compilación sin errores del programa para el receptor

TRANSMISOR

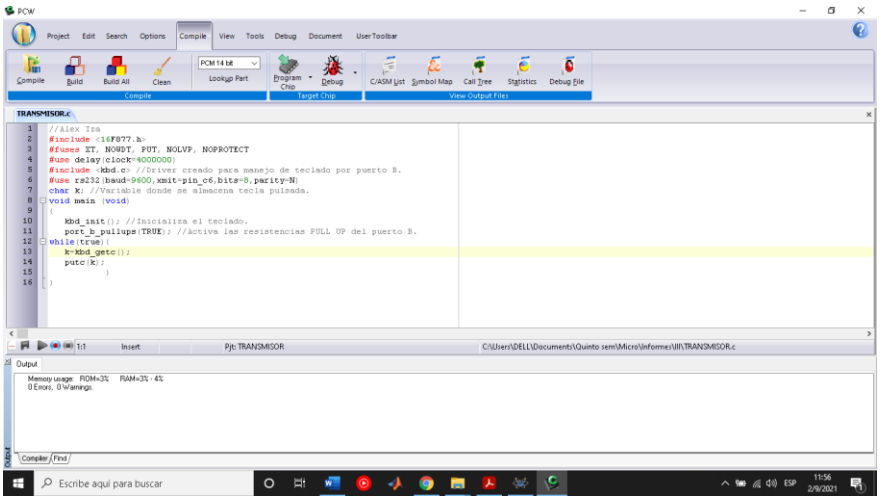

Figura 2, Compilación sin errores del programa para el transmisor

2) Trabajo en el paquete PROTEUS.

- Realice el diagrama esquemático.
- Cargue el programa compilado en el microcontrolador.

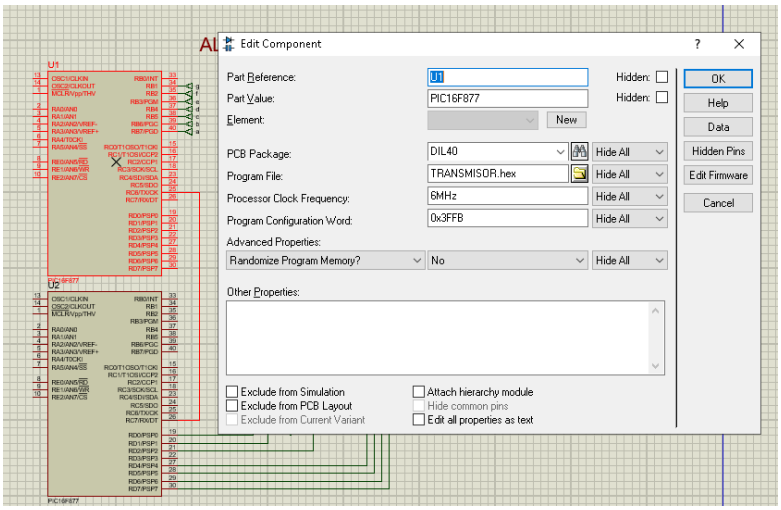

Figura 3, Asignación del programa en el Pic para el transmisor

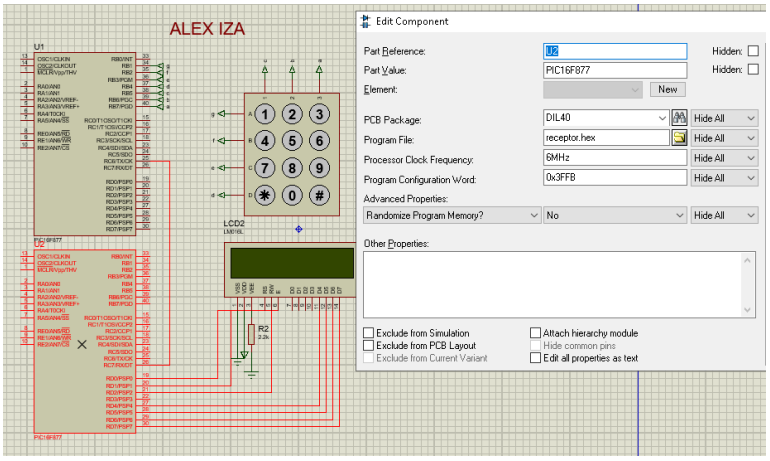

Figura 4, Asignación del programa en el Pic para el receptor

5) RESULTADOS

- Explique los errores cometidos en el ejercicio realizado (si los tuvo) y la forma de corregirlos.

