



Universidad Nacional Autónoma de México

Facultad de Ingeniería

Laboratorio de Microcomputadoras

Nombre del Profesor:

Amaranto de Jesús Dávila Jauregui

Semestre 2023-2

Grupo 3

Práctica 4.

Nombre del alumno:

- Castelan Ramos Carlos
- Corona Nava Pedro Jair

Fecha de entrega: 30/05/2023

# Práctica 4.

## Introducción:

La comunicación serial es una forma de transferir datos de manera secuencial, es decir uno a la vez, a través de un solo canal de comunicación. El PIC16F877A es un microcontrolador que incluye módulos de comunicación serial, lo que permite establecer comunicación con otros dispositivos mediante este protocolo.

En el caso del PIC16F877A, cuenta con dos módulos de comunicación serial: USART (Universal Synchronous Asynchronous Receiver Transmitter) y MSSP (Master Synchronous Serial Port). Estos módulos permiten la comunicación mediante los protocolos UART (Universal Asynchronous Receiver Transmitter), SPI (Serial Peripheral Interface) e I2C (Inter-Integrated Circuit).

Para lograr este tipo de comunicación se tienen que realizar ciertas configuraciones, las cuales son:

- Configuración de pines: Para utilizar la comunicación serial UART en el PIC16F877A, se deben asignar los pines adecuados para la transmisión (TX) y la recepción (RX). Generalmente, los pines RC6 (TX) y RC7 (RX) se utilizan para esta comunicación, pero también se pueden usar otros pines configurables.
- Configuración del registro SPBRG: El registro SPBRG determina la velocidad de transmisión de los datos. Por lo que se debe configurar este registro según la velocidad de baudios deseada para establecer la misma velocidad tanto en el PIC como en el dispositivo con el que se comunica.
- Configuración de los registros TXSTA y RCSTA: Estos registros controlan el funcionamiento del módulo USART y los modos de operación. Deben ser configurados adecuadamente para habilitar la comunicación serial UART, establecer el modo de transmisión y recepción, y configurar las interrupciones si son necesarias.
- Envío y recepción de datos: Para enviar datos, se puede cargar el byte a transmitir en el registro TXREG y esperar a que el bit TRMT (Transmit Shift Register Empty) se establezca antes de cargar el siguiente byte. Para recibir datos, se debe verificar el bit RCIF (Receive Complete Interrupt Flag) y luego leer el byte recibido del registro RCREG.

