Universidad Nacional Autónoma de México

Facultad de Ingeniería

Semestre 2023-2

Laboratorio de Microcomputadoras

Práctica 7:

Puerto Serie SCI (Asíncrono)

Profesor:

M.I. Ruben Anaya García

Alumnos:	Número de cuenta:
Barreiro Valdez Alejandro	317520888
Zepeda Baeza Jessica	317520747

Grupo Laboratorio: 4

Grupo Teoría: 1

Fecha de realización: 24 de mayo de 2023

Fecha de entrega: 30 de mayo de 2023

Desarrollo

Para esta práctica se buscó crear programas en ensamblador que implementaran la comunicación asíncrona mediante las terminales, registros y banderas que incluye el microcontrolador PIC16F877. Para ello, se entendió que lo primero a definir es la tasa de transferencia y los diferentes valores con los que se puede configurar. También se abordaron los registros, junto con sus banderas, que permiten la transmisión como TXSTA (donde TXEN habilita la transmisión, SYNC indica si es comunicación asíncrona, BRGH indica la selección de velocidad de baudios, TRMT indica si el dato se ha transmitido, etc.) y TXREG (donde se guarda el dato a transmitir). También están los registros que permiten la recepción como RCSTA (donde SPEN habilita el puerto serie, CREN configura la recepción contínua, etc.) y RCREG (donde se guarda el dato recibido). Por último se tiene el registro PIR1 que contiene las banderas RCIF y TXIF que indican si se completó la recepción o transmisión respectivamente.

Los programas desarrollados fueron:

1. Un programa que recibe un dato y transmite de vuelta el mismo dato, además de mandarlo como salida al Puerto B.
2. Un programa que muestra “HOLA UNAM” en la terminal.
3. Un programa que recibe un dato a través del puerto serie y prende la terminal 0 del Puerto B si ese dato es 1 y apaga dicha terminal si el dato es 0.
4. Un programa que recibe un dato a través del puerto serie y si ese dato es ‘d’ o ‘D’ hace un corrimiento de bits a la derecha en el Puerto B empezando con 10000000 y terminando con 00000001. Si el dato es ‘I’ o ‘i’ hace el corrimiento de bits a la izquierda en el Puerto B empezando con 00000001 y terminando con 10000000.
5. Un programa que recibe un dato a través del puerto serie mediante una conexión inalámbrica. Y dependiendo el valor de ese dato, controla motores como se muestra en la tabla:

Comando Puerto serie	**ACCION**	
	MOTOR M1	**MOTOR M2**
‘S’	PARO	PARO
‘A’	DERECHA	DERECHA
‘T’	IZQUIERDA	IZQUIERDA
‘D’	DERECHA	IZQUIERDA
‘I’	IZQUIERDA	DERECHA

Tabla 7.4 Control de motores, comunicación serie

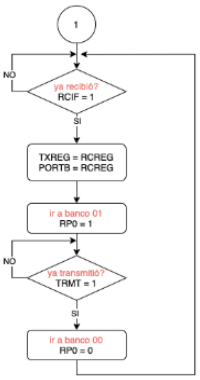
6. Un programa que convierte la entrada analógica de un termómetro, calcula el valor de la temperatura según la variación del sensor y muestra dicho valor en la terminal mediante el puerto serie.

Algoritmos

Para casi todos los programas realizados se tiene una configuración inicial cuya carta ASM es la siguiente. En ella se accede al banco 1 para establecer al Puerto B como salida, se define una tasa de alta velocidad con de 9600 de Baud Rate. También se activa el modo asíncrono y se habilita la transmisión. En el banco 0, se habilita el puerto serie y la recepción de datos.

Programa 1 (ejercicio 1 y 2)

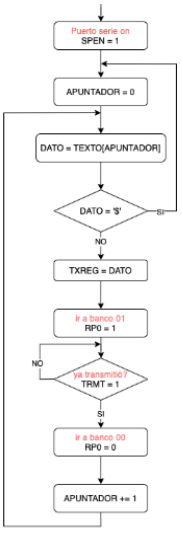
Para el ejercicio 1 y 2 se realizó la siguiente carta ASM.



Después de hacer las configuraciones iniciales, se espera a recibir un dato. Una vez que se recibe, se pasa el dato tanto al registro donde se ubica el dato a transmitir como al Puerto B. Se cambia de banco para mandar el dato y se espera hasta que la bandera indique que la transmisión se completó. Por último regresa al banco 0 para esperar a recibir otro dato y repetir todo el proceso.

