Universidad Nacional Autónoma de México

Facultad de Ingeniería

Microcomputadoras

Profesor: M.I. Rubén Anaya García

Proyecto 6. Control Automático y Manual de PWM

Suárez Espinoza Mario Alberto

Semestre 2020-2

Entrega: miércoles 27 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 2 modos de funcionamiento diferentes, estos son:
    a. Control manual de una señal PWM. Este control se realiza mediante una entrada analógica.
    b. Control automático de una señal PWM. Esta consistirá en un contador unitario iniciando con 00 hasta llegar a FF, y después decrementar. El tiempo de cambio será de 1 segundo.
4. La selección de modo se realizará mediante un switch.

Diseño:

Rutina principal

El programa configura los registros relacionados a todos los módulos que se usan, tales son puertos paralelos, comunicación serial, timer0, timer2, PWM, convertidor AD e interrupciones.

Posteriormente entra en un ciclo, en el que se pregunta que tipo de modo se utilizará. El siguiente pseudocódigo describe la estructura de este ciclo, el cual fue llamando “main”.

```
Inicia main
Si PORTE = 0
  Ejecuta caso0
Si PORTE = 1
  Ejecuta caso1
Fin main
```

Caso0 y caso1 encapsulan el procedimiento necesario para el control manual y automático respectivamente.

Modo manual

En este modo, el programa desactiva las interrupciones, realiza la conversión A/D.

A continuación, el programa ejecuta un conjunto de instrucciones “común” para ambos casos, por lo que se describirá su funcionamiento más adelante.

El algoritmo para el caso0 (modo manual).

```
Inicia caso0
  INTCON ← 0
  Llama subrutina conversionAD
  Ejecuta casoComun
Fin caso0

Inicio conversionAD
  Cambia al banco 1
  ADCON1 ← 0
  Cambia al banco 0
  Inicia la conversión Análogo-Digital
  Espera hasta que la conversión haya finalizado
  W ← ADRESH
  Cambia al banco 1
  ADCON1 ← 0x07
  Cambia al banco 0
```

```
Fin conversionAD
```

Caso común

Dependiendo del caso anterior que se haya ejecutado, se almacena en el registro W el contenido ya sea del contadorAutomatico o de la conversionAD.
El valor del registro W se escribe en el registro CCPR1L, el cual se usa para controlar el tiempo en alto de la señal PWM en el módulo CCP1 (pin C2), en adición, también se escribe este valor en el puerto D, como una alternativa más para visualizar los datos.

Posteriormente, se especifica el procedimiento para transformar el contenido del registro “numerador” en 3 registros diferentes, con dígitos decimales para posteriormente decodificar cada uno de estos, e imprimirlos tanto en el puerto serie, como en la LCD. La ventaja es que ambos dispositivos de despliegue utilizan codificación ASCII, así que esta se puede reutilizar.
La diferencia está en que en la LCD se muestra en la segunda columna de despliegue el mensaje “Nivel CT”, y una imagen de tipo pila, que se rellena con el nivel del ciclo de trabajo, mientras que en la terminal serial se imprime el mensaje “CT=n%”, dónde n corresponde al porcentaje de ciclo de trabajo.
Para el nivel del ciclo de trabajo se dividió a este en 9 niveles. Descritos mediante el siguiente algoritmo

```
Si CCPR1L = 0
  Imprime nivel 0%
En caso contrario si CCPR1L < 32
  Imprime nivel 12.5%
En caso contrario si CCPR1L < 64
  Imprime nivel 25%
En caso contrario si CCPR1L < 96
  Imprime nivel 37.5%
En caso contrario si CCPR1L < 128
  Imprime nivel 50%
En caso contrario si CCPR1L < 160
  Imprime nivel 62.5%
En caso contrario si CCPR1L < 192
  Imprime nivel 75%
En caso contrario si CCPR1L < 224
  Imprime nivel 87.5%
En caso contrario
  Imprime nivel 100%
```

Dependiendo de cada nivel, en la LCD se dibuja la pila. Para el nivel 0% se dibuja vacía, mientras que para el 100% se dibuja llena (representado con 8 caracteres con todos los pixeles llenos).

