	Laboratorio de Microcomputadoras

Profesor(a): Dra. Lourdes Angelica Quiñonez Juárez

Asignatura: Laboratorio de Microcomputadoras

Grupo Laboratorio: 5

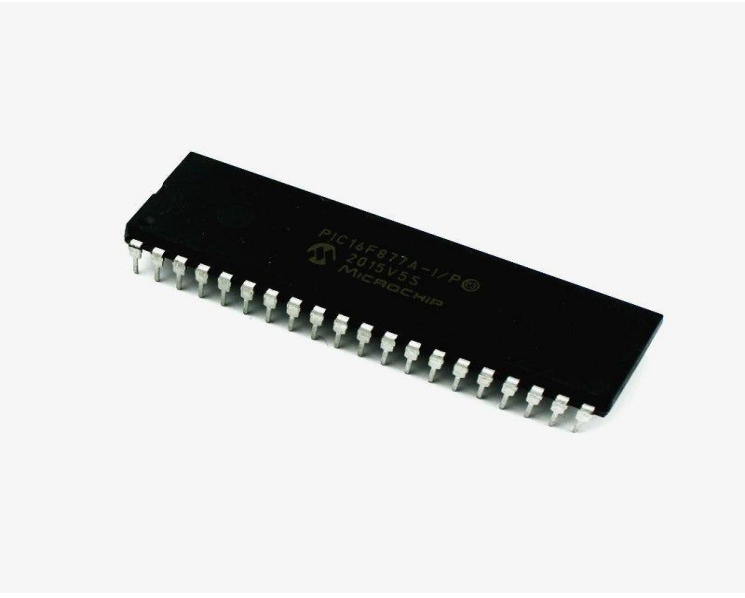
Grupo Teoría: 3

No. de Práctica(s): 7

Integrante(s): Hernández Diaz Sebastián

Semestre: 2025-2

Fecha de entrega: 7 de abril del 2025



Hernández Diaz Sebastian

Laboratorio de Microcomputadoras

Práctica No. 7

Puerto Serie SCI (Asíncrono)

Objetivo. Familiarizar al alumno en el uso de una Interfaz de Comunicación Serie Asíncrona de un microcontrolador.

Desarrollo. Realizar los siguientes apartados:

1.- Utilizando el programa resuelto en la práctica No. 3, ejercicio 2 (Control de acciones), realizar las modificaciones necesarias para que ahora se controle por medio del teclado de su PC, el cuál transmitirá el comando de la acción a ejecutar.

Abrir una Terminal, usando la Hyper Terminal que contiene Windows o la Terminal incluida en el IDE PIC C Compiler, consultar los apéndices A y B.

TECLA	ACCION Puerto B
0	Todos los bits del puerto apagados
1	Todos los bits del puerto encendidos
2	Corrimiento del bit más significativo hacia la derecha
3	Corrimiento del bit menos significativo hacia la izquierda
4	Corrimiento del bit más significativo hacia la derecha y a la izquierda
5	Apagar y encender todos los bits.

Tabla 7.1 Controla a través del puerto serie

El código usado en esta actividad es el siguiente:

Hernández Diaz Sebastian

```
processor 16f877          ; Se especifica el microcontrolador PIC16F877
include<p16f877.inc>      ; Se incluye el archivo de definiciones del PIC16F877

    ; Declaración de variables
    G   equ 0x20            ; Registro G en la dirección 0x20
    Y   equ 0x24            ; Registro Y en la dirección 0x24
    X   equ 0x25            ; Registro X en la dirección 0x25

    ; Definición de constantes
    VALOR1 equ H'21'        ; Valor 1 en hexadecimal
    VALOR2 equ H'22'        ; Valor 2 en hexadecimal
    VALOR3 equ H'23'        ; Valor 3 en hexadecimal
    cte1 equ 20H            ; Constante 1
    cte2 equ 50H            ; Constante 2
    cte3 equ 60H            ; Constante 3

    org 0                   ; Punto de inicio del código
        goto inicio          ; Salto a la etiqueta 'inicio'

    org 5                   ; Dirección de memoria donde comienza la ejecución del código
inicio:
    BSF STATUS,RP0          ; Seleccionar banco 1
    BCF STATUS,RP1          ; Mantener en banco 1
    BSF TXSTA,BRGH          ; Configurar velocidad alta para USART
    CLRF TRISD              ; Configurar PORTD como salida
    MOVLW D'32'             ; Cargar divisor de baudrate (Fosc=20MHz, baudrate=38400)
    MOVWF SPBRG             ; Configurar la velocidad de transmisión
    BCF TXSTA,SYNC          ; Configurar modo asíncrono
    BSF TXSTA,TXEN          ; Habilitar transmisión USART

    BCF STATUS,RP0          ; Volver al banco 0
    BSF RCSTA,SPEN          ; Habilitar el puerto serie
    BSF RCSTA,CREN          ; Activar recepción

RECIBE:
    BTFSS PIR1,RCIF         ; Verificar si hay un dato recibido en RCREG
    GOTO RECIBE             ; Si no hay dato, permanecer en bucle
    MOVF RCREG,W            ; Mover el dato recibido a W
    MOVWF Y                 ; Guardar en Y
```