Programa 2 (ejercicio 3)

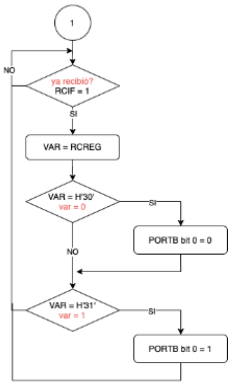
Para el segundo programa se realizó la siguiente carta ASM:



Se declara un apuntador que guardará la posición de cada caracter del texto a mostrar y se inicializa en 0. Después se obtiene dicho caracter y se guarda en una variable DATO. Esta variable se compara con '$' ya que este último es el caracter de fin de cadena. En caso de que DATO y '$' sean diferentes, se pasa DATO a TXREG y se transmite. Se verifica que ya se transmitió desde el banco 1 y si aún no se completa la transmisión, se espera y vuelve a verificar. Después se cambia al banco 0 para incrementar en 1 el apuntador y obtener el nuevo caracter a transmitir. En caso de que DATO sea igual a '$', el apuntador vuelve a ser 0 para volver a transmitir la cadena desde el inicio.

Programa 3 (ejercicio 4)

Para el tercer programa se realizó la siguiente carta ASM:

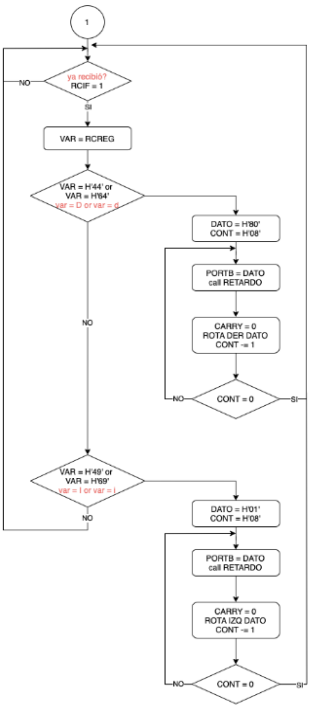


Para este programa primero se verifica si ya recibió un dato, en caso de no ser así vuelve a verificar hasta que se reciba. Una vez recibido, se guarda en una variable que después verifica si su valor es igual al valor ASCII de 0. En caso de ser así, la terminal 0 del Puerto B se vuelve 0. En caso contrario, se verifica si el valor de la variable es igual al valor ASCII de 1 y si es verdadero, la terminal 0 del Puerto B se vuelve 1. Si la variable no es 0 ni 1 o al terminar de hacer las comparaciones y asignaciones, el programa regresa al inicio a esperar recibir otro dato.

Programa 4 (ejercicio 5)

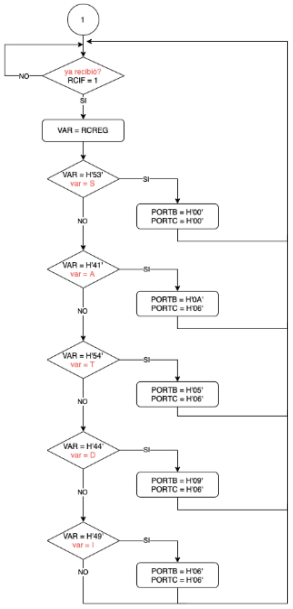
Para el cuarto programa se realizó algo parecido al programa anterior con la diferencia de que compara el valor de la variable con 'D','d','I' e 'i'. En caso de ser 'D' o 'd', inicializa un contador en 8 y una variable DATO con su bit más significativo encendido. Después pasa esa variable al puerto B y llama un retardo para que se muestre por ½ segundo. Después hace 0 al Carry para que no afecte la rotación y rota DATO a la derecha. Decrementa en 1 el contador y lo compara con 0. En caso de ser diferentes, regresa a mostrar el dato en el Puerto B y rotar. En caso de ser iguales, regresa el programa a esperar recibir una nueva

variable. En caso de que la variable recibida sea ‘I’ o ‘i’ hace lo mismo con la diferencia de que DATO se inicializa con el bit menos significativo encendido y la rotación se hace a la izquierda. De igual forma, al terminar las 8 rotaciones, regresa a esperar recibir un nuevo dato. En caso de que la variable no sea ‘D’,’d’,’I’ ni ‘i’, espera a recibir un nuevo dato. La carta ASM del programa es la siguiente:



Programa 5 (ejercicio 6)

