

Programación en PIC

Los primeros pasos

- *Encender y apagar un LED*
- *Secuenciador con cuatro LED's*
- *Leer los pulsadores y encender el LED correspondiente*
- *Programa avanzado utilizando el entrenador K-061*

Como la práctica hace al maestro, veremos que con la realización de ejemplos prácticos se pueden despejar muchas dudas. Los ejercicios se basan en el entrenador PIC K-061 de CEKIT, cuyo diagrama esquemático se encuentra al final del capítulo II “Conozca el PIC16C84”. Aunque lo más conveniente es tener el entrenador, usted puede ensamblar los circuitos equivalentes en un protoboard.

Ejercicio 1: Encender y apagar un LED

Este es el ejercicio básico por excelencia y ayuda a todas las personas a perder el miedo al manejo del microcontrolador. Si se observa el diagrama del entrenador, los ocho pines del puerto B han sido dispuestos como salidas con el objeto de controlar LED's, necesitando sólo una resistencia en serie con éste para limitar la corriente. El integrado ULN2803 se encarga de aislar el circuito y de entregar una buena cantidad de corriente, suficiente para proporcionar iluminación adecuada al LED.

Para encender el primer LED, tiene que colocarse un estado lógico alto en el pin 6 del microcontrolador, mientras que un estado bajo hará que éste se apague; esto se debe al transistor interno del ULN2803 que invierte la señal. La función del microcontrolador será entonces sencilla: alternar estados lógicos altos y bajos en su pin RB0 con un retardo tal, que nos permita visualizar el efecto. Para una mejor comprensión del proceso se muestra en la figura 1 el diagrama de flujo respectivo, el mismo que nos sirve para la realización del programa.

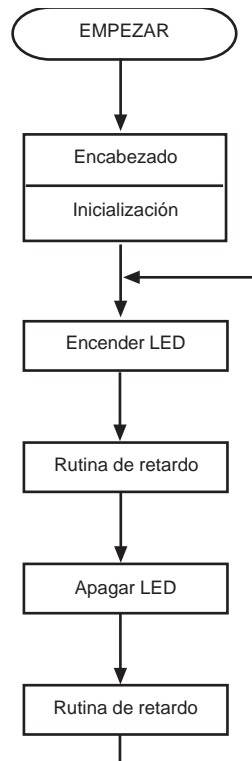


Figura 1. Diagrama de flujo del primer ejercicio.

;=====			
Este programa prende y apaga un LED			
list	p=16c84	;se utiliza el PIC16C84	
;-----			
ptob	equ 06	;el puerto b está en la dirección 06 de la memoria RAM	Encabezado
reg1	equ 0b	;aquí se asignan nombres para reemplazar los registros	
reg2	equ 0c		
reg3	equ 0d		
valor1	equ 30	;aquí se asignan nombres a los valores	
valor2	equ 40	;constantes	
valor3	equ 50		
;-----			
reset	org 0	;el vector de reset es la dirección 00	Iniciación
	goto inicio	;se envía al inicio del programa	
	org 7	;empieza el programa en la posición de memoria 7	
inicio	movlw 00	;se carga el registro W con el valor 00	Encender LED
	tris ptob	;se programan los pines del puerto B como salidas	
otra	movlw 01	;se carga el registro W con el valor 01	
	movwf ptob	;se pasa ese valor al puerto B para encender el LED que se encuentra conectado al pin RB0	Llamar retardo
	call retardo	;se mantiene el LED encendido por un momento	Apagar LED
	movlw 00	;se carga el registro W con el valor 00	Llamar retardo
	movwf ptob	;se carga el registro puerto B con cero para apagar el LED	
	call retardo	;se mantiene el LED apagado por un momento	
	goto otra	;se continúa el programa en otra, es decir que se queda ejecutando la misma acción	Volver a encender el LED
		;esta rutina genera un tiempo de espera	
		;se carga el registro W con el número valor1, es decir con 30	
retardo	movlw valor1	;se carga el registro W con el número valor1, es decir con 30	Rutina de retardo
	movwf reg1	;se traslada ese número al registro reg1	
tres	movlw valor2	;se carga el registro W con el número valor2, es decir con 40	
	movwf reg2	;se traslada ese número al registro reg2	
dos	movlw valor3	;se carga el registro W con el número valor3, es decir con 50	
	movwf reg3	;se traslada ese número al registro reg3	
uno	decfsz reg3	;se decrementa el registro reg3 y si el valor es cero se salta a la	
	goto uno	;instrucción siguiente, si no es cero se continúa decrementando	
	decfsz reg2	;igual que el anterior	
	goto dos		
	decfsz reg1	;igual que el anterior	Final del programa
	goto tres		
	retlw 0		
	end		
;=====			