Hernández Diaz Sebastian

```
LOOP:
    CLRF PORTB              ; Limpiar PORTB
    BCF STATUS,Z            ; Limpiar bit Z

    MOVF Y,0                ; Mover Y a W
    xorlW A'0'              ; Comparar con ASCII '0'
    BTFSC STATUS,Z
    GOTO NINGUNO            ; Si es '0', ir a NINGUNO

    MOVF Y,0
    SUBLW A'1'
    BTFSC STATUS,Z
    GOTO TODOS              ; Si es '1', activar TODOS

    MOVF Y,0
    SUBLW A'2'
    BTFSC STATUS,Z
    GOTO DERECHA            ; Si es '2', activar DERECHA

    MOVF Y,0
    SUBLW A'3'
    BTFSC STATUS,Z
    GOTO IZQUIERDA          ; Si es '3', activar IZQUIERDA

    MOVF Y,0
    SUBLW A'4'
    BTFSC STATUS,Z
    GOTO RL                 ; Si es '4', activar RL

    MOVF Y,0
    SUBLW A'5'
    BTFSC STATUS,Z
    GOTO ONOFF              ; Si es '5', activar ONOFF

NINGUNO:
    CLRF PORTD              ; Apagar PORTD
    MOVLW H'00'
    MOVWF PORTD
    GOTO RECIBE2            ; Regresar a recepción
```

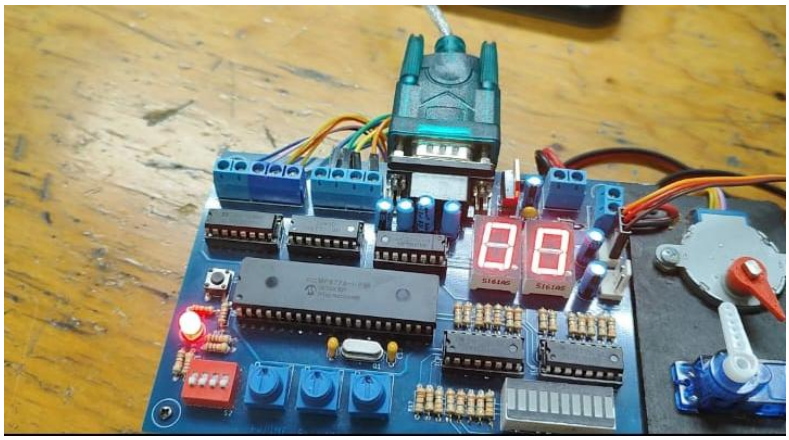
Hernández Diaz Sebastian

```
TODOS:
    CLRF PORTD
    MOVLW H'FF'
    MOVWF PORTD             ; Encender todas las salidas
    GOTO RECIBE2

DERECHA:
    MOVLW H'80'       ; Cargar el valor 0x80 en el registro W (bit más significativo activado)
    MOVWF G           ; Almacenar en la variable G
    MOVWF PORTD       ; Enviar el valor a PORTD para encender el LED más a la derecha
    BCF STATUS,C      ; Borrar el bit de acarreo (Carry) para el desplazamiento

DERECHA2:
    RRF G,0           ; Rotar G a la derecha sin afectar W
    MOVWF G           ; Guardar el nuevo valor en G
    MOVF G,0          ; Cargar G en W
    MOVWF PORTD       ; Enviar el valor actualizado a PORTD
    CALL RETARDO      ; Llamar la subrutina de retardo para visualizar el movimiento
    MOVF G,0          ; Cargar nuevamente el valor de G en W
    SUBLW H'01'       ; Comparar con 0x01 (última posición del LED encendido)
    BTFSS STATUS,Z    ; Si no se ha alcanzado el último LED, continuar el desplazamiento
    GOTO DERECHA2     ; Volver a repetir el ciclo de desplazamiento hacia la derecha
    GOTO RECIBE2      ; Una vez terminado, volver a la espera de nuevos comandos

IZQUIERDA:
    MOVLW H'01'       ; Cargar el valor 0x01 en W (bit menos significativo activado)
    MOVWF G           ; Almacenar en la variable G
    MOVWF PORTD       ; Enviar el valor a PORTD para encender el LED más a la izquierda
    CALL RETARDO      ; Llamar retardo para visualizar la activación del LED
    BCF STATUS,C      ; Borrar el bit de acarreo

IZQUIERDA2:
    RLF G,0           ; Rotar G a la izquierda sin afectar W
    MOVWF G           ; Guardar el nuevo valor en G
    MOVF G,0          ; Cargar G en W
    MOVWF PORTD       ; Enviar el nuevo valor a PORTD para desplazar el LED encendido
    CALL RETARDO      ; Esperar un momento antes de seguir
    MOVF G,0          ; Cargar nuevamente el valor de G en W
    BCF STATUS,C      ; Borrar el bit de acarreo antes de la comparación
    ADDLW H'80'       ; Comparar con 0x80 (última posición del LED encendido)
    BTFSS STATUS,Z    ; Si no se ha alcanzado, continuar desplazamiento
    GOTO IZQUIERDA2   ; Repetir el ciclo hasta alcanzar el final
    GOTO RECIBE2      ; Volver a la espera de nuevos datos
```