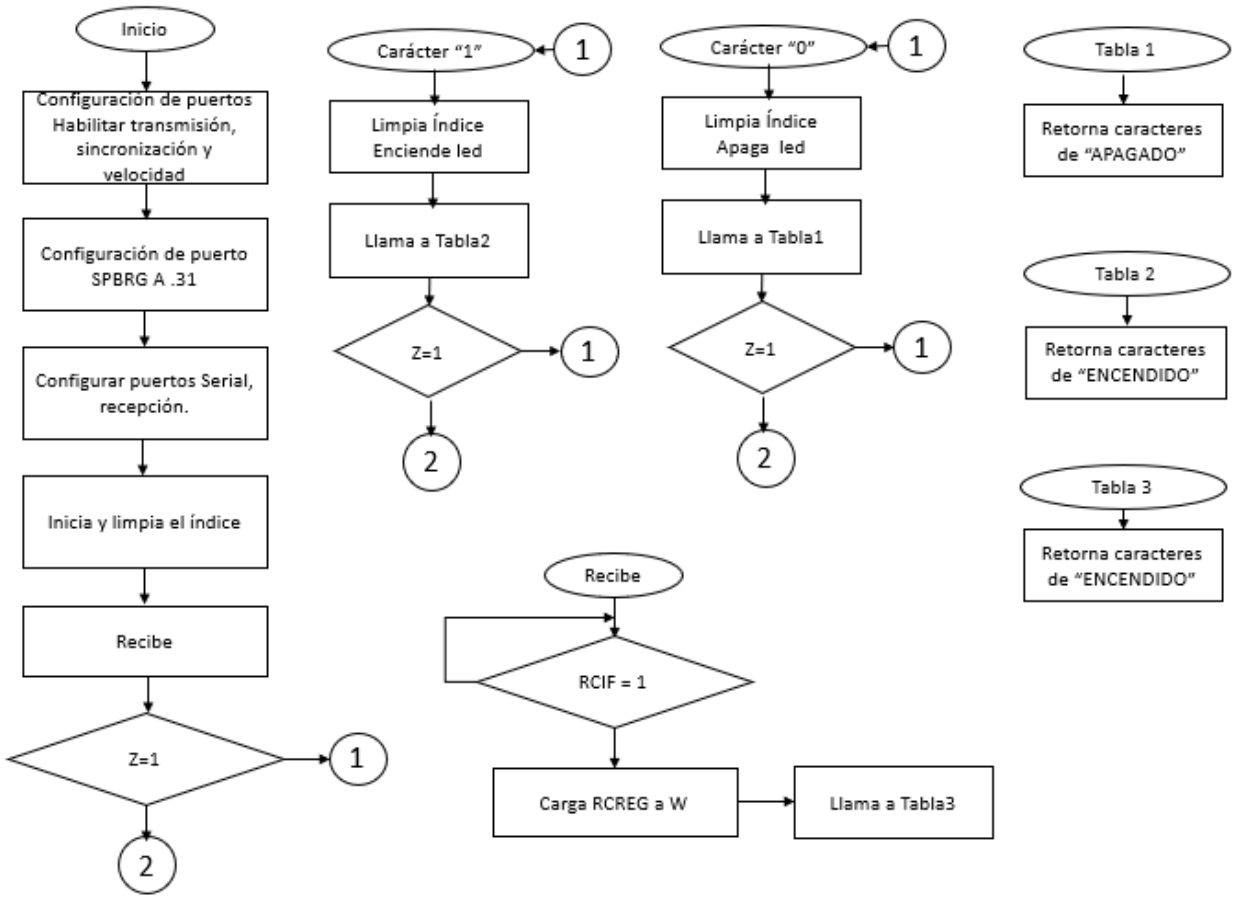
**Desarrollo:**

Ejercicio: .Diseñar un programa que reciba un carácter ASCII por la terminal serial, si el carácter es '1' encenderá el puerto B y enviará a través del puerto serial la palabra “Encendido” (se debe visualizar en la terminal. Si recibe '0' apagará el LED y enviará la palabra "Apagado", para cualquier otro caracter recibido solo envía “Error”.

**Propuesta de solución:**.

Para lograr esto emplearemos la comunicación serial.UART, por lo cual es necesario configurar de forma correcta los distintos registros que ésta involucra (TXREG, TRMT, RCREG, TXSTA, RCSTA, entre otros).