Para el quinto programa también se espera a recibir un dato que se guarda en una variable. Después se compara el valor de la variable con el ASCII de ‘S’,’A’,’T’,’I’,’D’. En caso de no ser ninguno de estos caracteres, vuelve a esperar a recibir un nuevo dato. En caso de ser alguna de las letras antes mencionadas, se modifica el Puerto B y C para controlar los motores y regresa a recibir un nuevo dato. En caso de que la variable sea ‘S’, el Puerto B y C toman el valor de 0 para que los motores no se muevan. En caso de que sea ‘A’, el Puerto B es A y Puerto C es 6 para hacer que los motores vayan a la misma dirección. En caso de que sea ‘T’, el Puerto B es 5 y Puerto C es 6 para que los motores vayan en la misma dirección pero al revés de ‘A’. En caso de que sea ‘I’, el Puerto B y Puerto C son 6 para hacer que gire a la izquierda con las llantas moviéndose en dirección contraria. Y en caso de que sea ‘D’, el Puerto B es 9 y el Puerto C es 6 para hacer que gire a la derecha con las llantas moviéndose en dirección contraria pero inverso al movimiento en ‘I’. La carta ASM realizada se muestra a continuación.



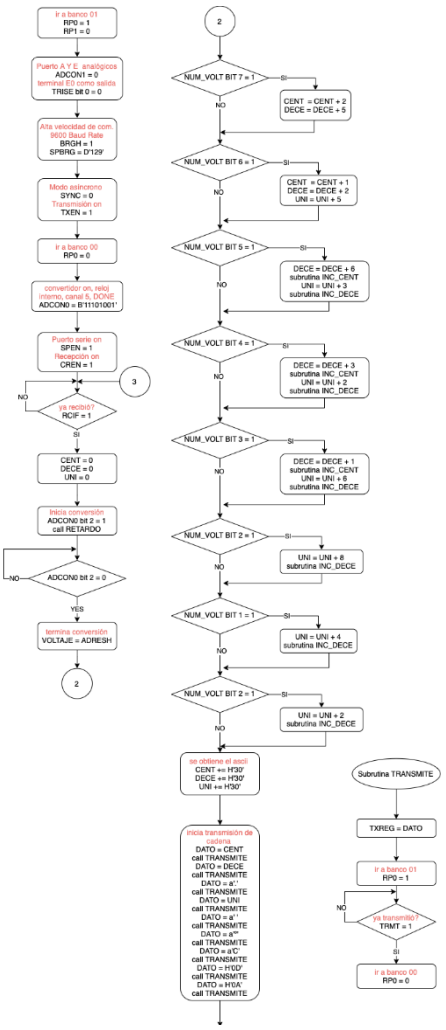
Programa 6 (ejercicio 7)

Para el último programa se utilizó el convertidor analógico-digital por lo que se agregaron configuraciones iniciales: se definieron los puertos A y E como analógicos (ADCON0), se habilitó el convertidor, se indicó que se usará el reloj interno y el canal 5 (ADCON1).

El código inicia cuando se recibe un dato así que espera a que RCIF sea igual a 1. Se declaran tres variables CENT, DECE y UNI que guardarán el valor en decimal de la temperatura y se inicializan en 0. Después se prende el bit 2 de ADCON1 para iniciar la conversión y se espera hasta que dicho bit vuelva a ser 0. Cuando termina la conversión se pasa el valor a una variable VOLTAJE y se va verificando si está encendido cada bit de dicha variable para sumarle el respectivo voltaje asociado a ese bit. Por ejemplo si está prendido el bit más significativo se suma 2 a CENT y 5 a DECE ya que ese bit indica 2.5 V. Cuando se modifica DECE o UNI se manda a llamar una subrutina que verifica su valor ya que en caso de ser A o mayores, se le suma uno a la siguiente unidad y se modifica el valor de la actual; esto para que se queden los valores siempre en un formato decimal.

Una vez que se verificaron todos los bits de VOLTAJE, se le suma 30 a DECE, CENT y UNI para obtener el ASCII de ese número. Y se van transmitiendo uno por uno los caracteres formando la cadena: [CENT][DECE].[UNI] ºC. Para transmitir, se pasa el dato a TXREG y se verifica que TRMT sea 1. Cuando se transmiten todos los datos, el programa regresa a esperar recibir un nuevo dato para volver a obtener una temperatura.

La carta ASM realizada se muestra a continuación.