  Tuve un error para que el LCD empiece a funcionar y dar los valores solicitados, ya que no lo está inicializando, lo solucione colocando la instrucción lcd_init().

  Y también al momento de funcionar el programa me daba el valor de 0 sin pulsar ningún teclado, lo solucione borrando ese dato desde el inicio y que en el LCD no se muestre nada

- Ponga las características del protocolo de comunicación serie RS232, el mismo que es usado en la comunicación del módulo USART.

  1. Los datos que se envían a través de las líneas RS232 son simplemente datos binarios ceros y unos
  2. Los datos enviados por un dispositivo son cuidadosamente cronometrados por el dispositivo receptor y decodificado.
  3. Define un marco eléctrico general relativamente flexible para transmitir y recibir impulsos eléctricos.
  4. La codificación de caracteres, el espaciado, los bits de inicio, los bits de parada, el orden de bits, la detección de errores, la velocidad de transmisión de bits, etc. no son responsabilidad del ámbito RS-232

6) DISEÑO

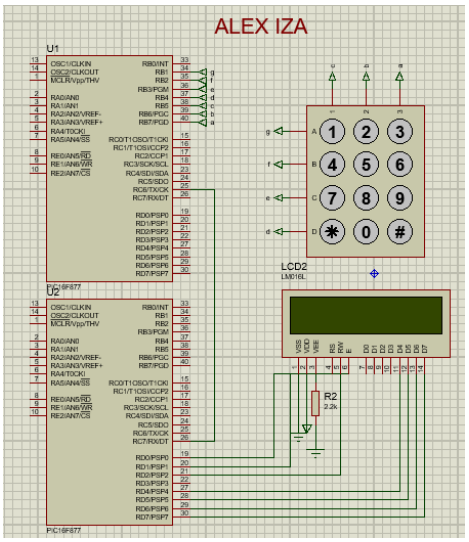

Figura 5, Diseño del programa

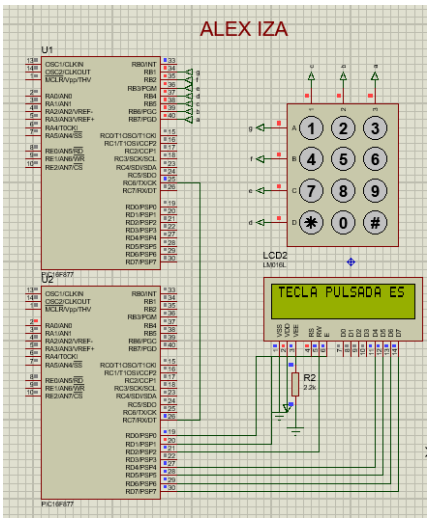

Figura 6, Estado de espera para que ingrese un dato

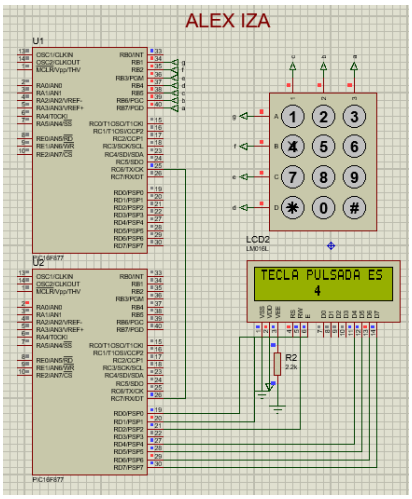

Figura 7, Visualización de la tecla pulsada

7) DIAGRAMA DE FLUJO

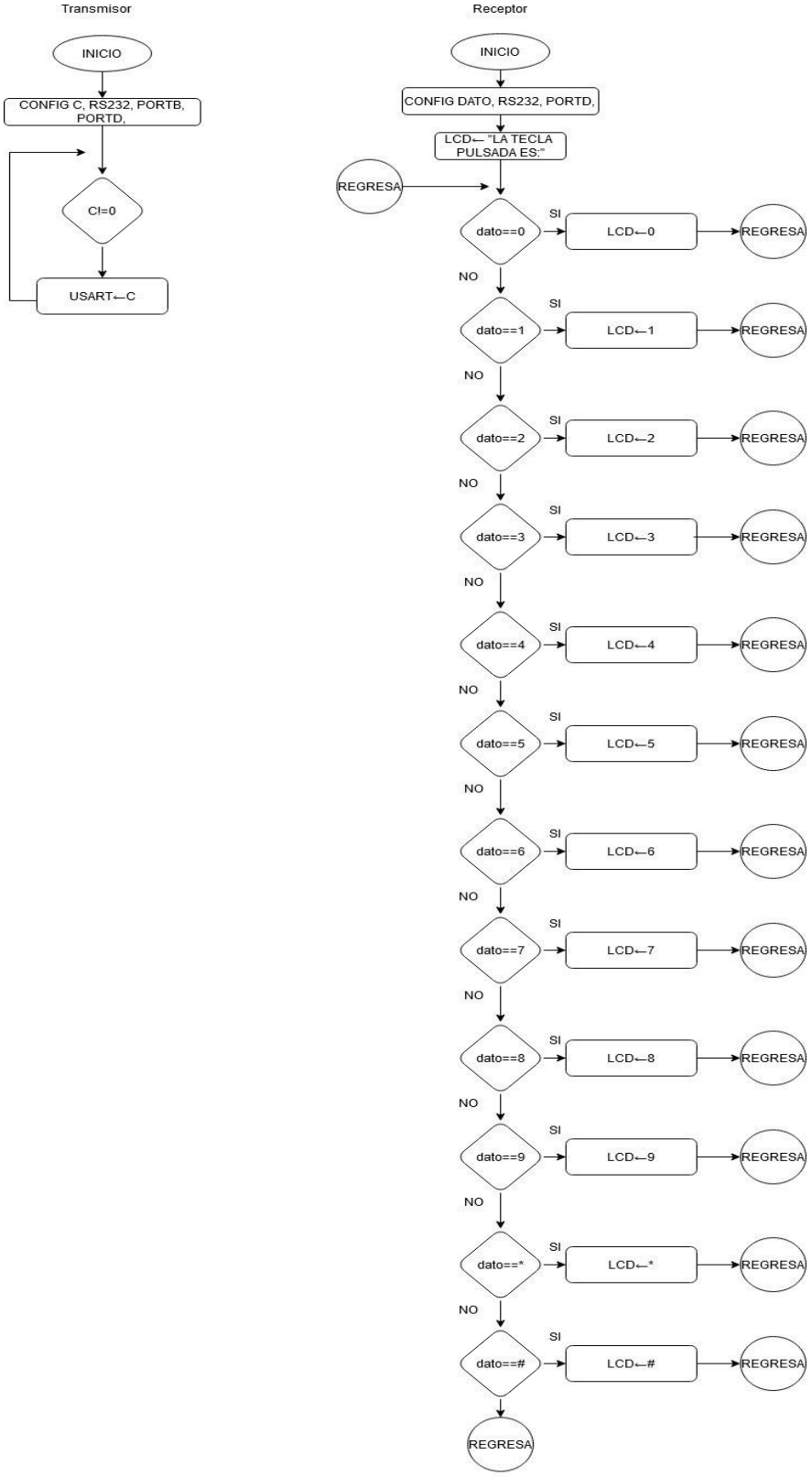

Diagrama 1

8) PROGRAMA

PROGRAMA DEL RECEPTOR

```
//Alex Iza
#include <16F877.h>
#fuses XT, NOWDT, PUT, NOPROTECT, NOLVP