**Diagrama de flujo:**



## Código comentado:

```
processor PIC16f877A ; procesador
org 0x00                        ; vector de reset
Goto 0x05                               ; envia PC a la direccion 0x05
org 0x05                        ; origen del programa

#include <P16F877A.inc>

INDICE EQU 0X20 ; declaracion del Indice
DATO EQU 0X21   ; declaracion del auxiliar DATO
DATO_R EQU 0X22  ; declaracion del segundo auxiliar

BSF STATUS,RP0 ;Cambio de banco
BCF STATUS,RP1 ;Cambio de banco
BCF TRISC,RC6  ; Configramos RC6 como salida
BSF TXSTA,TXEN ; se habilita a transmision
BCF TXSTA,SYNC ;se desactiva la sincronizacion de la transmision
BCF TXSTA,BRGH ;se desactiva la alta velocidad de transmision

MOVLW .31 ; movemos el 31 decimal a w
MOVWF SPBRG ; movemos el valor de w al registro SPBRG

BCF STATUS,RP0 ; cambio al banco 0
BSF RCSTA,SPEN ;Habilitamos el puerto serial
BSF RCSTA,SREN ;Habilitamos la recepcion continua
BSF RCSTA,CREN ;Habilitamos la recepcion continua

INICIA ; Bandera INICIO
        CLRF INDICE ; Limpiamos el registro INDICE
REPITE ; Bandera REPITE
        CALL RECIBE ; Llamamos a la rutina RECIBE
        MOVWF DATO_R ; moemos el valor de w al auxiliar
        XORLW '1' ; Realizamos un XOR de W con 1
        BTFSS STATUS,Z ; verificamos si se encendio la bandera Z
        GOTO DOS? ; si no se encendio llamamos a la rutina DOS?
        CALL UNO ; si se encendio brinacmos a la rutina UNO
        GOTO INICIA ; Regresamos a INICIA

DOS? ; Bandera de DOS?
        MOVF DATO_R,W ; movemos el valor del auxiliar a W
        XORLW '2' ; realizamos un XOR de W con 2
        BTFSS STATUS,Z ; Verificamos si la bandera Z se encendio
        GOTO OTRO ; si no se encendio brinacmos a la rutina de error
        CALL DOS ; si si se encendio nos movemos a la rutina DOS
        GOTO INICIA ; Regresamos a INICIA


UNO ; Bandera UNO
        CLRF INDICE ; Limpiamos el registro INDICE
REGRESA ; Bandera REGRESA
        BSF PORTB, 0 ; Endcendemos el LED del PORTB
        MOVF INDICE,W ; Movemos el valor del auxiliar INDICE a W
        CALL TABLA2 ; Llamamos a la rutina que contiene el mensaje
        MOVWF DATO ; Movemos a W el valor de DATO
        XORLW '$' ; Verificamos si se llego al caracter de escape
        BTFSC STATUS,Z ; Checamos si la bandera Z esta apagada
        RETURN ; Regresamos
        MOVF DATO,W ; Movemos el valor de DATO a W
        CALL ENVIA ; Llamamos a la rutina para enviar los datos
        INCF INDICE ;Incrementamos el valor de INDICE
        GOTO REGRESA; Saltamos a REGRESA

DOS ; Bandera DOS
```

```
        CLRF INDICE ; Limpiamos el registro INDICE
REGRESA_DOS ; Bandera REFRESA_DOS
        BCF PORTB, 0 ; Apagamos el LED del PORTB
        MOVF INDICE,W ; Movemos el valor del auxiliar INDICE a W
        CALL TABLA1 ;Llamamos a la rutina que contiene el mensaje adecuado
        MOVWF DATO ; Movemos a W el valor de DATO
        XORLW '$' ; Verificamos si se llego al caracter de escape
        BTFSC STATUS,Z ; Verificamos si la bandera Z esta apagada
        RETURN ; Regresamos
        MOVF DATO,W ; Movemos el valor de DATO a W
        CALL ENVIA ; Llamamos a la rutina para enviar los datos
        INCF INDICE ;Incrementamos el valor de INDICE
        GOTO REGRESA_DOS; Saltamos a REGRESA_DOS

OTRO; Bandera de OTRO
        CLRF INDICE ; Limpiamos el registro INDICE
REGRESA_OTRO
        MOVF INDICE,W ; Movemos el valor del auxiliar INDICE a W
        CALL TABLA3 ;Llamamos a la rutina que contiene el mensaje adecuado
        MOVWF DATO ; Movemos a W el valor de DATO
        XORLW '$' ; Verificamos si se llego al caracter de escape
        BTFSC STATUS,Z ; Verificamos si la bandera Z esta apagada
        RETURN ; Regresamos
        MOVF DATO,W ; Movemos el valor de DATO a W
        CALL ENVIA ; Llamamos a la rutina para enviar los datos
        INCF INDICE ;Incrementamos el valor de INDICE
        GOTO REGRESA_OTRO; Saltamos a REGRESA_OTRO

TABLA1 ; Bandera de TABLA1
        ADDWF PCL,1 ;sumamos PCL con 1
        RETLW 'A' ; Cargamos y retornamos la letra A a W
        RETLW 'p' ; Cargamos y retornamos la letra p a W
        RETLW 'a' ; Cargamos y retornamos la letra a a W
        RETLW 'g' ; Cargamos y retornamos la letra g a W
        RETLW 'a' ; Cargamos y retornamos la letra a a W
        RETLW 'd' ; Cargamos y retornamos la letra d a W
        RETLW 'o' ; Cargamos y retornamos la letra o a W
        RETLW '\n' ; Cargamos y retornamos un salto de linea W
        RETLW '\r' ; Cargamos y retornamos un retorno de cursor a W
        RETLW '$' ; Cargamos y retornamos el caracter $ a W

TABLA2 ; Bandera de TABLA2
        ADDWF PCL,1 ; Sumamos 1 a PCL
        RETLW 'E' ; Cargamos y retornamos la letra E a W
        RETLW 'n' ; Cargamos y retornamos la letra n a W
        RETLW 'c' ; Cargamos y retornamos la letra c a W
        RETLW 'e' ; Cargamos y retornamos la letra e a W
        RETLW 'n' ; Cargamos y retornamos la letra n a W
        RETLW 'd' ; Cargamos y retornamos la letra d a W
        RETLW 'i' ; Cargamos y retornamos la letra i a W
        RETLW 'd' ; Cargamos y retornamos la letra d a W
        RETLW 'o' ; Cargamos y retornamos la letra o a W
        RETLW '\n' ; Cargamos y retornamos un salto de linea W
        RETLW '\r' ; Cargamos y retornamos un retorno de cursor a W
        RETLW '$' ; Cargamos y retornamos el caracter $ a W

TABLA3 ; Bandera para la TABLA3
        ADDWF PCL,1 ; sumamos PCL con 1
        RETLW 'E' ; Cargamos y retornamos la letra E a W
        RETLW 'r' ; Cargamos y retornamos la letra r a W
        RETLW 'r' ; Cargamos y retornamos la letra r a W
        RETLW 'o' ; Cargamos y retornamos la letra o a W
        RETLW 'r' ; Cargamos y retornamos la letra r a W
        RETLW '\n' ; Cargamos y retornamos un salto de linea W
        RETLW '\r' ; Cargamos y retornamos un retorno de cursor a W
        RETLW '$' ; Cargamos y retornamos el caracter $ a W
```

```
ENVIA ; Rutina de transmision
        BSF STATUS,RP0 ; cambio de banco
        BCF STATUS,RP1 ; cambio de banco
        BTFSS TXSTA,TRMT ;Verifica si el bit TRMT esta encendido
        GOTO $-1 ; regresamos una instrccion (generando un loop)

        BCF STATUS,RP0 ; cambio al banco 0
        MOVF DATO,W ; Cargamos el valor de DATO a W
        MOVWF TXREG ; Cargamos el valor de W al registro TXREG
        RETURN ; retornamos

RECIBE ; Rutina para la recepcion
        BTFSS PIR1,RCIF ; Verificamos si el bit RCIF esta en alto
        GOTO $-1 ; regresamos una instruccion
        MOVF RCREG,W ; Cargamos el valor de RCREG a W
        RETURN; Retornamos

END ; Fin del programa
```