Programas comentados

Ejercicio 1

```
processor 16f877
include<p16f877.inc>

    ORG 0
    GOTO inicio
    ORG 5

inicio
    BSF STATUS,RP0
    BCF STATUS,RP1      ;Se cambia al banco 01
    CLRF TRISB          ;Puerto B como salida
    BSF TXSTA,BRGH      ;Cambia bandera BRGH
    MOVLW D'129'
    MOVWF SPBRG         ;Baud rate = 9600
    BCF TXSTA,SYNC      ;Comunicación síncrona
    BSF TXSTA,TXEN      ;Activa transmisión

    BCF STATUS,RP0      ;Se cambia al banco 00

    BSF RCSTA,SPEN      ;Habilita puerto serie
    BSF RCSTA,CREN      ;Recepción sencilla

RECIBE:                 ;Recepción de un dato
    BTFSS PIR1,RCIF     ;Bandera recepción completa
    GOTO RECIBE         ;Esperar a la información

    MOVF RCREG,W
    MOVWF TXREG         ;Transmisión = Recepción
    MOVWF PORTB         ;Salida = Transmisión

    BSF STATUS,RP0      ;Se cambia al banco 01

TRASMITE:               ;Transmisión de dato
    BTFSS TXSTA,TRMT    ;Bandera transmisión completa
    GOTO TRASMITE       ;Esperar transmisión
```

```
    BCF STATUS,RP0      ;Se cambia al banco 00
    GOTO RECIBE         ;Volver a recibir

END
```

Ejercicio 3

```
processor 16f877
include<p16f877.inc>

APUN EQU 0X20
DATO EQU 0X21

;Valores para el retardo
valor1 equ h'21
valor2 equ h'22
valor3 equ h'23
cte1 equ 20h
cte2 equ 50h
cte3 equ 60h

    ORG 0
    GOTO inicio
    ORG 5

inicio
    BSF STATUS,RP0
    BCF STATUS,RP1      ;Se cambia al banco 01
    CLRF TRISB          ;Puerto B como salida
    BSF TXSTA,BRGH      ;Cambia bandera BRGH
    MOVLW D'129'
    MOVWF SPBRG         ;Baud rate = 9600
    BCF TXSTA,SYNC      ;Comunicación síncrona
    BSF TXSTA,TXEN      ;Activa transmisión

    BCF STATUS,RP0      ;Se cambia al banco 00

    BSF RCSTA,SPEN      ;Habilita puerto serie
    BSF RCSTA,CREN      ;Recepción sencilla

REP
    CLRF APUN           ;Apuntador = 0

CICLO
    CALL TEXTO          ;Se apunta al texto
    MOVWF DATO
    SUBLW "$"
    BTFSC STATUS,Z      ;Verificando que no se haya llegado al final
    GOTO REP            ;Si es el final se repite
    CALL TRANSMITE      ;Si no es el final se transmite
    INCF APUN,1         ;Se apunta a la siguiente letra
    GOTO CICLO          ;Se repite el ciclo

TRANSMITE:
    MOVF DATO,W
    MOVWF TXREG         ;Transmisión = Dato
TRA
    BSF STATUS,RP0      ;Se cambia al banco 01
ESP
    BTFSS TXSTA,TRMT    ;Bandera de transmisión
    GOTO ESP            ;Esperar transmisión
    CALL retardo        ;Se llama el retardo
    BCF STATUS,RP0      ;Se cambia al banco 00
    RETURN

TEXTO
    MOVF APUN,W
    ADDWF PCL,1     ;Se mueve a la siguiente letra del texto
    DT "HOLA UNAM",0X0D,0X0A,"$" ; Texto a mostrar

;Subrutina para el retardo
retardo
    MOVLW cte1
    MOVWF valor1
tres
    MOVLW cte2
    MOVWF valor2
dos
    MOVLW cte3
    MOVWF valor3
uno
    DECFSZ valor3
    GOTO uno
    DECFSZ valor2
    GOTO dos
    DECFSZ valor1
    GOTO tres
    RETURN

END
```

Ejercicio 4

```
processor 16f877
include<p16f877.inc>

VAR EQU 0X20 ;Variable de control

    ORG 0
    GOTO inicio
    ORG 5

inicio
    BSF STATUS,RP0
    BCF STATUS,RP1      ;Se cambia al banco 01
    CLRF TRISB          ;Puerto B como salida
    BSF TXSTA,BRGH      ;Cambia bandera BRGH
    MOVLW D'129'
    MOVWF SPBRG         ;Baud rate = 9600
    BCF TXSTA,SYNC      ;Comunicación síncrona
    BSF TXSTA,TXEN      ;Activa transmisión

    BCF STATUS,RP0      ;Se cambia al banco 00

    BSF RCSTA,SPEN      ;Habilita puerto serie
    BSF RCSTA,CREN      ;Recepción sencilla

RECIBE:
    BTFSS PIR1,RCIF     ;Bandera de recepción completa
    GOTO RECIBE         ;Esperar recepción

    MOVF RCREG,W
    MOVWF VAR           ;Control = recepción

    MOVLW 0x30
    SUBWF VAR,W         ;Control == 0x30??
    BTFSC STATUS,Z      ;Si son iguales ->
    BCF PORTB,0         ;Salida = 0

    MOVLW 0x31
    SUBWF VAR,W         ;Control == 0x31??
    BTFSC STATUS,Z      ;Si son iguales ->
    BSF PORTB,0         ;Salida = 1

    GOTO RECIBE         ;Recibir otro dato

END
```