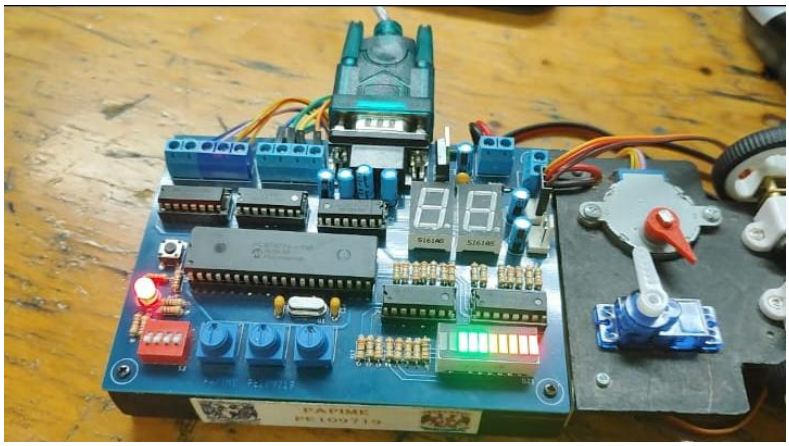
Hernández Diaz Sebastian

```
RL:
    MOVLW H'80'       ; Cargar el valor 0x80 en W (bit más significativo activado)
    MOVWF G           ; Almacenar en G
    BCF STATUS,C      ; Borrar bit de acarreo
    MOVWF PORTD       ; Enviar el valor inicial a PORTD

DER:
    RRF G,0           ; Rotar G a la derecha
    MOVWF G           ; Guardar el nuevo valor en G
    MOVF G,0          ; Cargar G en W
    MOVWF PORTD       ; Enviar el nuevo valor a PORTD
    CALL RETARDO      ; Esperar un momento
    MOVF G,0          ; Cargar G en W
    SUBLW H'01'       ; Comparar con 0x01
    BTFSS STATUS,Z    ; Si no es 0x01, continuar rotación a la derecha
    GOTO DER          ; Repetir el ciclo de rotación

    GOTO IZQUIERDA    ; Una vez alcanzado el extremo derecho, comenzar rotación a la izquierda

ONOFF:
    MOVLW H'FF'       ; Cargar 0xFF en W (todos los bits en alto)
    MOVWF PORTD       ; Encender todos los LEDs en PORTD
    CALL RETARDO      ; Esperar un momento
    MOVLW H'00'       ; Cargar 0x00 en W (todos los bits en bajo)
    MOVWF PORTD       ; Apagar todos los LEDs en PORTD
    CALL RETARDO      ; Esperar un momento
    GOTO RECIBE2      ; Volver a recibir datos

RECIBE2:
    MOVF RCREG,W      ; Leer el dato recibido
    MOVWF TXREG       ; Cargarlo en el registro de transmisión USART
    BSF STATUS,RP0    ; Seleccionar banco 1

TRANSMITE:
    BTFSS TXSTA,TRMT  ; Esperar a que la transmisión finalice
    GOTO TRANSMITE    ; Seguir esperando si aún no ha terminado
    BCF STATUS,RP0    ; Volver al banco 0
    GOTO RECIBE       ; Volver a recibir más datos

RETARDO:
    MOVLW cte1        ; Cargar la primera constante en W
    MOVWF VALOR1      ; Guardar en VALOR1

        '
        TRES:
            MOVLW cte2        ; Cargar la segunda constante
            MOVWF VALOR2      ; Guardar en VALOR2

        DOS:
            MOVLW cte3        ; Cargar la tercera constante
            MOVWF VALOR3      ; Guardar en VALOR3

        UNO:
            DECFSZ VALOR3     ; Decrementar VALOR3 y verificar si es 0
            GOTO UNO          ; Si no es 0, seguir decrementando
            DECFSZ VALOR2     ; Decrementar VALOR2 y verificar si es 0
            GOTO DOS          ; Si no es 0, seguir decrementando
            DECFSZ VALOR1     ; Decrementar VALOR1 y verificar si es 0
            GOTO TRES         ; Si no es 0, seguir decrementando
            RETURN            ; Terminar retardo y regresar

        FIN:
            END
```

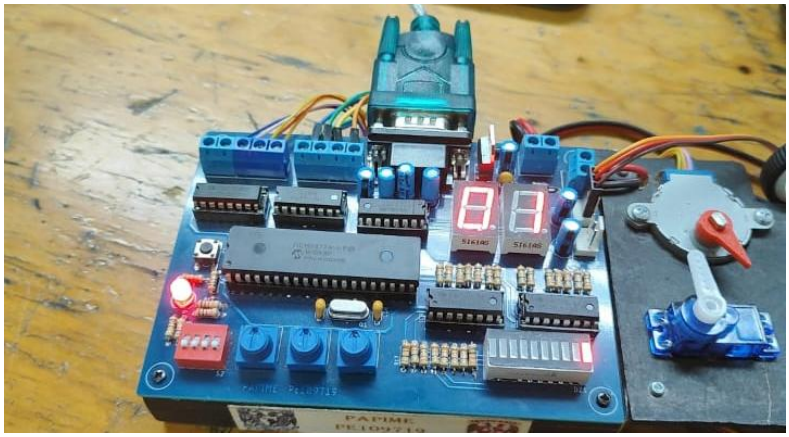
El código en ejecución es el siguiente:



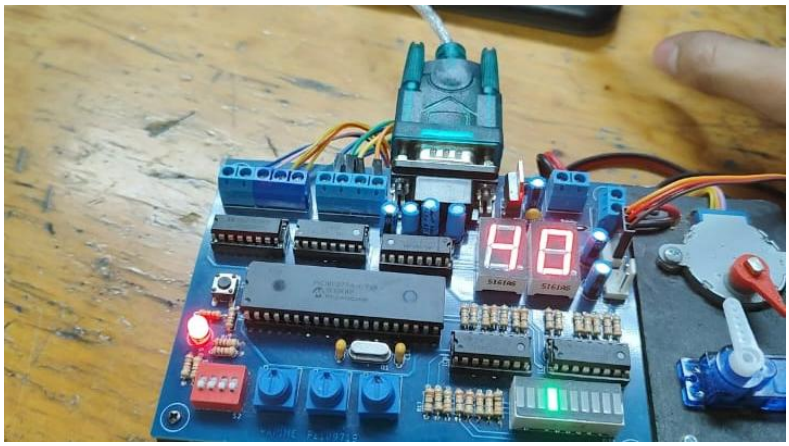
En este primer caso tenemos un cero donde todos los leds estan apagados.



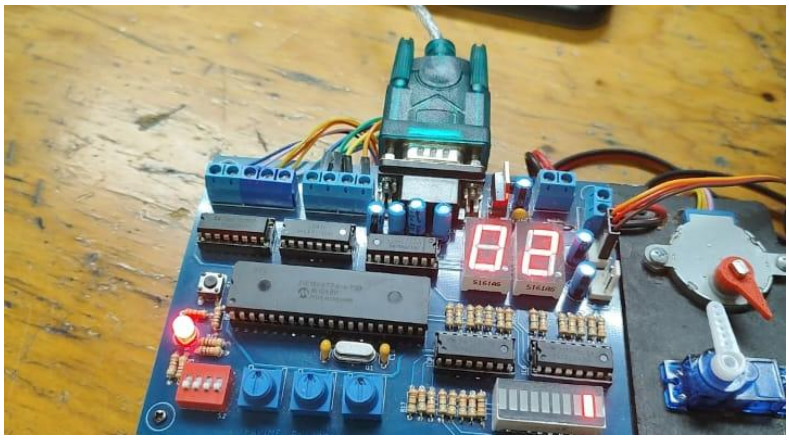
En la selección de 1 tenemos todos los bits del uerto encendidos.



En la selección de 2 tenemos corrimiento del bit más significativo hacia la derecha.



En la selección de 3 tenemos corrimiento del bit menos significativo hacia la izquierda.



En la selección de 4 tenemos corrimiento del bit más significativo hacia la derecha y a la izquierda.

En la selección de 5 tenemos apagar y encender todos los bits.

Aunque en algunos casos no se logra apreciar la ejecución debido a lo que se pide en clase se mostro que todo se ejecuto de manera correcta y se obtuvo el resultado esperado.

2.- Realizar un programa que muestre las vocales (mayúsculas y minúsculas en un display de 7 segmentos, las cuales serán enviadas vía serie a través del teclado de la PC.

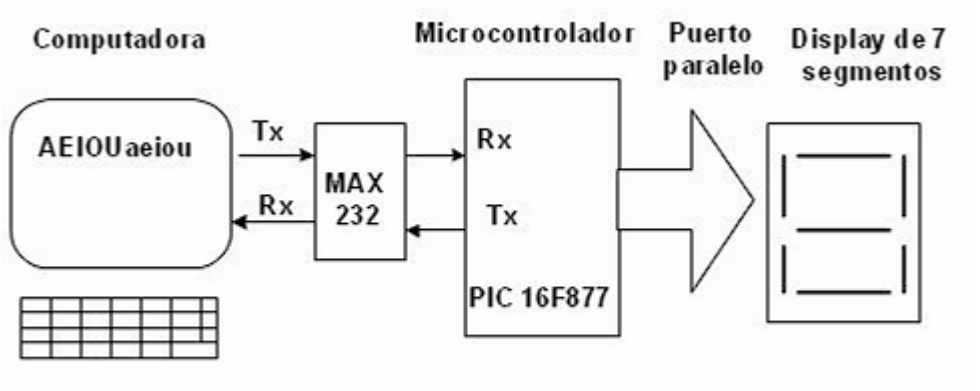


Figura 7.1 Control de despliegue de vocales

El código usado en esta actividad es el siguiente:

Hernández Diaz Sebastian

```
; Definimos el procesador PIC16F877
processor 16f877
include <p16f877.inc>

; Definir una variable para almacenar el dato recibido
Resp equ h'20'

; Iniciar la ejecución del código desde la dirección 0
org 0H
goto inicio  ; Saltar a la etiqueta inicio

; Definir la dirección de inicio del código
org 05H
inicio:
    BSF STATUS,RP0      ; Seleccionar banco 1
    BCF STATUS,RP1      ; Asegurar que estamos en banco 1
    BSF TXSTA,BRGH      ; Configurar alta velocidad en transmisión
    CLRF TRISB          ; Configurar el puerto B como salida
    MOVLW D'32'         ; Valor para 9600 baudios con oscilador de 4MHz
    MOVWF SPBRG         ; Cargar el valor en el registro del baud rate
    BCF TXSTA,SYNC      ; Configurar modo asíncrono
    BSF TXSTA,TXEN      ; Habilitar la transmisión

; Configuración del puerto serie
    BCF STATUS,RP0      ; Regresar al banco 0
    BSF RCSTA,SPEN      ; Habilitar puerto serie
    BSF RCSTA,CREN      ; Habilitar receptor

RECIBE:
    BTFSS PIR1,RCIF     ; Verificar si llegó un dato
    GOTO RECIBE         ; Si no ha llegado, seguir esperando
    MOVF RCREG,W        ; Cargar el dato recibido en W
    MOVWF Resp          ; Guardarlo en Resp
```