#use delay(clock=4000000)
#include <lcd.c> //Driver para el LCD por el puerto D.
#use rs232(baud=9600,rcv=pin_c7,bits=8,parity=N)
char valor;
void main() //Función principal main.{
lcd_init(); //Inicializa el LCD.
printf(lcd_putc,"\fTECLA PULSADA ES:\n"); //Mensaje para la tecla pulsada.
lcd_gotoxy(8,2);
printf(lcd_putc,"         ");
while (TRUE){
valor=getc();
switch (valor)
{ //Muestra en el LCD el valor de la tecla pulsada.
case '0':lcd_gotoxy(8,2);
        printf(lcd_putc,"0");
break;
case '1':lcd_gotoxy(8,2);
        printf(lcd_putc,"1");
break;
case '2':lcd_gotoxy(8,2);
        printf(lcd_putc,"2");
break;
case '3':lcd_gotoxy(8,2);
        printf(lcd_putc,"3");
```

```
break;
case '4':lcd_gotoxy(8,2);
        printf(lcd_putc,"4");
break;
case '5':lcd_gotoxy(8,2);
        printf(lcd_putc,"5");
break;
case '6':lcd_gotoxy(8,2);
        printf(lcd_putc,"6");
break;
case '7':lcd_gotoxy(8,2);
        printf(lcd_putc,"7");
break;
case '8':lcd_gotoxy(8,2);
        printf(lcd_putc,"8");
break;
case '9':lcd_gotoxy(8,2);
        printf(lcd_putc,"9");
break;
case '#':lcd_gotoxy(8,2);
        printf(lcd_putc,"#");
break;
case '*':lcd_gotoxy(8,2);
        printf(lcd_putc,"*");
break;
default:
 break;
}
}
} //Fin del bucle infinito.
```

PROGRAMA DEL TRANSMISOR

```
//Alex Iza
#include <16F877.h>
#fuses XT, NOWDT, PUT, NOLVP, NOPROTECT
#use delay(clock=4000000)
#include <kbd.c> //Driver creado para manejo de teclado por puerto B.
#use rs232(baud=9600,xmit=pin_c6,bits=8,parity=N)