Ejercicio 5

```
processor 16f877
include<p16f877.inc>

VAR EQU 0X20        ;Variable de control
DATO EQU 0X24       ;Dato de salida

;Valores para el retardo
valor1 equ h'21
valor2 equ h'22
valor3 equ h'23
cte1 equ 20h
cte2 equ 50h
cte3 equ 60h


CONT equ 0X25 ;Contador para número de rotaciones

    ORG 0
    GOTO inicio
    ORG 5

inicio
    BSF STATUS,RP0
    BCF STATUS,RP1      ;Se cambia al banco 01
    CLRF TRISB          ;Puerto B como salida
    BSF TXSTA,BRGH      ;Cambia bandera BRGH
    MOVLW D'129'
    MOVWF SPBRG         ;Baud rate = 9600
    BCF TXSTA,SYNC      ;Comunicación síncrona
    BSF TXSTA,TXEN      ;Activa transmisión

    BCF STATUS,RP0      ;Se cambia al banco 00

    BSF RCSTA,SPEN      ;Habilita puerto serie
    BSF RCSTA,CREN      ;Recepción sencilla
```

```
RECIBE:
    BTFSS PIR1,RCIF     ;Bandera de recepción completa
    GOTO RECIBE         ;Esperar la recepción

    MOVF RCREG,W
    MOVWF VAR           ;Control = Recepción

    MOVLW 0x44
    SUBWF VAR,W
    BTFSC STATUS,Z      ;Si Control = 0x44 "d"
    GOTO DER            ;Rotación a la derecha
    MOVLW 0x64
    SUBWF VAR,W
    BTFSC STATUS,Z      ;Si Control = 0x64 "D"
    GOTO DER            ;Rotación a la derecha
    MOVLW 0x49
    SUBWF VAR,W
    BTFSC STATUS,Z      ;Si Control = 0x49 "i"
    GOTO IZQ            ;Rotación a la izquierda
    MOVLW 0x69
    SUBWF VAR,W
    BTFSC STATUS,Z      ;Si Control = 0x69 "I"
    GOTO IZQ            ;Rotación a la izquierda
    GOTO RECIBE         ;Recibir otro dato

DER                     ;Rotación a la derecha
    MOVLW 0X80
    MOVWF DATO          ;Dato = 1000 0000
    MOVLW 0X08
    MOVWF CONT          ;Contador = 8 rotaciones
LOOPD
    MOVF DATO,W
    MOVWF PORTB         ;Salida = Dato
    CALL retardo        ;Se llama al retardo
    BCF STATUS,C        ;Se limpia el acarreo
    RRF DATO            ;Rotación a la derecha
    DECFSZ CONT         ;Contador == 0?
    GOTO LOOPD          ;Si no es continuar el loop
    GOTO RECIBE         ;Si es cero volver a recibir

IZQ                     ;Rotación a la izquierda
    MOVLW 0X01
    MOVWF DATO          ;Dato = 0000 0001
    MOVLW 0X08
    MOVWF CONT          ;Contador = 8 rotaciones
LOOPI
    MOVF DATO,W
    MOVWF PORTB         ;Salida = Dato
    CALL retardo        ;Se llama al retardo
    BCF STATUS,C        ;Se limpia el acarreo
    RLF DATO            ;Rotación a la izquierda
    DECFSZ CONT         ;Contador == 0?
    GOTO LOOPI          ;Si no es cero continuar el loop
    GOTO RECIBE         ;Si es cero volver a recibir

;Subrutina para el retardo
retardo
    MOVLW cte1
    MOVWF valor1
tres
    MOVLW cte2
    MOVWF valor2
dos
    MOVLW cte3
    MOVWF valor3
uno
    DECFSZ valor3
    GOTO uno
    DECFSZ valor2
    GOTO dos

   DECFSZ valor1
   GOTO tres
   RETURN

END
```