**Explicación del código**

Como paso inicial se selecciona el microcontrolador con el que se desea trabajar, posteriormente se definen los registros auxiliares (**INDICE, DATO y DATO_R**). Después de esto configuramos los registros de la comunicación serial. Una vez realizado esto verificamos si el número ingresado en la terminal es 1, si si lo es se realizan los pasos necesarios para encender el PORTB y mostrar la palabra “Encendido”, si no es 1 se hace un brinco a una rutina que nos sirve para verificar si el dato ingresado es 0. Si el dato es igual a cero se hace el mismo proceso que para el uno pero en esta ocasión apagamos el LED del PORTB y mostramos en la terminal la palabra “Apagado”, si no es 1 ni 0 simplemente mostramos la palabra “Error” sin hacer modificación alguna al PORTB.

Para mostrar las palabras según sea el número ingresado contamos con una función que emplea comunicación serial y hace uso de los registros TXSTA y TXREG para lograr esto..

**Conclusiones:**

- Carlos Castelan Ramos: En esta práctica hicimos se hizo uso de los puertos seriales del PIC logrando configurarlos desde su transmisión, pasando por su sincronización así como su velocidad, así como la configuración de recepción, esto lo logramos a partir de una terminal virtual de proteus, así logramos implementar este código de encendido y apagado de leds donde observamos la posibilidad de usos de periféricos para la interpretación de información y aplicación de funcionalidades específicas en los PICS.

- Corona Nava Pedro Jair: Con la realización de esta práctica me quedó más clara la utilidad que tiene la comunicación serial UART. ya es una característica valiosa con la que cuenta el microcontrolador con el que trabajamos debido a que permite establecer conexiones confiables y flexibles con otros dispositivos. Además considero que con la configuración adecuada y la comprensión de los protocolos y modos de operación podemos lograr aplicaciones prácticas y eficientes, lo que nos puede ser de gran ayuda en futuros proyectos ya sea académicos o en el ámbito laboral. Por todas estas razones considero que se cubrieron los objetivos propuestos al inicio de la práctica y logramos realizar todas las actividades en tiempo y forma. Concluyendo así la práctica de forma satisfactoria.