Figura 2. Programa que enciende y apaga un LED (ejer1.asm)

Consideremos este primer programa; en él se pueden observar una serie de líneas de cabecera, antes de la etiqueta *inicio*. Ellas, como ya lo habíamos mencionado, son una herramienta que nos proporciona el ensamblador para asignar nombres

lógicos, fácilmente memorizables, a algunas posiciones de memoria, bits, puertos y registros para utilizar éstos en el cuerpo principal del programa; el ensamblador se encarga de remplazar estos nombres por los valores numéricos correspondientes, formando así el programa en código objeto.

Con lo anterior, la dirección 06 corresponde al puerto B, al cual hemos llamado *ptob*, de más fácil recordación que un número. De igual manera tres posiciones auxiliares de memoria: la 0B, 0C y 0D, que contendrán valores a decrementar para establecer una rutina de retardo, las hemos llamado *reg1*, *reg2* y *reg3*. También se han utilizado tres cantidades a las cuales les hemos asignado nombres, con el objeto de facilitar el manejo ya que, al momento de ensamblar el programa, podemos remplazarlos rápidamente si ellos no eran los adecuados. Veamos ahora algunos bloques de instrucciones más detalladamente:

Inicio. La primera instrucción carga el registro W con 0 para configurar, con la segunda instrucción (*tris*), el puerto B como salidas. Se debe recordar que un *cero*, en el registro de trabajo W, configura el pin respectivo como salida, mientras que un *uno* lo hace como entrada. Esta instrucción se da, por lo regular una sola vez por puerto.

Otra. Aquí se carga el registro de trabajo W con uno, con el objeto de sacar estados lógicos altos por el pin RB0 del puerto B, siendo este estado el encargado de encender el LED. Después de configurado el puerto, éste puede ser tratado como cualquier otro registro, admitiendo transferencia de datos hacia (escribir sobre el puerto) y desde (leer el puerto) el registro de trabajo W. También se pueden realizar operaciones lógicas, de rotación, etc. La tercera línea hace un llamado a una rutina de retardo, la cual se explicará más adelante, retornando al programa principal tan pronto ésta es ejecutada.

La parte que sigue carga el registro de trabajo W con el literal cero, el cual se sacará por el puerto B, con el objeto de apagar el LED. De nuevo se hace un llamado a la rutina de retardo, para visualizar el efecto; después de la ejecución de la rutina, el control del programa salta a la dirección dada por la etiqueta *otra* para continuar realizando la primera parte del mismo.