Una vez que se imprimió el mensaje en el LCD y en la terminal, se envían por el puerto serie los caracteres de “retorno de carro” y “salto de línea” para que en una terminal compatible se visualice una nueva línea por cada mensaje enviado.

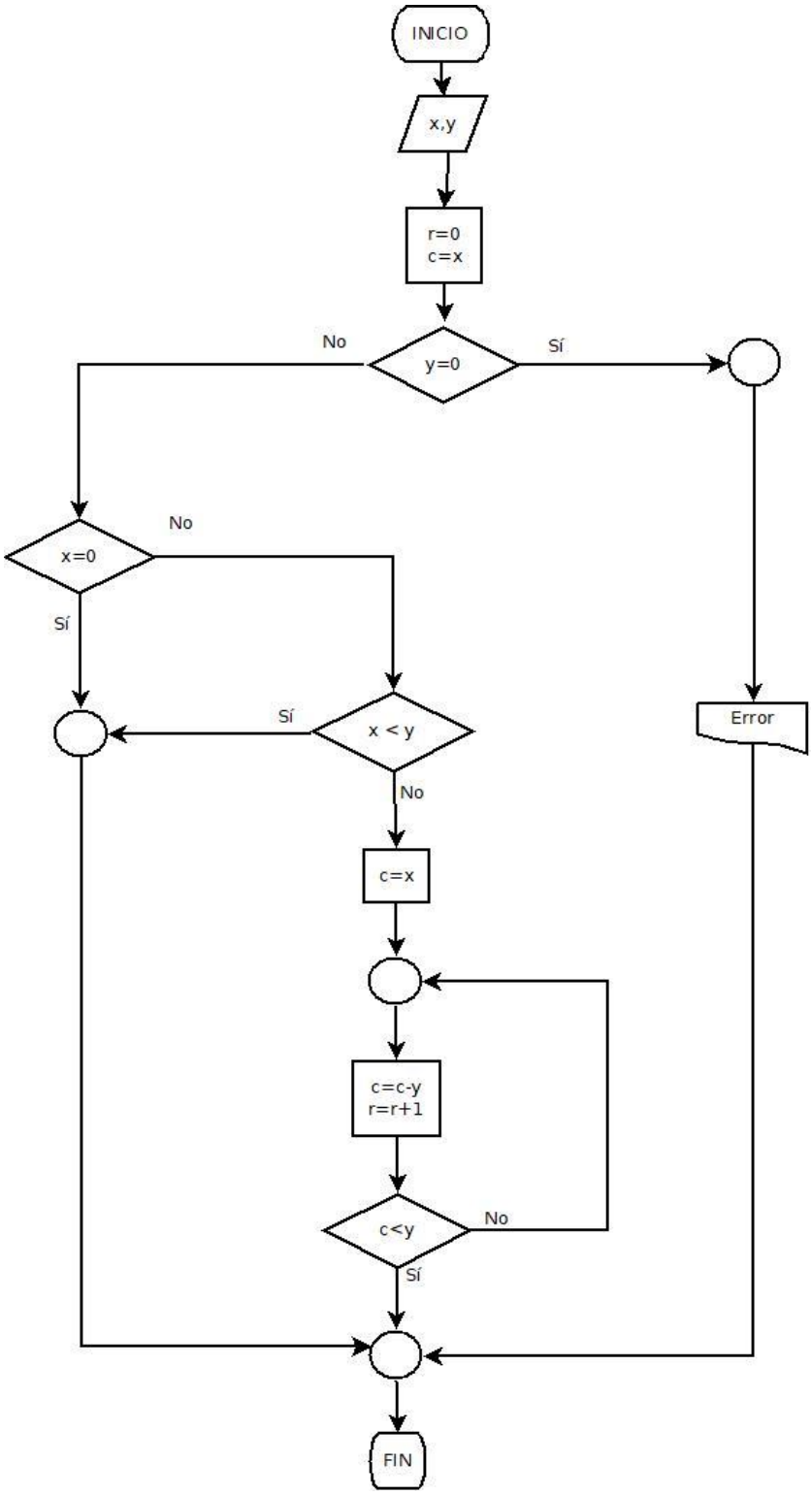
El pseudocódigo de el caso común es el siguiente

```
Inicio casoComun
  numerador ← W
  CCPR1L ← W
  PORTD ← W
  Llama a la subrutina division_dec1
  numerador ← cocien_hex
  Llama a la subrutina division_dec2
  Imprime en LCD y transmite la cadena “PWM=”
  Decodifica e imprime el número contenido en cocien_dec
  Decodifica e imprime el número contenido en res_dec
  Decodifica e imprime el número contenido en res_hexa
  Llama a la subrutina impr_nivel // Pseudocódigo ya documentado en
   líneas previas
  Transmite 0x0D //Retorno de carro
  Transmite 0x0A //Salto de línea
Fin casoComun

Inicio division_dec1
      cocien_hex ← numerador / 0x0A
res_hex ← numerador mod 0x0A
Fin division_dec1

Inicio division_dec2
      cocien_dec ← numerador / 0x0A
res_dec ← numerador mod 0x0A
Fin division_dec2
```

