Universidad Nacional Autónoma de México

Facultad de Ingeniería

Microcomputadoras

Profesor: M.I. Rubén Anaya García

Proyecto 5. Voltímetro Digital con Puerto Serie y Finalización Automática

Suárez Espinoza Mario Alberto

Semestre 2020-2

Entrega: martes 12 de mayo de 2020

Requerimientos:

1. El proyecto consistirá en un microcontrolador que controlará a una pantalla LCD 16x2.
2. El microcontrolador además de desplegar información por la LCD 16x2, la desplegará también en una terminal propia.
3. El sistema tendrá 4 modos de funcionamiento diferentes, estos son:
    a. Despliegue de la conversión A/D en formato Decimal.
    b. Despliegue de la conversión A/D en formato Hexadecimal.
    c. Despliegue de la conversión A/D en formato Binario.
    d. Despliegue de la conversión A/D en Volts (usando tres dígitos).
4. El sistema tendrá un tiempo de funcionamiento de 3 minutos. Después de terminado este tiempo lanzará el mensaje TIEMPO TERMINADO, y dejará de realizar cualquier acción hasta que se presione el botón de reset.

Diseño:

El programa contiene los mismos algoritmos que en fases pasadas. Sólo se agregó interrupciones por Timer0.
Para cubrir el requerimiento acerca del tiempo de funcionamiento se utilizaron contadores anidados. La interrupción por timer0 se lanza cada cierto periodo, por lo que en cada ejecución de la interrupción se incrementa un contador. Como este contador no es suficiente para el requerimiento del tiempo, se anida con otro contador.
El tiempo máximo en iniciarse la interrupción por timer0 se obtiene de la siguiente manera

$$T_{INT} = (T_{PIC})(PRE)(256 - TMRO_{INICIAL})$$

Con
$T_{PIC} = \frac{4}{20\times10^6}$, $PRE = 256$, $TMRO_{INICIAL} = 0$
Resultando en

$$T_{INT\ Max} = \left(\frac{4}{20 \times 10^6}\right)(256)(256) = 13.1072[ms]$$

Agregándose el contador, el tiempo en el que se ejecuta una instrucción puede obtenerse de la siguiente manera

$T_{CONTADOR1} = n_1 * 13.1072[ms]$, con $n \in \mathbb{Z} \cap [0,255]$
Tomando el máximo, $n_1 = 255$ se obtiene

$$T_{CONTADOR1} = 255 * 13.1072[ms] = 3.342336[s]$$

Debido a que el tiempo sigue siendo muy corto, se agrega un contador anidado, entonces la expresión queda

$T_{CONTADOR2} = n_2 * 3.342336[s]$, con $n \in \mathbb{Z} \cap [0,255]$

Tomando el máximo, $n_2 = 255$ se obtiene

$$T_{CONTADOR2} = 255 * 3.342336[s] = 14[min], 12.3[s]$$

Lo cual ya permite el tiempo necesario (3 min)