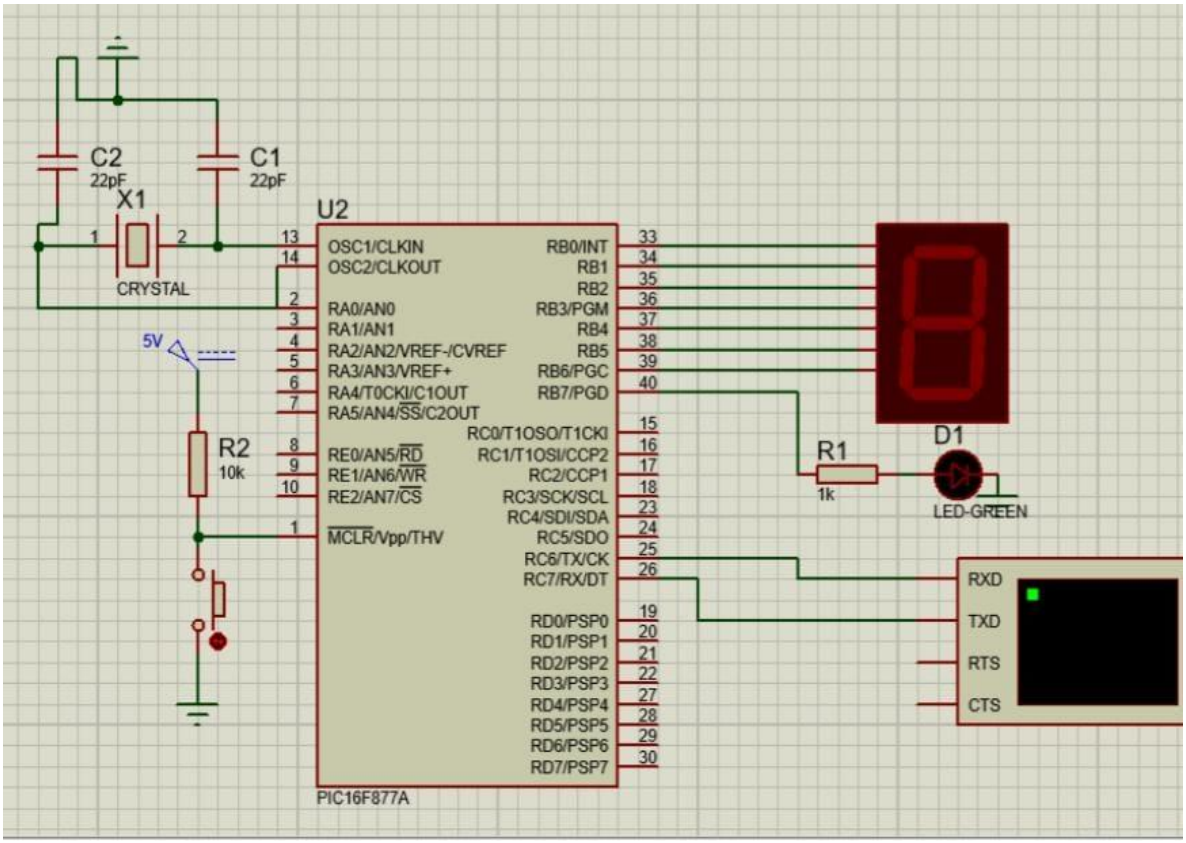
Hernández Diaz Sebastian

```
;
CICLO:
    CLRF PORTB          ; Limpiar el puerto B
    BCF STATUS,Z        ; Limpiar el bit Z

    MOVF Resp,0         ; Cargar el dato recibido
    SUBLW A'A'          ; Comparar con 'A'
    BTFSC STATUS,Z
    GOTO LETRAA         ; Si es 'A', ir a LETRA A

    MOVF Resp,0
    SUBLW A'E'
    BTFSC STATUS,Z
    GOTO LETRAE         ; Si es 'E', ir a LETRA E

    MOVF Resp,0
    SUBLW A'I'
    BTFSC STATUS,Z
    GOTO LETRAI         ; Si es 'I', ir a LETRA I

    MOVF Resp,0
    SUBLW A'O'
    BTFSC STATUS,Z
    GOTO LETRAO         ; Si es 'O', ir a LETRA O

    MOVF Resp,0
    SUBLW A'U'
    BTFSC STATUS,Z
    GOTO LETRAU         ; Si es 'U', ir a LETRA U

    MOVF Resp,0
    SUBLW A'a'
    BTFSC STATUS,Z
    GOTO LETRAAm        ; Si es 'a', ir a LETRA a

    MOVF Resp,0
    SUBLW A'e'
    BTFSC STATUS,Z
    GOTO LETRAEm        ; Si es 'e', ir a LETRA e

    MOVF Resp,0
    SUBLW A'i'
    BTFSC STATUS,Z
    GOTO LETRAIm        ; Si es 'i', ir a LETRA i
```

```
    MOVF Resp,0
    SUBLW A'o'
    BTFSC STATUS,Z
    GOTO LETRAOm        ; Si es 'o', ir a LETRA o

    MOVF Resp,0
    SUBLW A'u'
    BTFSC STATUS,Z
    GOTO LETRAUm        ; Si es 'u', ir a LETRA u

LETRAA:
    CLRF PORTB
    movlw h'f7'         ; Código hexadecimal para mostrar 'A' en display
    MOVWF PORTB
    goto RECIBIR2

LETRAE:
    CLRF PORTB
    movlw h'f9'         ; Código hexadecimal para mostrar 'E' en display
    MOVWF PORTB
    goto RECIBIR2

LETRAI:
    CLRF PORTB
    movlw h'86'         ; Código hexadecimal para mostrar 'I' en display
    MOVWF PORTB
    goto RECIBIR2

LETRAO:
    CLRF PORTB
    movlw h'Bf'         ; Código hexadecimal para mostrar 'O' en display
    MOVWF PORTB
    goto RECIBIR2

LETRAU:
    CLRF PORTB
    movlw h'BE'         ; Código hexadecimal para mostrar 'U' en display
    MOVWF PORTB
    goto RECIBIR2
```

```
; Configuración para minúsculas
LETRAAm:
    CLRF PORTB
    movlw b'11011111'  ; Código binario para 'a'
    MOVWF PORTB
    goto RECIBIR2

LETRAEm:
    CLRF PORTB
    movlw b'11111011'  ; Código binario para 'e'
    MOVWF PORTB
    goto RECIBIR2

LETRAIm:
    CLRF PORTB
    movlw b'10000100'  ; Código binario para 'i'
    MOVWF PORTB
    goto RECIBIR2

LETRAOm:
    CLRF PORTB
    movlw b'11011100'  ; Código binario para 'o'
    MOVWF PORTB
    goto RECIBIR2

LETRAUm:
    CLRF PORTB
    movlw b'10011100'  ; Código binario para 'u'
    MOVWF PORTB
    goto RECIBIR2

RECIBIR2:
    MOVF RCREG, W      ; Leer el dato recibido
    MOVWF TXREG        ; Cargar en el registro de transmisión
    BSF STATUS,RP0     ; Cambiar al banco 1 para transmisión

TRANSMITIR:
    BTFSS TXSTA, TRMT  ; Verificar si la transmisión ha finalizado
    GOTO TRANSMITIR    ; Si no, seguir esperando

    BCF STATUS,RP0     ; Regresar a banco 0 para recibir otro dato
    GOTO RECIBE        ; Volver a esperar un nuevo dato

End
```