char k; //Variable donde se almacena tecla pulsada.
void main (void)
{
  kbd_init(); //Inicializa el teclado.
  port_b_pullups(TRUE); //Activa las resistencias PULL UP del puerto B.
while(true){
  k=kbd_getc();
  putc(k);
        }
    }
```

9) CONCLUSIONES

- Se ha concluido que los pines RX Y TX sirven para recibir los datos y trasmitir datos respectivamente
- Se ha concluido que el PIC 16F877 solo tiene un puerto UART físico para realizar la comunicación serial, aunque existen otros PICS que poseen más de un UART.
- Se ha concluido que, para esta práctica, los datos se transmiten bit a bit, por lo tanto, los procesos se hacen mucho más lentos.
- Se ha concluido que, para la recepción y transmisión de datos de un Pic a otro, cada uno de ellos debe tener su propio programa.

10) RECOMENDACIONES

- Se recomienda que el valor de los baudios sean 300, 600,1200, 2400, 9600, 14400 y 28800 baudios, ya que estos son los más comunes en usar.
- Se recomienda que para que el microcontrolador reciba los datos del teclado, este

se lo realice con una interrupción por recepción serial.

- Se recomienda que en el LCD no se muestren los datos hasta que se se presione una tecla.

11) BIBLIOGRAFIA

Reinoso, S., Mena, L., Pilatasig, M., & Sánchez, J. (2018). *Programación de microcontroladores PIC con Lenguaje C, Tomo I.* Sangolquí, Ecuador: Universidad de lasFuerzas Armadas Espe.

Meythaler, A. (2021). Sistemas basados en MCU. Ecuador: UFA ESPE