Para obtener el tiempo necesario se despeja de la formula, quedando

$$n_2 = \frac{T_{CONTADOR2}}{3.342336[s]}$$

Se desea $T_{CONTADOR2} = 3[min] = 60[s]$, entonces

$$n_2 = \frac{60[s]}{3.342336[s]} = 17.951516 \approx 18$$

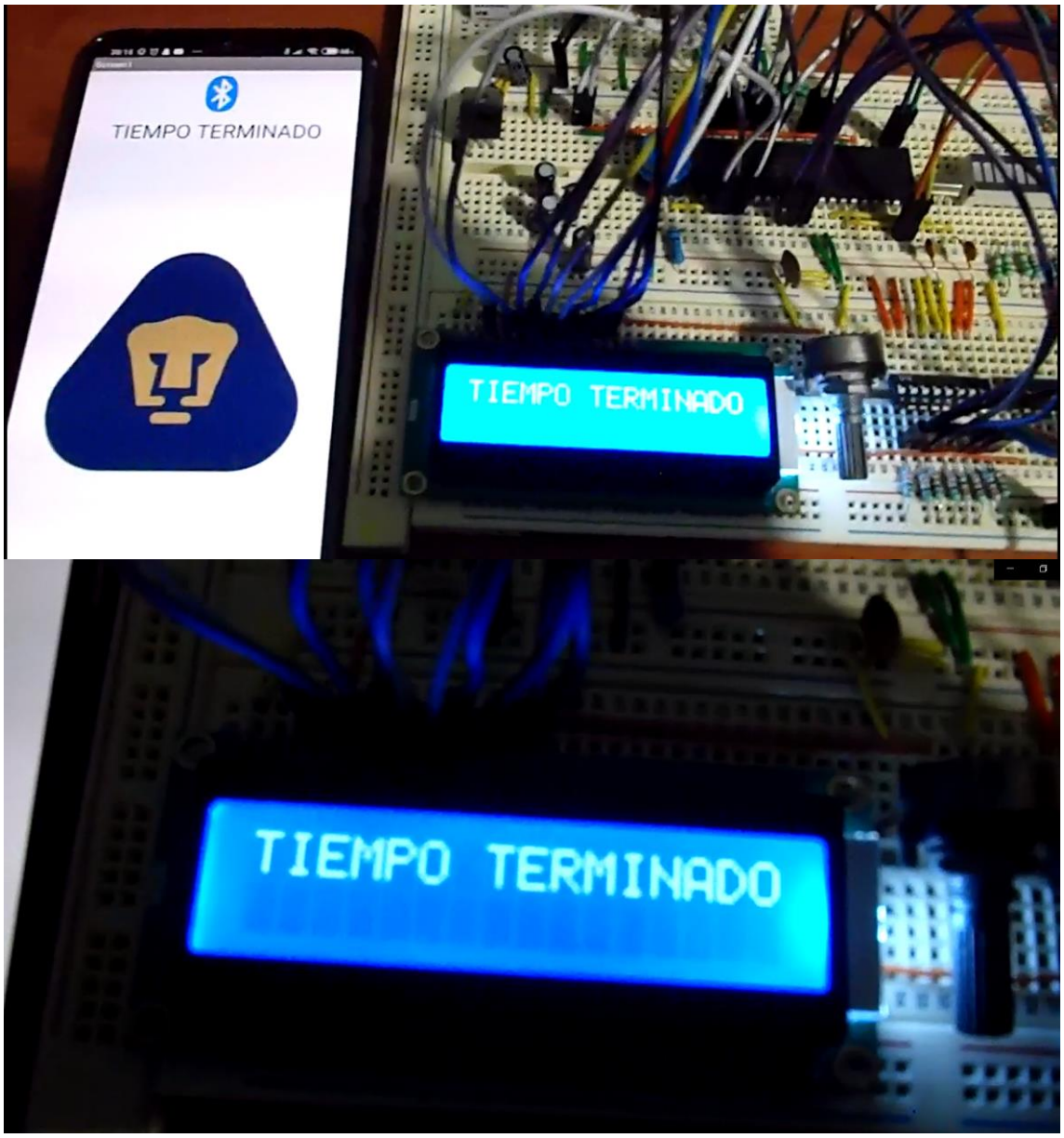
Por lo tanto, los valores máximos de los contadores anidados deben ser 18 y 255.