El circuito en proteus resulto de la siguiente manera:



En este caso no se pudo probar el funcionamiento de la ejecución del código puesto que para utilizar el PIC16F877A se requiere de una licencia de proteus profesional, pero dado que el código fue hecho de manera correcta y el circuito enlazado de igual forma se puede concluir que la actividad se realizó de forma esperada.

3.- Empleando el programa No. 3 de la práctica 6 (convertidor analógico digital), realizar las modificaciones necesarias para desplegar el número de canal, de valor mayor a las otras entradas, en el monitor de la PC.

El código usado en esta actividad es el siguiente:

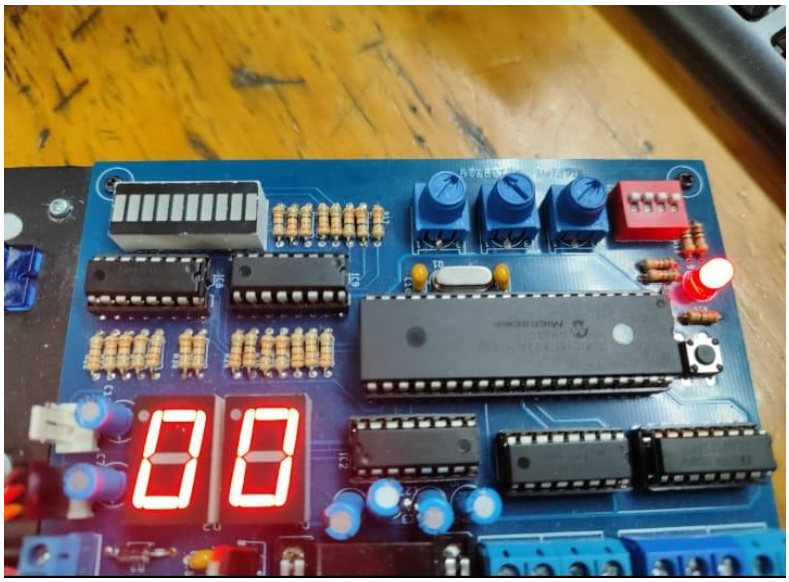
Hernández Diaz Sebastian

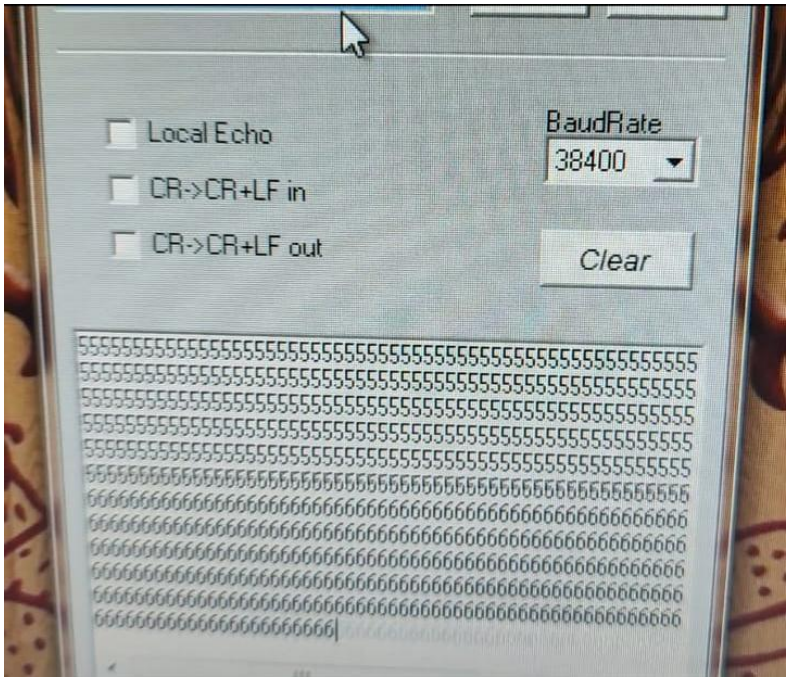
```
processor 16f877                    ; Especifica el procesador PIC16F877
include <p16f877.inc>

; Declaración de variables en la memoria
Y equ h'20'                         ; Variable Y en la dirección de memoria 0x20
X equ h'21'                         ; Variable X en la dirección de memoria 0x21
G equ h'22'                         ; Variable G en la dirección de memoria 0x22

org 0                               ; Dirección de inicio del programa
goto inicio                          ; Salta a la etiqueta inicio

org 5                               ; Dirección donde comienza la ejecución principal
inicio:
BSF STATUS,RP0                      ; Cambia al Banco 1
BCF STATUS,RP1                      ; Mantiene RP1 en 0 para configurar los registros
CLRF TRISD                          ; Configura PORTD como salida
CLRF ADCON1                         ; Configura los pines de entrada analógica
BSF TXSTA,BRGH                      ; Configura la velocidad de transmisión alta
MOVLW D'32'                         ; Carga el valor 32 en W (Baud Rate)
MOVWF SPBRG                         ; Guarda el Baud Rate en el registro SPBRG
BCF TXSTA,SYNC                      ; Configura comunicación asíncrona
BSF TXSTA,TXEN                      ; Habilita el transmisor serial
BCF STATUS,RP0                      ; Cambia a Banco 0
BSF RCSTA,SPEN                      ; Habilita el puerto serie
BSF RCSTA,CREN                      ; Habilita la recepción continua

CICLO:
MOVLW H'E9'                         ; Configura el canal analógico en ADCON0
MOVWF ADCON0                        ; Guarda la configuración en ADCON0

LEER:
BSF ADCON0,2                        ; Inicia la conversión analógica-digital
CALL RETARDO                        ; Espera tiempo para la conversión (varias llamadas)

ESPERA:
BTFSC ADCON0,2                       ; Espera a que la conversión termine
GOTO ESPERA
MOVF ADRESH,W                        ; Mueve el resultado de la conversión a W
MOVWF X                              ; Guarda el valor en la variable X

CICLOY:
MOVLW H'F1'                          ; Configura otro canal analógico
MOVWF ADCON0
```