Como el PIC16F877A no posee instrucción para dividir, se implementó el siguiente algoritmo.



Modo automático

En este modo, el programa debe activar las interrupciones para que el incremento sea automático. Para ello configura al timer0, y habilita las interrupciones generales y por timer0.

En la rutina de interrupción se modifica un registro llamado "contadorAutomatico", así que el valor de este se salva en el registro numerdor, y a continuación realiza las operaciones del caso común.
El pseudocódigo de este caso es el siguiente:

```
Inicio caso1
  Configura al timer 0 en modo de trabajo temporizador y predivisor
   256
  Habilita las interrupciones generales y por timer0
  W ← contadorAutomatico
  Ejecuta casoComun
Fin caso1
```

Rutina de interrupción

Como se menciona previamente, la rutina de interrupción es la encargada de generar la cuenta. Es requerimiento que la cuenta cambie cada segundo. Como el timer0 se configuró para que se ejecute cada $13.1072[ms]$, es necesario implementar un contador interno dentro de la interrupción, que cuente las veces que se entra a esta, para poder generar un retraso de 1segundo. Para obtener el valor de este contador se ocupa la fórmula:
$T_{CONTADOR} = n * 13.1072[ms]$, con $n \in \mathbb{Z} \cap [0,255]$
Se requiere $T_{CONTADOR} = 1s$ entonces:

$$n = \frac{1[s]}{13.1072[ms]} = 76.293945 \approx 76$$

Por lo tanto, se debe poner el valor de 76 en el contador de la interrupción.
Posteriormente, la rutina de interrupción llama a otra subrutina, la cual realizará el incremento o decremento de un contador, dependiendo de si ha llegado al límite inferior (0x00) o al superior (0xFF).
Los pseudocódigos de estas rutinas son:

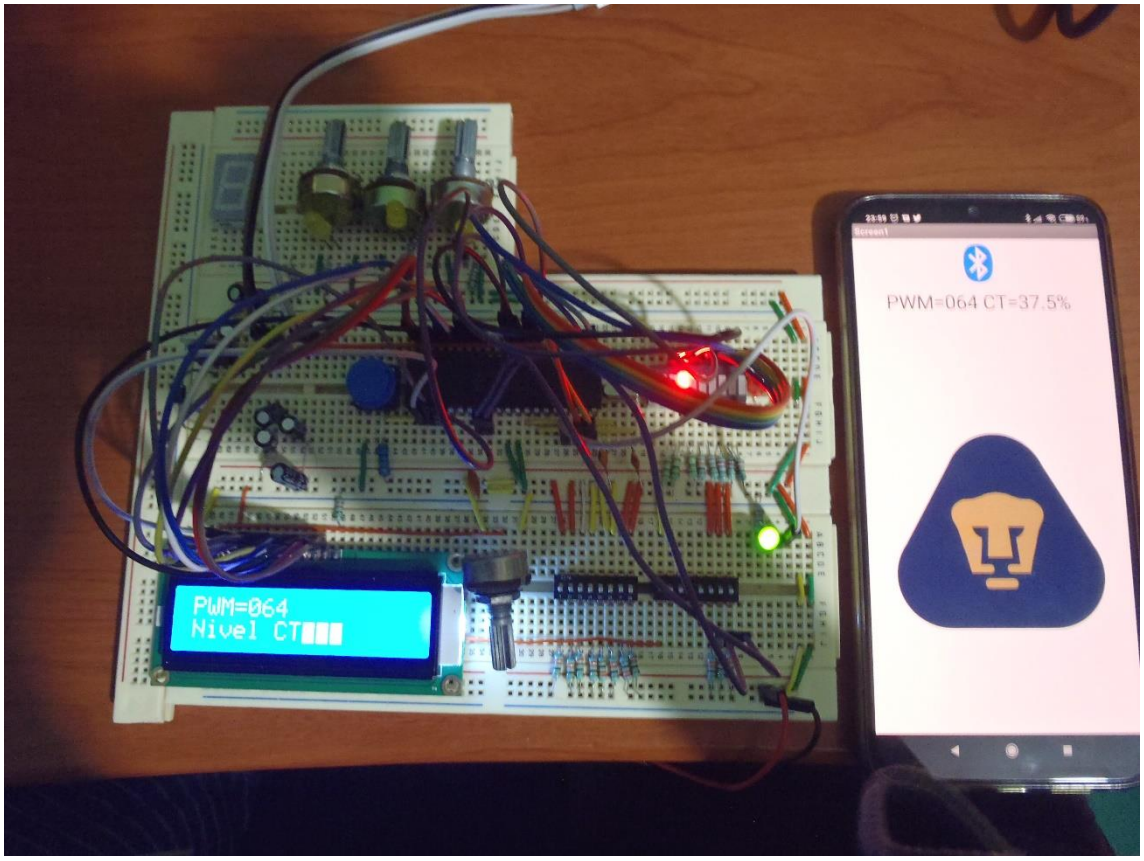
```
Inicio interrupciones
  Si la interrupción fue por Timer0
    contaInt1 ← contaInt1 + 1
    Si contaInt1 = 76
      contaInt1 ← 0
      Llama a subrutina conclusion
Fin interrupciones


Inicio conclusion
  Si banderaIncremental = 1
    contadorAutomatico ← contadorAutomatico + 1
    Si contadorAutomatico = 255
```

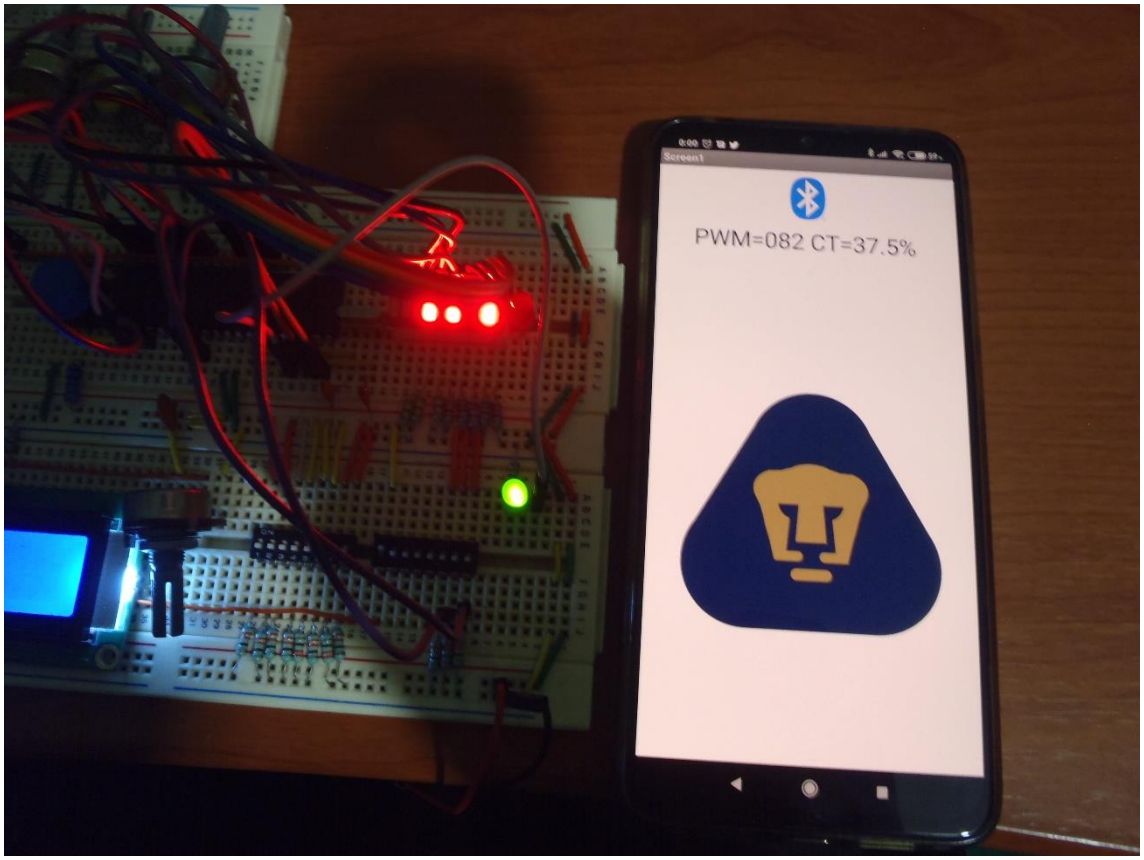
```
      banderaIncremental ← 0
  En caso contrario
    contadorAutomatico ← contadorAutomatico - 1
    Si contadorAutomatico = 0
      banderaIncremental ← 1
Fin conclusion
```