El pseudocódigo de la rutina de interrupción es:

```
Inicia subrutina de interrupción
Si interrupción fue por desbordamiento de Timer0
      contaInt1 ← contaInt1 + 1
      Si contaInt1 = 18
            contaInt1 ← 0
            contaInt2 ← contaInt2 + 1
            Si contaInt2 = 255
                  Imprime el mensaje TIEMPO TERMINADO
                  Espera de manera indefinida
            En caso contrario
                  Termina y Retorna de la rutina de interrupción
      En caso contrario
            Termina y Retorna de la rutina de interrupción
En caso contrario
      Termina y Retorna de la rutina de interrupción
Fin subrutina de interrupción
```

Resultados obtenidos:

Se agregan imágenes de su funcionamiento



Comentarios:

Las rutinas de interrupción pueden tener diferentes aplicaciones, principalmente cuando se tiene proyectos en los que se realizan varias acciones al mismo tiempo sin llegar a ser paralelismo, pues, al fin y al cabo, la arquitectura del microcontrolador sólo soporta la ejecución de una instrucción al mismo tiempo.

En cuanto al proyecto, basto con realizar los cálculos para la ejecución de la instrucción en el tiempo requerido, y agregarlo al proyecto que se había desarrollado con anterioridad.

Código ASM:

```
;Mario ALberto Suárez Espinoza
;masues64@gmail.com

; PORTB BUS DE DATOS B0-D0 ... B7-D7
; RS - A0
; E - A1
; R/W - GND

; A2 entrada analógica

;Puerto A: Entrada analógica y control display (3 pines de 6)
;Puerto B: Datos display (8 pines de 8)
;Puerto E: Selección de modo (2 pines de 3)

    processor 16f877
    include<p16f877.inc>
valor equ h'20'
valor1 equ h'21'
valor2 equ h'22'
contador equ h'23'
dato equ h'24'
constA:  equ d'255'
constB:  equ d'255'
constC:  equ d'255'
regA:  equ h'25'
regB:  equ h'26'
regC:  equ h'27'
indice: equ h'28'
input: equ h'29'
deco_var: equ h'30'
numerador:  equ h'30'; registro que almacena al numerador
```

```
deno_hex:  equ h'10'; literal denominador hexadecimal
deno_dec:  equ h'0A'; literal denominador decimal
cocien_hex:  equ h'34'; registro que almacena el resultado de la operació
n hexadecimal
res_hex:  equ h'33'; registro que almacena al resto hexadecimal
cocien_dec:  equ h'31'; registro que almacena el resultado de la operació
n hexadecimal
res_dec:  equ h'32'; registro que almacena al resto hexadecimal
regVol2: equ h'33'; registro que almacena a la cifra más significativa de
l voltaje
regVol1: equ h'34'; registro que almacena a la cifra media del voltaje
regVol0: equ h'35'; registro que almacena a la cifra más significativa de
l voltaje
contaVolt: equ h'36'; contador auxiliar para transformar a volts
contaInt1: equ h'37'; contador de interrupción 1
contaInt2: equ h'38'; contador de interrupción 2

    org 0
    goto inicio; Vector de reset
    org 4
    goto interrupciones; Vector de interrupciones
    org 5
inicio:
    clrf PORTA
    CLRF PORTB
    BSF STATUS,RP0 ;Cambia al banco 1
    BCF STATUS,RP1

    ;Configuración de puertos paralelos
    movlw b'000000000'; Puerto B como salida
    movwf TRISB
    movlw h'07'; puerto A inicialmente como e/s digital
    movwf ADCON1
    movlw b'000000100'; sólo A2 es entrada, lo demás es salida
    movwf TRISA
    movlw b'11'; E0 y E1 como entradas
    movwf TRISE

    ;Configuración de comunicación serial
    BSF TXSTA,BRGH ; BRGH <- 1. Selecciona alta velocidad de baudios