Ejercicio 6

```
processor 16f877
include<p16f877.inc>

VAR EQU 0X20 ;Valor de control

;Valores para el retardo
valor1 equ h'21
valor2 equ h'22
valor3 equ h'23
cte1 equ 20h
cte2 equ 50h
cte3 equ 60h

    ORG 0
    GOTO inicio
    ORG 5

inicio
    BSF STATUS,RP0
    BCF STATUS,RP1      ;Se cambia al banco 01
    CLRF TRISB          ;Puerto B como salida
    CLRF TRISC          ;Puerto C como salida
    BSF TXSTA,BRGH      ;Cambia bandera BRGH
    MOVLW D'129'
    MOVWF SPBRG         ;Baud rate = 9600
    BCF TXSTA,SYNC      ;Comunicación síncrona
    BSF TXSTA,TXEN      ;Activa transmisión

    BCF STATUS,RP0      ;Se cambia al banco 00

    BSF RCSTA,SPEN      ;Habilita puerto serie
    BSF RCSTA,CREN      ;Recepción sencilla

RECIBE:
    BTFSS PIR1,RCIF     ;Bandera de recepción completa
```

```
    GOTO RECIBE         ;Esperar recepción

    MOVF RCREG,W
    MOVWF VAR           ;Control = Recepción

    MOVLW 0x53
    SUBWF VAR,W
    BTFSC STATUS,Z      ;Si control = 0x53 "S"
    GOTO STOP           ;Entonces parar
    MOVLW 0x41
    SUBWF VAR,W
    BTFSC STATUS,Z      ;Si control = 0x41 "A"
    GOTO ADELANTE       ;Entonces avanzar
    MOVLW 0x54
    SUBWF VAR,W
    BTFSC STATUS,Z      ;Si control = 0x54 "T"
    GOTO TRAS           ;Entonces retroceder
    MOVLW 0x44
    SUBWF VAR,W
    BTFSC STATUS,Z      ;Si control = 0x44 "D"
    GOTO DER            ;Mover a la derecha
    MOVLW 0x49
    SUBWF VAR,W
    BTFSC STATUS,Z      ;Si control = 0x49 "I"
    GOTO IZQ            ;Mover a la izquierda
    GOTO RECIBE         ;Recibir otro dato

STOP                    ;Parar ambos motores
    CLRF PORTC          ;C = 0
    CLRF PORTB          ;B = 0
    GOTO RECIBE         ;Recibir otro dato

ADELANTE                ;Horario ambos motores
    MOVLW 0X06
    MOVWF PORTC         ;C = 0x06

     MOVWF PORTB         ;B =  0x0A
     GOTO RECIBE         ;Recibir otro dato

TRAS                     ;Antihorario ambos motores
     MOVLW 0X06
     MOVWF PORTC         ;C = 0x06
     MOVLW 0X05
     MOVWF PORTB         ;B = 0x05
     GOTO RECIBE         ;Recibir otro dato

DER                      ;Dirección a la derecha
     MOVLW 0X06
     MOVWF PORTC         ;C = 0x06
     MOVLW 0X09
     MOVWF PORTB         ;C = 0x09
     GOTO RECIBE         ;Recibir otro dato

IZQ                      ;Dirección a la izquierda
     MOVLW 0X06
     MOVWF PORTC
     MOVWF PORTB         ;B = C = 0x06
     GOTO RECIBE         ;Recibir otro dato

;Subrutina para el retardo
retardo
    MOVLW cte1
    MOVWF valor1
tres
    MOVLW cte2
    MOVWF valor2
dos
    MOVLW cte3
    MOVWF valor3
uno
     DECFSZ valor3

    DECFSZ valor2
    GOTO dos
    DECFSZ valor1
    GOTO tres
    RETURN

END
```