Resultados obtenidos:

Se agregan imágenes de su funcionamiento




PWM=076
Nivel CT


Screen1
PWM=082 CT=37.5%

Comentarios:

El proyecto siguió siendo muy parecido a los anteriores, añadiendo la funcionalidad de PWM.

En lo personal el reto estuvo en dividir como se muestra el ciclo de trabajo tanto en la terminal como en la pantalla LCD, debido a que para mostrar el porcentaje se necesitaría otra vez divisiones de al menos 16 bits, operación que sería muy difícil de implementar dado que todo el proyecto se maneja con 8 bits. Para ello decidí dividir la resolución para mostrar el PWM en 10 niveles, lo cual redujo el problema mostrar el mensaje dependiendo del caso en el que se esté.

Código ASM:

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

; PORTB BUS DE DATOS B0-D0 ... B7-D7
; RS - A0
; E - A1
; R/W - GND
; PWM - C2 (CCP1)

; 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)
;Puerto C: PWM (1 pin de 8)
;Puerto D: Debug de CCPR1L para conocer el estado del PWM

    processor 16f877
    include<p16f877.inc>
;Registros para rutinas de retardo
valor equ h'20'
valor1 equ h'21'
valor2 equ h'22'
indice: equ h'23'; Registro índice para ciclos
deco_var: equ h'24'; Registro auxiliar para rutina de decodificación hexa
decimal
numerador:  equ h'25'; 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'26'; registro que almacena el resultado de la operació
n hexadecimal
res_hex:  equ h'27'; registro que almacena al resto hexadecimal
cocien_dec:  equ h'28'; registro que almacena el resultado de la operació
n hexadecimal
res_dec:  equ h'29'; registro que almacena al resto hexadecimal
contaInt1: equ h'30'; contador de interrupción 1
; bandera para saber si incrementa o decrementa la cuenta en modo automát
ico
banderaIncremental: equ h'31'
contadorAutomatico: equ h'32'; registro contador para el modo automático

    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'1'; E0 como entrada
    movwf TRISE
    clrf TRISD

    ;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

    ;Configuración PWM en banco 1
    BCF TRISC,2; coloca el bit 2 del puerto C como salida (pin CCP1)
```

```
    MOVLW D'255'
    MOVWF PR2; PR2 <- d'255'

    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
    CLRF TMR0; TMR0 <- 0
    CLRF contaInt1; contaInt1 <- 0

    ;Configuración de PWM en banco 0

    ;Configura a CCP1 como PWM
    MOVLW B'00001100'
    MOVWF CCP1CON; CCP1CON <- b'00001100'

    ;Configura a al timer2
    ;Configrua un postescala de 1:1
    ;Enciende al timer2
    ;Configura una preescala de 16
    MOVLW B'00000111'
    MOVWF T2CON; T2CON <- b'00000111'
    ;T PWM = (T pic) (TIMER2_Prescaler)(PR2 + 1)
    ;Significa que T PWM = (4/20x10^6)(16)(256)=819.2[us]