    MOVLW D'129' ; Coloca d'129' en el registro SPBRG. Registro para conf
igurar
    ; la velocidad de comunicación. En este caso, se trata de 9600[bauds]
    MOVWF SPBRG
    BCF TXSTA,SYNC ; SYNC <- 0. Configura el modo asíncrono
    BSF TXSTA,TXEN ; TXEN <- 1. Activa la transmisión

    ;Configuración de timer0
    MOVLW B'00000111'
```

```
    MOVWF OPTION_REG; Predivisor del TIMER0 = 256

    bcf STATUS, RP0; Cambia a banco 0

    ;frecuencia de reloj frc (reloj derivado de el oscilador interno A/D
RC)
    ;entrada analógica 2
    ;enciende al convertidor AD
    movlw b'11010001'; configuración de adcon0
    movwf ADCON0

    BSF RCSTA,SPEN ; Habilita el puerto serie (configura RX y TX como pue
rtos serie)
    BSF RCSTA,CREN ; Configura la recepción continua

    ;Configuración de interrupciones
    BCF INTCON,T0IF; Cero en bandera de desbordamiento
    BSF INTCON,T0IE; Habilita desbordamiento por TIMER0
    BSF INTCON,GIE; Habilita interrupciones generales
    CLRF contaInt1; contaInt1 <- 0
    CLRF contaInt2; contaInt2 <- 0

    call inicia_lcd

main:
    call ret100ms; Retardo para sincronizar los datos recibidos en la app
 android
    call ret100ms
    ;Caso0
    movlw h'00'
    subwf PORTE,0
    btfsc STATUS, Z
    goto caso0
    ;Caso1
    movlw h'01'
    subwf PORTE,0
    btfsc STATUS, Z
    goto caso1
    ;Caso2
    movlw h'02'
    subwf PORTE,0
    btfsc STATUS, Z
    goto caso2
    ;Caso3
    movlw h'03'
    subwf PORTE,0
    btfsc STATUS,Z
    goto caso3
```

```
    goto main

caso0:  ; Imprime entrada en decimal
    movlw 0x80
    call comando
    call conversionAD ; Transforma la entrada análoga a digital
    movwf numerador; Salva la entrada en el registro numerador
    ;Divide el contenido del numerador en dos registros, cocien_hex y res
_hex
    call division_dec1
    ;Salva el contenido del cocien_hex en el regitro numerador
    movf cocien_hex,0
    movwf numerador
    ;Divide el contenido de cocien_hex en dos registros, cocien_dec y res
_dec
    call division_dec2
    ;Decodifica a las variables para imprimirlas
    movf cocien_dec,0
    call deco_num; decodifica el número en formato hexadecimal
    call transmite; transmite el dato que está en W por el puerto serie
    call datos; envia el dato que está en W para desplegarlo en la LCD
    movf res_dec,0
    call deco_num; decodifica el número en formato hexadecimal
    call transmite; transmite el dato que está en W por el puerto serie
    call datos; envia el dato que está en W para desplegarlo en la LCD
    movf res_hex,0
    call deco_num; decodifica el número en formato hexadecimal
    call transmite; transmite el dato que está en W por el puerto serie
    call datos; envia el dato que está en W para desplegarlo en la LCD
    movlw a'D'
    call transmite; transmite el dato que está en W por el puerto serie
    call datos

    movlw h'0D'; Carriage Return (retorno de carro)
    call transmite; transmite el dato que está en W por el puerto serie
    movlw h'0A'; Line Feed (salto de línea)
    call transmite; transmite el dato que está en W por el puerto serie

    ;Ciclo para imprimir 12 espacios
    movlw d'12'
    movwf indice
c0_loop:
    movlw a' '
    call datos
c0_endLoop:
    decfsz indice
    goto c0_loop

    movlw 0xC0; Selecciona la dirección 0x40 de la DDRAM
```

```
    call comando

    ;Ciclo para imprimir 16 espacios
    movlw d'16'
    movwf indice
