**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

**FACULTAD DE INGENIERÍA**

**LABORATORIO DE MICROCOMPUTADORAS**

PRÁCTICA 7:

**PUERTIO SERIE SCI (ASÍNCRONO)**

**GRUPO:** 12

**PROFESORA**: M.I LOURDES ANGÉLICA QUIÑONES JUAREZ

**ALUMNO:** CHÁVEZ DELGADO JORGE LUIS

**N° DE CUENTA:** 312217493

**FECHA DE ASIGNACIÓN: FECHA DE ENTREGA:**

*31/03/17* 04*/04/17*

PRÁCTICA 7: Puerto Serie SCI (ASÍNCRONO)

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

**Ejercicio 1:** En la realización de este ejercicio aprendimos a configurar la comunicación serie asíncrona, así como su modulo USAR para la transmisión y recepción de datos, y lo aplicamos al ejercicio dos de la práctica tres del manual de prácticas, las teclas que leímos correspondían a cada acción o caso de los ya programados de la manera siguiente:

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

**Código:**

**processor 16f877**

**include <p16f877.inc>**

**Y equ H'24' ;Asignamos la memoria a Y**

**contador equ h'20'**

**valor1 equ h'21'**

**valor2 equ h'22'**

**valor3 equ h'23'**

**;Constantes para el retardo**

**cte1 equ 20h**

**cte2 equ 50h**

**cte3 equ 60h**

**org 0**

**goto inicio**

**org 5**

**inicio**

**clrf PORTB ;Limpiamos el puerto B**

**BSF STATUS,RP0 ;Cambiamos al banco 1**

**BCF STATUS,RP1**

**clrf TRISB ;Configuramos el puerto B como salida**

**trans:**

**bsf TXSTA, BRGH ; Para la velocidad de transmisión alta BRGH=1**

**movlw D'32' ;Este es el valor decimal de acuerdo baud rate =38400 y Fosc =20 Mhz**

**movwf SPBRG**

**bcf TXSTA, SYNC ;Para la comunicación serie asíncrona**

**bsf TXSTA, TXEN ;Habilitamos el transmisor**

**bcf STATUS, RP0**

**bsf RCSTA, SPEN ; Habilitamos el puerto serie**

**bsf RCSTA, CREN ;Dejamos disponibles TX y RX**

**recibe**

**btfss PIR1, RCIF ;Bandera de interrupción se activan cuando ocurre una interrupcion y termina la recepción**

**goto recibe ;Vamos a la subrutina recibe**

**movf RCREG, W ;si se pone en 1 RCIF, se guarda el resultado en RCREG la lectura del teclado**

**movwf Y ;Movemos el contenido de W a la variable Y**

**ciclo**

**clrf PORTB ;Limpiamos PORTB**

**bcf STATUS, Z ;Ponemos a 0 a la bandera Z**

**movf Y,W ;Movemos lo que tiene Y a W**

**xorlw A'0' ;ASCII que realiza la operación lógica xor entre la entrada del teclado y w**

**btfsc STATUS,Z ;si z=0 saltamos al caso 1**

**goto caso1**

**movf Y,W ;Movemos el contenido de Y a W**

**xorlw A'1' ;ASCII que realiza la operación lógica entre la entrada del teclado y w**

**btfsc STATUS,Z ;si z=0 saltamos a la subrutina caso2**

**goto caso2**

**movf Y,W ;Movemos el contenido de Y a W**

**xorlw A'2' ;ASCII que realiza la operación lógica entre la entrada del teclado y w**

**btfsc STATUS,Z ;si z=0 saltamos a la subrutina caso3**

**goto caso3**

**movf Y,W ;Movemos el contenido de Y a W**

**xorlw A'3' ;ASCII que realiza la operación lógica entre la entrada del teclado y w**

**btfsc STATUS,Z ;si z=0 saltamos a la subrutina caso4**

**goto caso4**

**movf Y,W ;Movemos el contenido de Y a W**

**xorlw A'4' ;ASCII que realiza la operación lógica entre la entrada del teclado y w**

**btfsc STATUS,Z ;si z=0 saltamos a la subrutina caso5**

**goto caso5**

**movf Y,W ;Movemos el contenido de Y a W**

**xorlw A'5' ;ASCII que realiza la operación lógica entre la entrada del teclado y w**

**btfsc STATUS,Z ;si z=0 saltamos a la subrutina caso6**

**goto caso6**

**recepcion**

**movf RCREG, W ;Movemos el contenido del registro de recepción RCREG a W**

**movwf TXREG ;Movemos el contenido de W al registro de transmisión TXREG**

**bsf STATUS, RP0 ;Nos cambiamos al banco 1**

**transmite**

**btfss TXSTA, TRMT ; Para verificar si se realizo la transmisión 1=TSR vacio**

**goto transmite ;saltamos a la subrutina transmite 0 =TSR lleno**

**bcf STATUS, RP0 ;Nos cambiamos al banco 0**

**goto recibe ;Saltamos a la subrutina recibe**

**caso1 ;dato 000, acción LED’S apagados**

**bcf STATUS,0 ;limpia el bit 0 del registro STATUS, evita que se prendan dos LED’S**

**clrf PORTB ;limpia el puerto B, LED’S apagados**

**goto recepcion ;Saltamos a la subrutina recepción**

**caso2 ;dato 001, acción LED’S encendidos**

**bcf STATUS,0 ;limpia el bit 0 del registro STATUS, evita que se prendan dos LED’S**