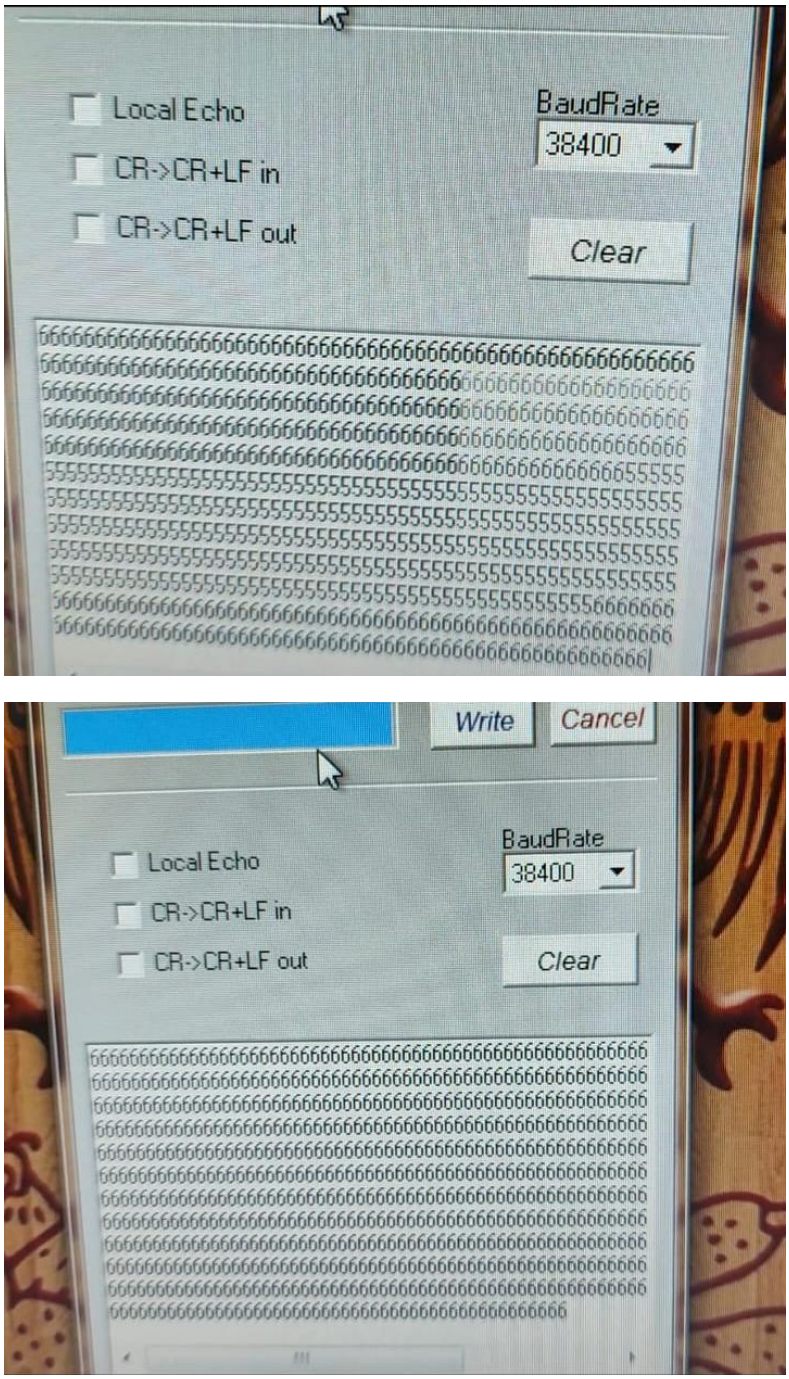
```
LEERY:
BSF ADCON0,2                    ; Inicia la conversión del segundo canal
CALL RETARDO

ESPERAY:
BTFSC ADCON0,2                  ; Espera a que termine la conversión
GOTO ESPERAY
MOVF ADRESH,W                   ; Mueve el resultado al registro W
MOVWF Y                         ; Guarda el resultado en Y

UNO:
MOVF X,W                        ; Mueve X a W
SUBWF Y                         ; Resta Y a W para comparar X con Y
BTFSS STATUS,C                  ; Si X < Y, salta a DOS
GOTO DOS
MOVLW A'5'                      ; Si X >= Y, prepara el caracter '5' para enviar
MOVWF TXREG                     ; Guarda '5' en TXREG para transmisión
BSF STATUS,RP0                  ; Cambia al Banco 1
GOTO TRANSMITE                  ; Va a la rutina de transmisión

DOS:
MOVLW A'6'                      ; Si X < Y, prepara el caracter '6' para enviar
MOVWF TXREG                     ; Guarda '6' en TXREG para transmisión
BSF STATUS,RP0                  ; Cambia al Banco 1

TRANSMITE:
BTFSS TXSTA,TRMT                ; Espera a que termine la transmisión
GOTO TRANSMITE
BCF STATUS,RP0                  ; Cambia a Banco 0
GOTO CICLO                      ; Repite el ciclo

RETARDO:
MOVLW H'FF'                     ; Carga un valor grande para el retardo
MOVWF G                         ; Guarda en G

LOOP:
DECFSZ G                        ; Decrementa G y salta si llega a 0
GOTO LOOP                       ; Repite el lazo de retardo
RETURN                          ; Retorna de la subrutina

END
```

El código en ejecución es el siguiente:

Hernández Diaz Sebastian







En este caso se puede observar que las comparaciones son correctas y la muestra en la terminal de la computadora lo muestra dando como resultado los valores de 5 y 6 dependiendo de cual potenciómetro es mayor.

Hernández Diaz Sebastian

Conclusiones