c0_2_loop:
    movlw a' '
    call datos
c0_2_endLoop:
    decfsz indice
    goto c0_2_loop

    goto main

division_dec1:
    clrf cocien_hex; coloca un cero en concien_hex

    movf numerador,0; coloca el contenido de numerador en W
    movwf res_hex; mueve el contenido de W a res_hex
    xorlw 0x00; compara el contenido de W con 0
    btfsc STATUS, Z; revisa si fue cero la operación anterior
    return; termina el algoritmo porque el numerador fue cero

    movlw deno_dec; coloca deno_dec en W
    subwf numerador,0; a 'numerador' le resta el contenido de W y lo alma
cena en W
    btfsc STATUS, C; revisa si 'deno_dec' es mayor que 'numerador'
    goto numerador_dec1_mayor; numerador>=deno_dec
    return; numerador<deno_dec, termina el algoritmo

numerador_dec1_mayor:
    movf numerador,0; mueve el contenido de numerador a W
    movwf res_hex; guarda el contenido de W en res_hex
    goto loop_division_dec1

loop_division_dec1:
    movlw deno_dec; coloca deno_dec en W
    subwf res_hex,1; resta el contenido de W a res_hex, y se almacena el
resultado en res_hex
    incf cocien_hex,1; incrementa en 1 a cocien_hex, el resultado se alma
cena en cocien_hex
    movlw deno_dec; coloca deno_dec en W
    subwf res_hex,0; a 'res_hex' le resta el contenido de W y lo almacena
 en W
    btfsc STATUS, C; revisa si 'y' es mayor que 'x'
    goto loop_division_dec1; c>=y
    return;termina el algoritmo

division_dec2:
```

```
    clrf cocien_dec; coloca un cero en concien_hex

    movf numerador,0; coloca el contenido de numerador en W
    movwf res_dec; mueve el contenido de W a res_dec
    xorlw 0x00; compara el contenido de W con 0
    btfsc STATUS, Z; revisa si fue cero la operación anterior
    return; termina el algoritmo porque el numerador fue cero

    movlw deno_dec; coloca deno_dec en W
    subwf numerador,0; a 'numerador' le resta el contenido de W y lo alma
cena en W
    btfsc STATUS, C; revisa si 'deno_dec' es mayor que 'numerador'
    goto numerador_dec2_mayor; numerador>=deno_dec
    return; numerador<deno_dec, termina el algoritmo

numerador_dec2_mayor:
    movf numerador,0; mueve el contenido de numerador a W
    movwf res_dec; guarda el contenido de W en res_dec
    goto loop_division_dec2

loop_division_dec2:
    movlw deno_dec; coloca deno_dec en W
    subwf res_dec,1; resta el contenido de W a res_dec, y se almacena el
resultado en res_dec
    incf cocien_dec,1; incrementa en 1 a cocien_dec, el resultado se alma
cena en cocien_dec
    movlw deno_dec; coloca deno_dec en W
    subwf res_dec,0; a 'res_dec' le resta el contenido de W y lo almacena
 en W
    btfsc STATUS, C; revisa si 'y' es mayor que 'x'
    goto loop_division_dec2; c>=y
    return;termina el algoritmo

conversionAD:
    bsf STATUS,RP0 ;Cambia al banco 1
    clrf ADCON1; Coloca un 0 en adcon1 para usar el convertidor A/D
    bcf STATUS,RP0 ;Cambia al banco 0

    bsf ADCON0, 2; inicia la conversión Análogo - Digital
    btfsc ADCON0, 2; pregunta si terminó la conversión
    goto $-1; si no ha terminado, se queda en un bucle
    ;Salva la entrada en el registro W
    movf ADRESH, 0; W <- ADRESH

    bsf STATUS,RP0 ;Cambia al banco 1
    bsf ADCON1,0; Coloca un 0x07 en adcon1
    bsf ADCON1,1
    bsf ADCON1,2
    bcf STATUS,RP0 ;Cambia al banco 0
```

```
    return

caso1:  ; Imprime entrada en hexadecimal
    movlw 0x80
    call comando