**movlw h'FF' ;Movemos un 1 al registro W**

**movwf PORTB ; Movemos el contenido de W a PORTB , enciende LED’S**

**goto recepcion ;salto a la subrutina recepción**

**caso3 ;dato 010, acción corrimiento de bit hacia la derecha**

**bcf STATUS,0 ;limpia el bit 0 del registro STATUS, evita que se prendan dos LED’S**

**movlw H'80' ;Movemos H’80’ a W**

**movwf PORTB ;Movemos el contenido de W a PORTB, fin de secuencia**

**call retardo ;Llamamos a la subrutina retardo**

**movlw H’08’' ;Movemos H‘08’ a W**

**movwf L ;Movemos el contenido de W a L, inicia secuencia**

**corrimiento:**

**call retardo ;Llamamos a la subrutina retardo**

**rrf PORTB,1 ;corrimiento hacia la derecha de bits de 1 en 1 sobre el puerto B**

**decf L,1 ;Decrementamos a L en 1**

**btfsc STATUS,2 ; cuando se esta en cero estado que nos brinda STATUS**

**goto recepcion ;salto a la subrutina recepción**

**goto corrimiento ;no hemos llegado a cero, seguimos con el corrimiento**

**caso4 ;dato 011, acción corrimiento de bit hacia la izquierda**

**bcf STATUS,0 ;limpia el bit 0 del registro STATUS, evita que se prendan dos LED’S**

**movlw H'01' ;Movemos H’80’ a W**

**movwf PORTB ;Movemos el contenido de W a PORTB, fin de secuencia**

**call retardo ;Llamamos a la subrutina retardo**

**movlw H’08’' ;Movemos H‘08’ a W**

**movwf L ;Movemos el contenido de W a L, inicia secuencia**

**corrimiento2:**

**call retardo ;Llamamos a la subrutina retardo**

**rlf PORTB,1 ;corrimiento hacia la derecha de bits de 1 en 1 sobre el puerto B**

**decf L,1 ;Decrementamos a L en 1**

**btfsc STATUS,2 ; Si el bit 2 es cero saltamos a...**

**goto recepcion ;salto a la subrutina recepción**

**goto corrimiento2 ;no hemos llegado a cero, seguimos con el corrimiento**

**caso5 ;dato 100, acción corrimiento de bit hacia la derecha y a la izquierda**

**bcf STATUS,0 ;limpia el bit 0 del registro STATUS, evita que se prendan dos LED’S**

**movlw H'80' ;Movemos H’80’ a W**

**movwf PORTB ;Movemos el contenido de W a PORTB, fin de secuencia**

**call retardo ;Llamamos a la subrutina retardo**

**movlw H’08’' ;Movemos H‘08’ a W**

**movwf L ;Movemos el contenido de W a L, inicia secuencia**

**corrimiento3:**

**call retardo ;Llamamos a la subrutina retardo**

**rrf PORTB,1 ;corrimiento hacia la derecha de bits de 1 en 1 sobre el puerto B**

**decf L,1 ;Decrementamos a L en 1**

**btfsc STATUS,2 ; cuando se esta en cero estado que nos brinda STATUS**

**goto caso4 ;salto a la subrutina corrimiento a la izquierda caso4**

**goto corrimiento3 ;no hemos llegado a cero, seguimos con el corrimiento**

**caso6 ;dato 101, acción se apagan y prenden todos los LED’S**

**movlw b'11111111' ;Movemos b’11111111’ a W**

**movwf PORTB ;Movemos el contenido de W a PORTB, para encender LED’S**

**call retardo ;Llamamos a la subrutina retardo**

**clrf PORTB ;se limpia el puerto B para que se apaguen los leds**

**call retardo ;Llamamos a la subrutina retardo**

**goto recepcion ;salto a la subrutina recepción**

**retardo**

**movlw cte1 ;Movemos el contenido del cte1 a W**

**movwf valor1 ;Movemos el contenido de w a la dirección de valor1**

**tresillos**

**movlw cte2 ;Movemos el contenido del cte2 a W**

**movwf valor2 ;Movemos el contenido de w a a la dirección de valor2**

**dos**

**movlw cte3 ;Movemos el contenido del cte3 a W**

**movwf valor3 ;Movemos el contenido de w a a la dirección de valor3**

**uno**

**decfsz valor3 ;Decrementamos a valor3 y escapa a uno cuando llegue a 0**

**goto uno ;saltamos a la dirección de uno para iniciar de nuevo**

**decfsz valor2 ;Decrementamos a valor2 y escapa a dos cuando llegue a 0**

**goto dos ;Saltamos a la direccion de dos para iniciar de nuevo**

**decfsz valor1 ;decrementa a valor1 y escapa a tres cuando llegue a 0**

**goto tres ;saltamos a la direccion de tres para iniciar de nuevo**

**return ;retorno a la subrutina retardo**

**end**

**Ejercicio 2:**

Ya habiendo comprendido la configuración de la comunicación serie asíncrona y resuelto los conflictos del primer ejercicio, procedimos a realizar este ejercicio en el cual nos dimos cuenta que utilizando el código ascii y la terminal pudimos mandar ese dato al display de siete segmentos, que en este caso fueron las vocales tanto mayúsculas como minúsculas.