    CLRF PORTD; PORTD <- 0
    bsf banderaIncremental,0; Inicialmente incrementa la cuenta
    CLRF contadorAutomatico; contadorAtomatico <- 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

    goto main

caso0:  ; Modo manual
    clrf INTCON; Desactivación de interrupciones
    call conversionAD ; Transforma la entrada análoga a digital
    ; W <- conversionAD
casoComun:
    movwf PORTD; PORTD <- W
    movwf numerador; numerador <- W

    ;Modifica el CT del PWM de acuerdo a el contador automático o a la co
nversión AD
    movwf CCPR1L; CCPR1L <- W

    ;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

    ;Impresión de "PWM="
    movlw 0x80
    call comando
    movlw a'P'
    call transmite
    call datos
    movlw a'W'
    call transmite
    call datos
    movlw a'M'
    call transmite
    call datos
    movlw a'='
    call transmite
    call datos
```

```
    ;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

    ;Impresión del nivel
    call impr_nivel

    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

    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

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

impr_nivel:; Rutina para imprimir el nivel del CT PWM
    movlw 0xC0; Selecciona la dirección 0x40 de la DDRAM
    call comando

    movlw a'N'
    call datos
    movlw a'i'
    call datos
    movlw a'v'
    call datos
    movlw a'e'
    call datos
    movlw a'l'
    call datos
    movlw a' '
    call transmite
    call datos
    movlw a'C'
```

```
    call transmite
    call datos
    movlw a'T'
    call transmite
    call datos
    movlw a'='
    call transmite

    movf CCPR1L,0
    movlw d'0'
    subwf CCPR1L,0
    btfsc STATUS,Z; Salta si CCPR1L!=0
    goto impr_nivel_0; Imprime 0%
    movf CCPR1L,0
    movlw d'32'
    subwf CCPR1L,0
    btfss STATUS,C; Salta si CCPR1L>=32
    goto impr_nivel_1; Imprime 12.5%
    movlw d'64'
    subwf CCPR1L,0
    btfss STATUS,C; Salta si CCPR1L>=32
    goto impr_nivel_2; Imprime 25%
    movlw d'96'
    subwf CCPR1L,0
    btfss STATUS,C; Salta si CCPR1L>=32
    goto impr_nivel_3; Imprime 37.5%
    movlw d'128'
    subwf CCPR1L,0
    btfss STATUS,C; Salta si CCPR1L>=32
    goto impr_nivel_4; Imprime 50%
    movlw d'160'
    subwf CCPR1L,0
    btfss STATUS,C; Salta si CCPR1L>=32
    goto impr_nivel_5; Imprime 62.5%
    movlw d'192'
    subwf CCPR1L,0
    btfss STATUS,C; Salta si CCPR1L>=32
    goto impr_nivel_6; Imprime 75%
    movlw d'224'
    subwf CCPR1L,0
    btfss STATUS,C; Salta si CCPR1L>=32
    goto impr_nivel_7; Imprime 87.5%
    goto impr_nivel_8; Imprime 100%

impr_nivel_0:
    ;Ciclo para imprimir 7 espacios
    movlw d'8'
    movwf indice
    movlw a' '
```

```
    call datos
    decfsz indice
    goto $-3

    movlw a'0'
    call transmite
    movlw a'%'
    call transmite
    return

impr_nivel_1:
    movlw h'ff'
    call datos
    ;Ciclo para imprimir 7 espacios
    movlw d'7'
    movwf indice
    movlw a' '
    call datos
    decfsz indice
    goto $-3

    movlw a'1'
    call transmite
    movlw a'2'
    call transmite
    movlw a'.'
    call transmite
    movlw a'5'
    call transmite
    movlw a'%'
    call transmite
    return

impr_nivel_2:
    ;Ciclo para imprimir 2 chars
    movlw d'2'
    movwf indice
    movlw h'ff'
    call datos
    decfsz indice
    goto $-3

    ;Ciclo para imprimir 6 espacios
    movlw d'6'
    movwf indice
    movlw a' '
    call datos
    decfsz indice
    goto $-3
```

```
    movlw a'2'
    call transmite
    movlw a'5'
    call transmite
    movlw a'%'
    call transmite
    return

impr_nivel_3:
    ;Ciclo para imprimir 3 chars
    movlw d'3'
    movwf indice
    movlw h'ff'
    call datos
    decfsz indice
    goto $-3