    call conversionAD ; Transforma la entrada análoga a digital

    ;Salva la entrada en el registro numerador
    movwf numerador
    ;Divide el contenido del numerador en dos registros, cocien_hex y res
_hex
    call division_hexa
    ;Decodifica a las variables para imprimirlas
    movf cocien_hex,0
    call deco_num; decodifica el número en formato hexadecimal
    call transmite; transmite el dato que está en W por el puerto serie
    call datos; envia el dato que está en W para desplegarlo en la LCD
    movf res_hex,0
    call deco_num; decodifica el número en formato hexadecimal
    call transmite; transmite el dato que está en W por el puerto serie
    call datos; envia el dato que está en W para desplegarlo en la LCD
    movlw a'H'
    call transmite; transmite el dato que está en W por el puerto serie
    call datos

    movlw h'0D'; Carriage Return (retorno de carro)
    call transmite; transmite el dato que está en W por el puerto serie
    movlw h'0A'; Line Feed (salto de línea)
    call transmite; transmite el dato que está en W por el puerto serie

    ;Ciclo para imprimir 13 espacios
    movlw d'13'
    movwf indice
c1_loop:
    movlw a' '
    call datos
c1_endLoop:
    decfsz indice
    goto c1_loop

    movlw 0xC0; Selecciona la dirección 0x40 de la DDRAM
    call comando

    ;Ciclo para imprimir 16 espacios
    movlw d'16'
    movwf indice
c1_2_loop:
```

```
    movlw a' '
    call datos
c1_2_endLoop:
    decfsz indice
    goto c1_2_loop

    goto main

division_hexa:
    clrf cocien_hex; coloca un cero en concien_hex

    movf numerador,0; coloca el contenido de numerador en W
    movwf res_hex; mueve el contenido de W a res_hex
    xorlw 0x00; compara el contenido de W con 0
    btfsc STATUS, Z; revisa si fue cero la operación anterior
    return; termina el algoritmo porque el numerador fue cero

    movlw deno_hex; coloca deno_hex en W
    subwf numerador,0; a 'numerador' le resta el contenido de W y lo alma
cena en W
    btfsc STATUS, C; revisa si 'deno_hex' es mayor que 'numerador'
    goto numerador_hexa_mayor; numerador>=deno_hex
    return; numerador<deno_hex, termina el algoritmo

numerador_hexa_mayor:
    movf numerador,0; mueve el contenido de numerador a W
    movwf res_hex; guarda el contenido de W en res_hex
    goto loop_division_hex

loop_division_hex:
    movlw deno_hex; coloca deno_hex en W
    subwf res_hex,1; resta el contenido de W a res_hex, y se almacena el
resultado en res_hex
    incf cocien_hex,1; incrementa en 1 a cocien_hex, el resultado se alma
cena en cocien_hex
    movlw deno_hex; coloca deno_hex en W
    subwf res_hex,0; a 'res_hex' le resta el contenido de W y lo almacena
 en W
    btfsc STATUS, C; revisa si 'y' es mayor que 'x'
    goto loop_division_hex; c>=y
    return;termina el algoritmo

deco_num: ;Subrutina que imprime el número hexadecimal en la pantalla LCD
    movwf deco_var
    movlw h'0A'; coloca 0x0A en W
    subwf deco_var,0; a 'deco_var' le resta el contenido de W y lo almace
na en W
    btfss STATUS, C; revisa si '0x0A' es mayor o igual que 'deco_var'
    goto dec_dec; deco_var<0x0A, por lo tanto decodifica  deco_var + 0x30
```

```
    goto dec_hex; deco_var>=0x0A, por lo tanto decodifica  deco_var + 0x3
7
dec_dec:
    movlw h'30'; W <- 0x30
    addwf deco_var,0; W <- deco_var + W
    return
dec_hex:
    movlw h'37'; W <- 0x37
    addwf deco_var,0; W <- deco_var + W