**Código:**

**processor 16f877**

**include<p16f877.inc>**

**;Variables para el Retardo**

**valor1 equ h'21'**

**valor2 equ h'22'**

**valor3 equ h'23'**

**;Configuración DISPLAY SIETE SEGMENTOS**

**;Para las vocales mayúsculas**

**AM EQU b'01110111'**

**EM EQU b'01111001'**

**IM EQU b'00000110'**

**OM EQU b'00111111'**

**UM EQU b'00111110'**

**;Para las vocales minúsculas**

**Am EQU b'11011111'**

**Em EQU b'11111011'**

**Im EQU b'10000100'**

**Om EQU b'11011100**

**Um EQU b'10011100'**

**;Para cargar valores usaremos var**

**var equ h'24'**

**org 0h;**

**goto INICIO**

**org 05h**

**INICIO:**

**clrf PORTB**

**bsf STATUS,RP0**

**bcf STATUS,RP1 ;Cambiamos al Banco 1**

**clrf TRISB ;Configura el puerto B como salida.**

**;Configuración del registro transmisor**

**bsf TXSTA, BRGH ;Bit de selección de velocidad alta,BRGH=1**

**movlw d'32' ;Velocidad 38400 baud**

**movwf SPBRG ;Cargamos la velocidad de comunicación**

**bcf TXSTA, SYNC ;Comunicación asincrona SYNC=0**

**bsf TXSTA, TXEN ;Activa la transmisión**

**bcf STATUS, RP0 ;Cambiamos al Banco 0**

**;Configuración del registro receptor**

**bsf RCSTA, SPEN ;Habilita puerto serie**

**bsf RCSTA, CREN ;Configura recepción continua en modo**

**;asíncrono**

**RECIBE:**

**;Registros Banderas**

**btfss PIR1, RCIF ;Verificamos si la recepción está completa**

**goto RECIBE ;Si no, sigue recibiendo**

**movf RCREG,w ;Si, mueve lo que recibe registro de**

**;recepción RCREG a W**

**CICLO:**

**clrf PORTB**

**;Verificamos si es "a"**

**movlw 'a' ;Mueve 'a' a W**

**movwf var ;Mueve el contenido de W a var, var=a**

**movfw RCREG ;Mueve el contenido de RCREG a W**

**xorwf var,w ;Realiza var xor W**

**btfsc STATUS,Z ;Verificamos si Z=0**

**goto a ;NO, son iguales**

**;SI, verifica la siguiente opción**

**;Verificamos si es "e"**

**movlw 'e'**

**movwf var**

**movfw RCREG**

**xorwf var,w**

**btfsc STATUS,Z**

**goto e**

**;Verificamos si es "i"**

**movlw 'i'**

**movwf var**

**movfw RCREG**

**xorwf var,w**

**btfsc STATUS,Z**

**goto i**

**;Verificamos si es "o"**

**movlw 'o'**

**movwf var**

**movfw RCREG**

**xorwf var,w**

**btfsc STATUS,Z**

**goto o**

**;Verificamos si es "u"**

**movlw 'u'**

**movwf var**

**movfw RCREG**

**xorwf var,w**

**btfsc STATUS,Z**

**goto u**

**;Verificamos si es "A"**

**movlw 'A'**

**movwf var**

**movfw RCREG**

**xorwf var,w**

**btfsc STATUS,Z**

**goto A**

**;Verificamos si es "E"**

**movlw 'E'**

**movwf var**

**movfw RCREG**

**xorwf var,w**

**btfsc STATUS,Z**

**goto E**

**;Verificamos si es "I"**

**movlw 'I'**

**movwf var**

**movfw RCREG**

**xorwf var,w**

**btfsc STATUS,Z**

**goto I**

**;Verificamos si es "O"**

**movlw 'O'**

**movwf var**

**movfw RCREG**

**xorwf var,w**

**btfsc STATUS,Z**

**goto O**

**;Verificamos si es "U"**

**movlw 'U'**

**movwf var**

**movfw RCREG**

**xorwf var,w**

**btfsc STATUS,Z**

**goto U**

**;Mostramos vocales minúsculas en el DISPLAY**

**a:**

**movlw Am ;Mueve el valor de Am a W**

**movwf PORTB ;Mueve el valor de W a PORTB**

**call retardo ;Tiempo de Retardo**

**goto CICLO**

**e:**

**movlw Em**

**movwf PORTB**

**call retardo**

**goto CICLO**

**i:**

**movlw Im**

**movwf PORTB**

**call retardo**

**goto CICLO**

**o:**

**movlw Om**

**movwf PORTB**

**call retardo**

**goto CICLO**

**u:**

**movlw Um**

**movwf PORTB**

**call retardo**

**goto CICLO**

**;Mostramos vocales mayúsculas en el DISPLAY**

**A:**

**movlw AM**

**movwf PORTB**

**call retardo**

**goto CICLO**

**E:**

**movlw EM**

**movwf PORTB**

**call retardo**

**goto CICLO**

**I:**

**movlw IM**

**movwf PORTB**

**call retardo**

**goto CICLO**

**O:**

**movlw OM**

**movwf PORTB**

**call retardo**

**goto CICLO**

**U:**

**movlw UM**

**movwf PORTB**

**call retardo**

**goto CICLO**

**retardo:**

**movlw h'10' ;RRetardo**

**movwf valor1**

**tres movlw h'50'**

**movwf valor2**

**dos movlw h'60'**

**movwf valor3**

**uno decfsz valor3**

**goto uno**

**decfsz valor2**

**goto dos**

**decfsz valor1**

**goto tres**

**return**

**end**

**Conclusiones:**

De esta práctica, se puede concluir que la comunicación serie asíncrona es de mucha utilidad a la hora de transmitir y recibir datos de un teclado, ya que nos facilita el uso del código ascii para ambos casos.

Se cumplió satisfactoriamente el objetivo de la práctica, pese a algunos problemas que tuvimos en el ejercicio 1 en cuanto a algunas subrutinas loop, pero finalmente localizamos el error y pudimos continuar con éxito el ejercicio número 2. En cuanto al ejercicio 2 es impresionante ver como de un código se puede transmitir por un cable el dato de una letra para que pueda visualizarse en el display de siete segmentos, esto puede tener muchas aplicaciones como por ejemplo los termometros de los refrigeradores, o medir el kilometraje de un auto e inclusive una caja registradora.

Finalmente reuniendo lo visto en las anteriores prácticas y esta en especial, puedo concluir que el uso de los pics así como de sus componentes es esencial ya que se puede realizar cualquier tipo de proyecto para cualquier ambito al que queramos referirnos.