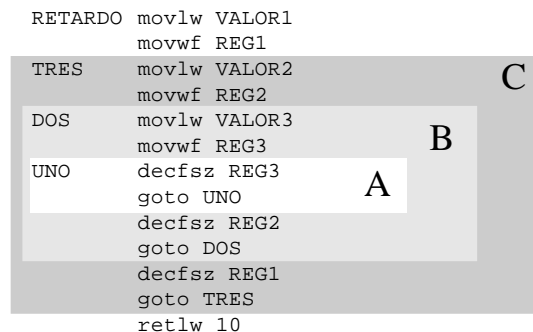


Figura 3. Rutina de retardo

Retardo. En la figura 3 se muestra la rutina de retardo mencionada anteriormente. Debido a la gran velocidad con la cual se ejecutan las instrucciones en el microcontrolador (un millón por segundo, con un reloj de 4 MHz), es necesario efectuar, con mucha frecuencia, retardos que nos permitan observar algunos fenómenos lentos, tales como el encendido de LEDs, lámparas, relés, etc. Tales retardos se logran cargando, con un determinado valor, posiciones auxiliares de memoria RAM o registros, decrementándolos posteriormente y consultando el momento en el cual llegan a cero; dependiendo del resultado de la consulta, el programa puede dirigirse a dos sitios diferentes, como puede observarse en el diagrama de flujo de la figura 4. En ella se han resaltado tres pequeños bloques, que corresponden a una sola instrucción en la familia del microcontrolador PIC: *decfsz*. Esta instrucción decrementa un registro y consulta si el contenido de éste ha llegado a cero; si es así, omite la siguiente instrucción; si no lo es, la ejecuta. Esto permite implementar bifurcaciones de acuerdo al cumplimiento de una condición.

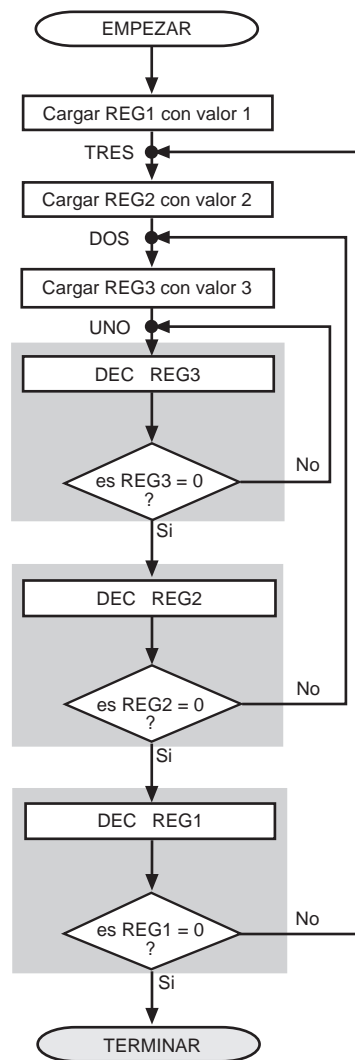


Figura 4. Diagrama de flujo de la rutina de retardo

Para conocer el tiempo que se tarda toda la rutina, podemos hacer algunos cálculos, siempre expresando las cantidades en términos de los ciclos de instrucción y para ello debemos saber cuanto tarda cada una de ellas. Afortunadamente, todas las instrucciones de los microcontroladores PIC se ejecutan en un ciclo, excepto cuando se realiza un salto, en cuyo caso la instrucción utiliza dos ciclos.

Con lo anterior, debemos empezar por la rutina más interna, marcada con la letra **A** en la figura 3. Mientras que no se cumpla la condición, la instrucción *decfsz reg3* consume 1 ciclo y la instrucción *goto* se ejecuta en dos ciclos; esta rutina interna gasta entonces tres ciclos, ejecutándose un número de veces que está dado por el valor contenido en el registro *valor3*, en este caso el valor hexadecimal 50 (observe la definición de éste en la cabecera y recuerde que por defecto se asumen cantidades hexadecimales). El número total de ciclos consumidos por esta rutina será entonces el producto:

$$\begin{aligned}\text{Número ciclos A} &= 3 \times \text{valor3} \\ &= 3 \times 80 \text{ (el decimal de 50Hex)} \\ &= 240 \text{ ciclos de instrucción}\end{aligned}$$

Haciendo un análisis similar, la subrutina **B** (que abarca la subrutina **A**) tendrá 245 (1 + 1 + 240 + 1 + 2) ciclos. Aquí el total de ciclos consumidos depende del registro *valor2* y ella se tardará:

$$\begin{aligned}\text{Número ciclos B} &= 245 \times \text{valor2} \\ &= 245 \times 64 \text{ (el decimal de 40Hex)} \\ &= 15860 \text{ ciclos de instrucción}\end{aligned}$$