    return


caso2:  ; Imprime la entrada en binario
    movlw 0x80
    call comando

    call conversionAD ; Transforma la entrada análoga a digital
    movwf input
    ;Ciclo para recorrer todos los bits de los datos de entrada
    movlw d'8'
    movwf indice
c2_loop:
    btfss input,7
    goto c2_0
    goto c2_1
c2_endLoop:
    call transmite; transmite el dato que está en W por el puerto serie
    call datos
    rlf input,1
    decfsz indice
    goto c2_loop

    movlw a'B'
    call transmite; transmite el dato que está en W por el puerto serie
    call datos

    movlw h'0D'; Carriage Return (retorno de carro)
    call transmite; transmite el dato que está en W por el puerto serie
    movlw h'0A'; Line Feed (salto de línea)
    call transmite; transmite el dato que está en W por el puerto serie

    ;Ciclo para imprimir 7 espacios
    movlw d'7'
    movwf indice
c2_loop2:
    movlw a' '
    call datos
c2_endLoop2:
    decfsz indice
```

```
    goto c2_loop2

    movlw 0xC0; Selecciona la dirección 0x40 de la DDRAM
    call comando

    ;Ciclo para imprimir 16 espacios
    movlw d'16'
    movwf indice
c2_2_loop:
    movlw a' '
    call datos
c2_2_endLoop:
    decfsz indice
    goto c2_2_loop

    goto main

c2_0:
    movlw a'0'
    goto c2_endLoop
c2_1:
    movlw a'1'
    goto c2_endLoop


caso3:
    movlw 0x80
    call comando

    call conversionAD ; Transforma la entrada análoga a digital
    movwf contaVolt; contaVolt <- W
    incf contaVolt;
    call toVolts; transforma a volts

    ;Decodifica a las variables para imprimirlas
    movf regVol2,0
    call deco_num; decodifica el número en formato hexadecimal
    call transmite; transmite el dato que está en W por el puerto serie
    call datos; envia el dato que está en W para desplegarlo en la LCD
    movlw a'.'
    call transmite; transmite el dato que está en W por el puerto serie
    call datos
    movf regVol1,0
    call deco_num; decodifica el número en formato hexadecimal
    call transmite; transmite el dato que está en W por el puerto serie
    call datos; envia el dato que está en W para desplegarlo en la LCD
    movf regVol0,0
    call deco_num; decodifica el número en formato hexadecimal
    call transmite; transmite el dato que está en W por el puerto serie
```

```
    call datos; envia el dato que está en W para desplegarlo en la LCD
    movlw a'V'
    call transmite; transmite el dato que está en W por el puerto serie
    call datos

    movlw h'0D'; Carriage Return (retorno de carro)
    call transmite; transmite el dato que está en W por el puerto serie
    movlw h'0A'; Line Feed (salto de línea)
    call transmite; transmite el dato que está en W por el puerto serie

    ;Ciclo para imprimir 7 espacios
    movlw d'11'
    movwf indice
c3_loop:
    movlw a' '
    call datos
c3_endLoop:
    decfsz indice
    goto c3_loop

    movlw 0xC0; Selecciona la dirección 0x40 de la DDRAM
    call comando

    ;Ciclo para imprimir 16 espacios
    movlw d'16'
    movwf indice
c3_2_loop:
    movlw a' '
    call datos
c3_2_endLoop:
    decfsz indice
    goto c3_2_loop

    goto main

toVolts:
    clrf regVol2; regVol2 <- 0
    clrf regVol1; regVol1 <- 0
    clrf regVol0; regVol0 <- 0
finLoopVolt:
    decfsz contaVolt,1
    goto loopVolt
    return; termina el algoritmo
loopVolt:
    movlw h'08'
    subwf regVol0,0
    btfss STATUS, Z; ¿regVol0 = 8?