Ejercicio 7

```
processor 16f877
include<p16f877.inc>

VOLTAJE EQU 0X22        ;Valor de voltaje

CENT EQU 0X27           ;Valor centenas
DECE EQU 0X28           ;Valor decenas
UNI EQU 0X29            ;Valor unidades

;Valores para el retardo
valor1 equ h'24
valor2 equ h'25
valor3 equ h'26
cte1 equ 20h
cte2 equ 50h
cte3 equ 60h

    ORG 0
    GOTO inicio
    ORG 5

inicio
    BSF STATUS,RP0
    BCF STATUS,RP1          ;Se cambi al banco 01
    CLRF ADCON1             ;ADCON1 = 00 -> PUERTO A Y E ANALOGICOS
    BSF TRISE,0
    BSF TRISE, 0X00
    BSF TXSTA,BRGH          ;Cambia bandera BRGH
    MOVLW D'129'
    MOVWF SPBRG             ;Baud rate = 9600
    BCF TXSTA,SYNC          ;Comunicación sincrona
    BSF TXSTA,TXEN          ;Activa transmisión

    BCF STATUS,RP0          ;Se cambia al banco 0
    MOVLW B'11101001'       ;W = 11101001
```

```
    MOVWF ADCON0            ;Configura ADCON0 con W: usara reloj interno, canal 5, DONE, conv.A/D encendido

    BSF RCSTA,SPEN          ;Habilita puerto serie
    BSF RCSTA,CREN          ;Recepción sencilla

RECIBE
    BTFSS PIR1,RCIF         ;Bandera de recepción completa
    GOTO RECIBE             ;Esperar recepción

    CLRF CENT               ;Centenas = 0
    CLRF DECE               ;Decenas = 0
    CLRF UNI                ;Unidades = 0
    CALL CONVIERTE          ;Convierte valor a digital
    MOVWF VOLTAJE           ;Voltaje = valor digital

    BTFSS VOLTAJE,7     ;Bit 7 = 1 salta
    GOTO V125            ;B7 = 0
    MOVLW 0X02           ;W = 2 (B7 = 1)
    ADDWF CENT           ;CENT + W
    MOVLW 0X05           ;W = 5
    ADDWF DECE           ;DECE + W

V125
    BTFSS VOLTAJE,6     ;Bit 6 = 1 salta
    GOTO V063            ;B6 = 0
    MOVLW 0X01           ;W = 1 (B6 = 1)
    ADDWF CENT           ;CENT + W
    MOVLW 0X02           ;W = 2
    ADDWF DECE           ;DECE + W
    MOVLW 0X05           ;W = 5
    ADDWF UNI            ;UNI + W

V063
    BTFSS VOLTAJE,5      ;Bit 5 = 1 salta
    GOTO V032            ;B5 = 0

    MOVLW 0X06           ;W = 6 (B5 = 1)
    ADDWF DECE           ;DECE + W
    CALL INC_CENT        ;Llamado a INC_CENT
    MOVLW 0X03           ;W = 3
    ADDWF UNI            ;UNI + W
    CALL INC_DECE        ;Llamado a INC_DECE

V032
    BTFSS VOLTAJE,4      ;Bit 4 = 1 salta
    GOTO V016            ;B4 = 0
    MOVLW 0X03           ;W = 3 (B4 = 1)
    ADDWF DECE           ;DECE + W
    CALL INC_CENT        ;Llamado a INC_CENT
    MOVLW 0X02           ;W = 2
    ADDWF UNI            ;UNI + W
    CALL INC_DECE        ;Llamado a INC_DECE

V016
    BTFSS VOLTAJE,3      ;Bit 3 = 1 salta
    GOTO V008            ;B3 = 0
    MOVLW 0X01           ;W = 1 (B3 = 1)
    ADDWF DECE           ;DECE + W
    CALL INC_CENT        ;Llamado a INC_CENT
    MOVLW 0X06           ;W = 6
    ADDWF UNI            ;UNI + W
    CALL INC_DECE        ;Llamado a INC_DECE

V008
    BTFSS VOLTAJE,2      ;Bit 2 = 1 salta
    GOTO V004            ;B2 = 0
    MOVLW 0X08           ;W = 8 (B2 = 1)
    ADDWF UNI            ;UNI + W
    CALL INC_DECE        ;Llamado a INC_DECE

V004
```

```
    BTFSS VOLTAJE,1       ;Bit 1 = 1 salta
    GOTO V002             ;B1 = 0
    MOVLW 0X04            ;W = 4 (B1 = 1)
    ADDWF UNI             ;UNI + W
    CALL INC_DECE         ;Llamado a INC_DECE

V002
    BTFSS VOLTAJE,0       ;Bit 0 = 1 salta
    GOTO ASCII            ;B0 = 0
    MOVLW 0X02            ;W = 2 (B0 = 1)
    ADDWF UNI             ;UNI + W
    CALL INC_DECE         ;Llamado a INC_DECE

;Transmitiendo el valor ASCII
ASCII
    MOVLW 0X30            ;Sumar 30 para valor ASCII a:
    ADDWF CENT            ;centenas
    ADDWF DECE            ;decenas
    ADDWF UNI             ;unidades

    MOVF CENT,W
    CALL TRANSMITE        ;Transmitir centenas
    MOVF DECE,W
    CALL TRANSMITE        ;Transmitir decenas
    MOVLW 0X2E
    CALL TRANSMITE        ;Transmitir un punto
    MOVF UNI,W
    CALL TRANSMITE        ;Transmitir unidades
    MOVLW a' '
    CALL TRANSMITE        ;Transmitir espacio
    MOVLW 0XA7
    CALL TRANSMITE        ;Transmitir grados
    MOVLW a'C'
    CALL TRANSMITE        ;Transmitir C
    MOVLW 0X0D
   CALL TRANSMITE
   MOVLW 0X0A
   CALL TRANSMITE      ;Transmite el fin de la cadena
   GOTO RECIBE         ;Recibe otro dato


TRANSMITE:
    MOVWF TXREG         ;Transmisión = W
TRA
    BSF STATUS,RP0      ;Cambiar al banco 01
ESP
    BTFSS TXSTA,TRMT    ;Bandera de transmisión completa
    GOTO ESP            ;Esperar transmisión
    CALL RETARDO        ;Se llama el retardo
    BCF STATUS,RP0      ;Cambiar al banco 00
    RETURN


CONVIERTE
    BSF ADCON0,2            ;prende bit 2 de ADCON0 -> GO: inicia la comparaciÃ³n
    CALL RETARDO            ;llama subrutina de retardo
ESPERA
    BTFSC ADCON0,2          ;checa bit 2 de ADCON0
    GOTO ESPERA             ;Si es 1, regresa a la etiqueta ESPERA
    MOVF ADRESH,W           ;Si es 0, W = ADRESH
    RETURN

INC_CENT
    MOVLW 0X0A              ;W=0A
    SUBWF DECE,W            ;W = DECE-0A
    BTFSS STATUS, C         ;salta si C = 1
    RETURN                  ;return si C=0
    INCF CENT               ;CENT = CENT + 1 (C=1)
    MOVWF DECE              ;DECE = W
    RETURN
```

```
INC_DECE
    MOVLW 0X0A              ;W=0A
    SUBWF UNI,W             ;W = UNI-0A
    BTFSS STATUS, C         ;salta si C = 1
    RETURN                  ;return si C=0
    INCF DECE               ;DECE = DECE + 1 (C=1)
    MOVWF UNI               ;UNI = W
    CALL INC_CENT           ;llama INC_CENT
    RETURN

;Subrutina para el retardo
RETARDO
    MOVLW cte1
    MOVWF valor1
tres
    MOVLW cte2
    MOVWF valor2
dos
    MOVLW cte3
    MOVWF valor3
uno
    DECFSZ valor3
    GOTO uno
    DECFSZ valor2
    GOTO dos
    DECFSZ valor1
    GOTO tres
    RETURN

END
```