De igual manera, podemos considerar que para la subrutina **C** (que abarca tanto la **A** como la **B**) se tienen un número de ciclos de 15865 (15860 + 5); considerando que el número total de ciclos depende del registro *valor1*, se tiene:

$$\begin{aligned}\text{Número ciclos C} &= 15865 \times \text{valor1} \\ &= 15865 \times 48 \text{ (el decimal de 30Hex)} \\ &= 761520 \text{ ciclos de instrucción}\end{aligned}$$

Podemos emplear una expresión algebraica para lo anterior, resultando más fácil realizar los cálculos:

$$\text{Total Ciclos} = ((\text{valor3} \times 3 + 5) \times \text{valor2} + 5) \times \text{valor1}$$

En este caso, la constante 5 corresponde al número de ciclos que están utilizando las instrucciones adicionales en las rutinas **B** y **C**, mientras que *valor1*, *valor2* y *valor3* corresponden a las constantes tomadas de la cabecera del programa. Si quiere conocer el tiempo empleado por esta rutina, debe saber que un ciclo de instrucción corresponde a cuatro ciclos del reloj oscilador; si la frecuencia del oscilador es de 4 MHz, la frecuencia del reloj de instrucciones será entonces de 1 MHz, en cuyo

caso el ciclo de instrucción es de 1 microsegundo. Para el ejemplo anterior, se tendría entonces que la rutina de retardo se tardaría un poco más de 765520 μ s, más que suficientes para visualizar el encendido y apagado de un LED.

Cuando se requiere gran precisión en el cumplimiento de rutinas de tiempo, se acude a un oscilador a cristal para el oscilador del microcontrolador. Cuando la precisión no es una preocupación, se puede acudir a una sencilla red RC, para proporcionar la frecuencia del oscilador. En este último caso, la frecuencia de oscilación dependerá de las tolerancias de la resistencia y del condensador, al igual que la temperatura y el nivel de voltaje de la alimentación. En caso de implementar una red RC, el fabricante sugiere para la resistencia valores comprendidos entre 5 K y 100 k, y para el condensador un valor mínimo de 20 pF.

End. Aunque ésta no es propiamente una instrucción, sí es necesaria para el ensamblador; si éste no la encuentra, emite un mensaje de error.

Así, estaría descrito todo el programa. Después de escribirlo en el editor de textos deberá ensamblarlo, con lo cual se genera el código objeto propio del microcontrolador. Si no ha cometido errores de sintaxis, estará listo para fijar este código en el microcontrolador por medio del sistema de desarrollo; para éste último paso debe asegurarse que los fusibles posean la siguiente configuración:

Osc	XT
Watchdog	OFF
CP	OFF
Power-Up-Timer	ON

Ejercicio 2: Secuenciador con cuatro LED's

Este ejercicio, con un número mayor o menor de LEDs, es de gran utilidad para desarrollar habilidades en el campo de la programación. Nuevamente basamos el ejercicio en el entrenador PIC K-061, cuyo diagrama se encuentra al final del capítulo II “Conozca el PIC16C84”.

Se trata de encender cuatro LEDs de manera consecutiva, sin que permanezcan encendidos dos de ellos simultáneamente. Si se observa el diagrama del entrenador, los ocho pines del puerto B han sido dispuestos como salidas, con el objeto de controlar LED's, necesitando sólo una resistencia en serie con éste para limitar la corriente. El integrado ULN2803 se encarga de aislar el circuito y de entregar una buena cantidad de corriente, suficiente para proporcionar iluminación adecuada a los LED's.