    goto noRegVol08; regVol0 != 8
    clrf regVol0; regVol0 = 8
```

```
    movlw h'09'
    subwf regVol1,0
    btfss STATUS, Z; ¿regVol1 = 9?
    goto noRegVol19; regVol1 != 9
    clrf regVol1; regVol1 = 9

    movlw h'09'
    subwf regVol2,0
    btfss STATUS, Z; ¿regVol2 = 9?
    goto noRegVol29; regVol2 != 9
    clrf regVol2; regVol2 = 9
    goto finLoopVolt

noRegVol08:
    movlw h'02'
    addwf regVol0,1; regVol0 <- regVol0 + 2
    goto finLoopVolt

noRegVol19:
    incf regVol1,1; regVol1 <- regVol1 + 1
    goto finLoopVolt

noRegVol29:
    incf regVol2,1; regVol2 <- regVol2 + 1
    goto finLoopVolt


inicia_lcd
    movlw 0x30
    call comando
    call ret100ms
    movlw 0x30
    call comando
    call ret100ms
    movlw 0x38
    call comando
    movlw 0x0c
    call comando
    movlw 0x01
    call comando
    movlw 0x06
    call comando
    movlw 0x02
    call comando
    return

comando
    movwf PORTB
```

```
    call ret200
    bcf PORTA,0
    bsf PORTA,1
    call ret200
    bcf PORTA,1
    return

datos
    movwf PORTB
    call ret200
    bsf PORTA,0
    bsf PORTA,1
    call ret200
    bcf PORTA,1
    call ret200
    call ret200
    return

transmite
    MOVWF TXREG ; Copia el contenido W a TXREG para transmitirlo
    BSF STATUS,RP0 ; Cambia al banco 1
    BTFSS TXSTA,TRMT ; Si terminó la transmisión, salta
    GOTO $-1 ; Si no se ha terminado la transmisión, se queda en un bucle
    BCF STATUS,RP0 ; Cambia al banco 0
    return; termina la subrutina

ret200
    movlw 0x02
    movwf valor1
loop
    movlw d'164'
    movwf valor
loop1
    decfsz valor,1
    goto loop1
    decfsz valor1,1
    goto loop
    return

ret100ms
    movlw 0x03
rr  movwf valor
tres movlw 0xff
    movwf valor1
dos movlw 0xff
    movwf valor2
uno decfsz valor2
    goto uno
    decfsz valor1
```

```
    goto dos
    decfsz valor
    goto tres
    return

interrupciones
    btfss INTCON,T0IF; Pregunta si se ha desbordado el Timer0
    goto SAL_NO_FUE_TMR0; No se ha desbordado el timer 0, salta a SAL_NO_
FUE_TMR0

    ;Primer ciclo
    incf contaInt1; contaInt1 <- contaInt1 + 1
    movlw d'18'; W <- 18
    subwf contaInt1,W; W <- contaInt1 - W
    btfss STATUS,Z; ¿W=0?
    goto SAL_INT; Termina la rutina

    ;Ciclo anidado
    clrf contaInt1; reincia contaInt1
    incf contaInt2; contaInt2 <- contaInt2 + 1
    movlw d'255'; W <- 255
    subwf contaInt2,W; W <- contaInt2 - W
    btfss STATUS,Z; ¿W=0?
    goto SAL_INT; Termina la rutina

    clrf contaInt2; reincia contaInt2
    call conclusion; llama a la subrutinac conclusion
    goto $; bucle
SAL_INT:
    bcf INTCON,T0IF; Apaga la bandera de interrupción por Timer0
SAL_NO_FUE_TMR0:
    retfie; Retorna de la interrupción

conclusion
    movlw 0x80
    call comando
    movlw "T"
    call transmite
    call datos
    movlw "I"
    call transmite
    call datos
    movlw "E"
    call transmite
    call datos
    movlw "M"
    call transmite
    call datos
    movlw "P"
```

```
call transmite
call datos
movlw "O"
call transmite
call datos
movlw " "
call transmite
call datos
movlw "T"
call transmite
call datos
movlw "E"
call transmite
call datos
movlw "R"
call transmite
call datos
movlw "M"
call transmite
call datos
movlw "I"
call transmite
call datos
movlw "N"
call transmite
call datos
movlw "A"
call transmite
call datos
movlw "D"
call transmite
call datos
movlw "O"
call transmite
call datos
return
end
```