    ;Ciclo para imprimir 5 espacios
    movlw d'5'
    movwf indice
    movlw a' '
    call datos
    decfsz indice
    goto $-3

    movlw a'3'
    call transmite
    movlw a'7'
    call transmite
    movlw a'.'
    call transmite
    movlw a'5'
    call transmite
    movlw a'%'
    call transmite
    return

impr_nivel_4:
    ;Ciclo para imprimir 4 chars
    movlw d'4'
    movwf indice
    movlw h'ff'
    call datos
    decfsz indice
    goto $-3

    ;Ciclo para imprimir 4 espacios
    movlw d'4'
```

```
    movwf indice
    movlw a' '
    call datos
    decfsz indice
    goto $-3

    movlw a'5'
    call transmite
    movlw a'0'
    call transmite
    movlw a'%'
    call transmite
    return

impr_nivel_5:
    ;Ciclo para imprimir 5 chars
    movlw d'5'
    movwf indice
    movlw h'ff'
    call datos
    decfsz indice
    goto $-3

    ;Ciclo para imprimir 3 espacios
    movlw d'3'
    movwf indice
    movlw a' '
    call datos
    decfsz indice
    goto $-3

    movlw a'6'
    call transmite
    movlw a'2'
    call transmite
    movlw a'.'
    call transmite
    movlw a'5'
    call transmite
    movlw a'%'
    call transmite
    return

impr_nivel_6:
    ;Ciclo para imprimir 6 chars
    movlw d'6'
    movwf indice
    movlw h'ff'
    call datos
```

```
    decfsz indice
    goto $-3

    ;Ciclo para imprimir 2 espacios
    movlw d'2'
    movwf indice
    movlw a' '
    call datos
    decfsz indice
    goto $-3

    movlw a'7'
    call transmite
    movlw a'5'
    call transmite
    movlw a'%'
    call transmite
    return

impr_nivel_7:
    ;Ciclo para imprimir 7 chars
    movlw d'7'
    movwf indice
    movlw h'ff'
    call datos
    decfsz indice
    goto $-3

    movlw a' '
    call datos

    movlw a'8'
    call transmite
    movlw a'7'
    call transmite
    movlw a'.'
    call transmite
    movlw a'5'
    call transmite
    movlw a'%'
    call transmite
    return

impr_nivel_8:
    ;Ciclo para imprimir 8 chars
    movlw d'8'
    movwf indice
    movlw h'ff'
    call datos
```

```
    decfsz indice
    goto $-3

    movlw a'1'
    call transmite
    movlw a'0'
    call transmite
    movlw a'0'
    call transmite
    movlw a'%'
    call transmite
    return

caso1:
    ; Configuración de Timer 0
    ; Modo de trabajo temporizador
    ; Predivisor = 256
    movlw h'03'
    movwf OPTION_REG
    ; Configuración de interrupciones
    ; Habilita interrupciones generales
    ; Habilita interrupción por desbordamiento de Timer0
    movlw h'A0'
    movwf INTCON

    movf contadorAutomatico,0; W <- contadorAutomatico

    goto casoComun

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

    ;Ciclo para contador
    incf contaInt1; contaInt1 <- contaInt1 + 1
    movlw d'76'; W <- 76. Para lograr contador de 1 segundo
    subwf contaInt1,W; W <- contaInt1 - W
    btfss STATUS,Z; ¿W=0?
    goto SAL_INT; Termina la rutina

    clrf contaInt1; reincia contaInt1
    call conclusion
SAL_INT:
    bcf INTCON,T0IF; Apaga la bandera de interrupción por Timer0
SAL_NO_FUE_TMR0:
    retfie; Retorna de la subrutina

conclusion:
    btfss banderaIncremental,0; Salta si debe incrementar el contador
    goto modoDecremental

    incf contadorAutomatico
    movlw d'255'
    subwf contadorAutomatico,0; W <- contadorAutomatico - W
    btfsc STATUS,Z; Salta si contadorAutomatico != 255
    bcf banderaIncremental,0; banderaIncremental <- 0
    return
modoDecremental:
    decf contadorAutomatico
    movlw d'0'
    subwf contadorAutomatico,0; W <- contadorAutomatico - W
    btfsc STATUS,Z; Salta si contadorAutomatico != 0
    bsf banderaIncremental,0; banderaIncremental <- 0
    return
    end
```