En la figura 5 se tiene el diagrama de flujo correspondiente para la implementación del programa. El diagrama muestra que este es un proceso que, después de iniciado, no tiene una terminación; cada vez que se termina la secuencia se vuelve a iniciar, a menos que apliquemos un *reset* al dispositivo o apaguemos la fuente de alimentación.

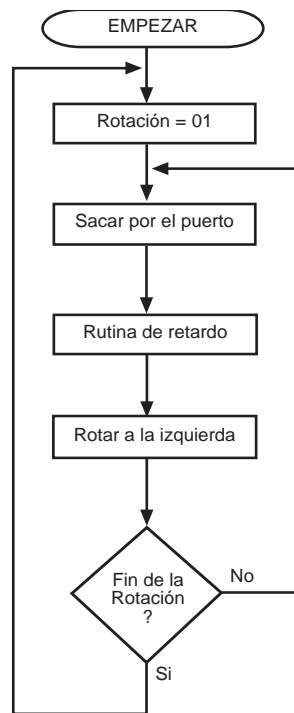


Figura 5. Diagrama de flujo del ejercicio 2

Para el programa son válidas las anotaciones hechas al anterior, salvo el hecho que se está utilizando una posición adicional de memoria RAM para almacenar un valor que corresponde al estado de la rotación presente en el puerto; ésta es la posición de memoria 0A, a la cual hemos llamado *rota* que pretende, también, ser de más fácil recordación y manejo que el número hexadecimal 0A.

En esta oportunidad, la parte que conviene resaltar es la rotación a la izquierda del registro *rota*, para ser sacado por el puerto B; si después de la rotación, el quinto bit aún no es uno, quiere decir que la rotación todavía no ha llegado a su fin, puesto que el ejercicio se hace sólo con cuatro LED's. La instrucción que consulta la condición es *btfs rota,4*; si ella se cumple (el bit es uno), el programa omite la instrucción *goto hol* y ejecuta la instrucción *goto otra*, empezando de nuevo la secuencia; si no se cumple la condición (este bit es cero), el programa ejecuta la instrucción *goto hol*, procediendo a sacar el dato rotado por el puerto B. La rutina de retardo es la misma del ejemplo anterior; por lo tanto el tratamiento hecho a los registros y valores es idéntico.

Al momento de fijar este programa en el microcontrolador, como en el caso anterior, se deben tener los fusibles configurados de la siguiente manera:

Osc	XT
Watchdog	OFF
CP	OFF
Power-Up-Timer	ON

```

;=====
;Este programa maneja una secuencia de cuatro LED's

list      p=16c84          ;se utiliza el PIC16C84
;-----

ptob      equ      06      ;el puerto b ésta en la dirección 06 de la
                           ;memoria RAM
rota      equ      0a      ;aquí se asignan nombres para reemplazar los
                           ;registros
reg1      equ      0b
reg2      equ      0c
reg3      equ      0d
valor1    equ      30      ;aquí se asignan nombres a los valores
valor2    equ      40      ;constantes
valor3    equ      10
;-----

reset     org      0       ;el vector de reset es la dirección 00 de la
                           ;memoria de programa
          goto     inicio  ;se envía al inicio del programa
          org      7       ;empieza el programa en la posición de
                           ;memoria 7
inicio    movlw    00      ;se carga el registro W con el valor 00
          tris     ptob    ;se programan los pines del puerto B como
                           ;salidas
otra      movlw    01      ;se carga el registro W con el valor 01
          movwf    rota    ;se dispone el dato para la rotación
hol       movf     rota,0   ;mueve el contenido del registro rota al
                           ;registro W
          movwf    ptob    ;enciende el LED correspondiente
          call     retardo  ;se mantiene el LED encendido por un momento
          rlf      rota    ;desplaza el contenido del registro un bit
                           ;a la izquierda
          btfss    rota,4  ;probar si el quinto bit se activa
          goto     hol     ;si no ha llegado al quinto bit se enciende
                           ;el LED que sigue
          goto     otra    ;si ya se encendió el último LED se inicia
                           ;nuevamente la secuencia
          ;esta rutina genera un tiempo de espera
retardo    movlw    valor1 ;se carga el registro W con el número valor1,
                           ;es decir con 30
          movwf    reg1    ;se traslada ese número al registro reg1
tres       movlw    valor2 ;se carga el registro W con el número valor2,
                           ;es decir con 40
          movwf    reg2    ;se traslada ese número al registro reg2
dos        movlw    valor3 ;se carga el registro W con el número valor3,
                           ;es decir con 50
          movwf    reg3    ;se traslada ese número al registro reg3
uno        decfsz   reg3    ;se decrementa el registro reg3 y si el valor
                           ;es cero se salta a la
          goto     uno     ;instrucción siguiente, si no es cero se
                           ;continúa decrementando
          decfsz   reg2    ;igual que el anterior
          goto     dos
          decfsz   reg1    ;igual que el anterior
          goto     tres
          retlw    0
          end
;=====