Conclusiones y/o comentarios

Zepeda Baeza Jessica:

Con esta práctica se trabajó la comunicación asíncrona por el puerto serie. Al igual que con otras funcionalidades del microcontrolador, primero se revisaron las configuraciones necesarias que se deben hacer en ciertos registros y ciertas banderas para habilitar la recepción y transmisión por el puerto serie. A través del programa ejemplo del ejercicio 1 se pudo observar de manera simple cómo funciona tanto la transmisión como la recepción. Y con ello se pudieron realizar los demás ejercicios que podían utilizar solamente la recepción o transmisión o incluso ambos. La mayoría de los ejercicios se basaron en hacer la comparación de un dato recibido con un caracter y hacer una cierta acción en base a ello. Esto fue sencillo ya que los motores, las rotaciones y otras asignaciones ya se habían trabajado anteriormente. De igual forma, el último ejercicio fue el más completo y largo ya que involucró el uso del convertidor analógico-digital pero al ya haber sido implementado antes así como la conversión de su voltaje, fue sencillo obtener un dato que se pudiera transmitir.

Barreiro Valdez Alejandro:

En esta práctica se pudo utilizar con diversos ejercicios la comunicación asíncrona por el puerto serie. Los ejercicios se basaron en la recepción y transmisión de caracteres con algún fin. Para realizar esta actividad se aprendió cómo configurar el puerto serie para habilitar la transmisión y la recepción. En el primer código se pudo observar un programa básico sobre la recepción y transmisión de caracteres y cómo se utilizan los valores ASCII para mostrar caracteres en el display de cristal líquido. Se utilizaron códigos que ya se

habían hecho en prácticas anteriores como el corrimiento de bits, la conversión a decimal, el uso de motores y la utilización del display de cristal líquido. Se pudo observar en esta práctica las diferentes maneras que se tienen de recibir y transmitir, utilizando la terminal y hasta un módulo Bluetooth, y con estas actividades se logró entender la utilidad de la comunicación asíncrona.