Estos ejercicios proporcionan un enfoque práctico para entender el funcionamiento del microcontrolador PIC16F877A, específicamente en la configuración y uso de la comunicación serie UART y el control de un display de 7 segmentos. La programación en ensamblador permite un control preciso del hardware, optimizando el uso de registros y recursos internos del microcontrolador.

La implementación de la comunicación serie con USART demuestra cómo los datos pueden enviarse y recibirse desde una PC, permitiendo la interacción con dispositivos externos. Este tipo de comunicación es ampliamente utilizado en sistemas embebidos para la transmisión de datos a sensores, módulos de comunicación y otros microcontroladores.

Además, el uso del display de 7 segmentos brinda experiencia en la gestión de salidas digitales, que es fundamental en sistemas de visualización como relojes, contadores y tableros de control. La decodificación de caracteres en el display a partir de los datos recibidos en serie refuerza la importancia del procesamiento de señales y la conversión de información en sistemas digitales.

La simulación en Proteus es una herramienta clave en el desarrollo de este tipo de proyectos, ya que permite probar y depurar el código sin necesidad de hardware físico. Esto facilita la identificación de errores en la lógica del programa, la verificación del funcionamiento del protocolo de comunicación y la correcta activación de los segmentos del display.

En general, este ejercicio es un excelente punto de partida para quienes desean profundizar en la arquitectura de microcontroladores PIC, programación en ensamblador y diseño de circuitos electrónicos con simulación virtual.

Bibliografía

Microchip Technology - Documentación oficial del PIC16F877A: https://www.microchip.com/en-us/product/PIC16F877A

MPLAB X IDE - Entorno de desarrollo para programar PICs: https://www.microchip.com/en-us/tools-resources/develop/mplab-x-ide

Manual de ensamblador para PIC - Guía sobre instrucciones y arquitectura: https://ww1.microchip.com/downloads/en/devicedoc/33023a.pdf

Programación en ensamblador para PIC - Cursos y ejemplos prácticos: https://www.todopic.com.ar