```

Figura 6. Programa secuenciador con 4 LED's (ejer2.asm)

Ejercicio 3: Leer los pulsadores y encender el LED correspondiente

Los ejercicios anteriores sólo han utilizado uno de los puertos como salida, no aprovechando las posibilidades de lectura de estos. Este ejercicio pretende entonces iniciarnos en la lectura de los puertos y tomar decisiones con respecto al valor obtenido de ella.

En el diagrama esquemático del entrenador K-061 se puede observar que hay dos pulsadores conectados al puerto A del microcontrolador, cada vez que se oprima uno de estos se debe encender el LED correspondiente del puerto B. Las teclas de lectura están conectadas a fuente, a través de resistencias de 4.7K, así cuando ninguna de ellas esté presionada, la lectura del puerto A será 03. En la figura 8 se tiene el diagrama de flujo correspondiente a este proceso, el cual servirá de base para la realización del programa respectivo.

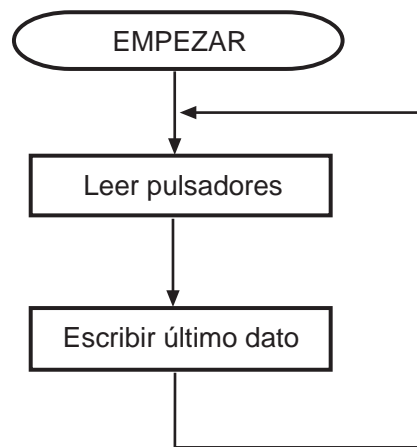


Figura 7. Diagrama de flujo del ejercicio 3

Para este tercer programa también son válidas algunas de las anotaciones hechas al primero, salvo que al registro 02 o contador de programa se le ha dado el nombre de *pc*; a un registro auxiliar para almacenar el último estado registrado de los pulsadores se le ha denominado *leido* (posición de memoria RAM 0A). Al fijar este programa en el microcontrolador, hemos utilizado la siguiente configuración de fusibles:

Osc	XT
Watchdog	OFF
CP	OFF
Power-Up-Timer	ON

No hacemos ninguna anotación al programa, ya que consideramos que con los dos anteriores se tienen bases para éste, además de los comentarios realizados al final de cada instrucción.

```

;=====
;Este programa lee dos pulsadores y enciende dos LED's, según el que se oprima
;se encenderá el LED correspondiente

list    p=16c84           ;se utiliza el PIC16C84
;-----

pc      equ    02          ;dirección del contador de programa
ptoa    equ    05          ;el puerto a está en la dirección 05 de la
                           ;memoria RAM
ptob    equ    06          ;el puerto b está en la dirección 06 de la
                           ;memoria RAM
leido    equ    0a         ;aquí se asignan nombres para reemplazar los
                           ;registros
;-----

reset   org    0           ;el vector de reset es la dirección 00 de la
                           ;memoria de programa
        goto   inicio     ;se envía al inicio del programa
        org    7           ;empieza el programa en la posición de
                           ;memoria 7
inicio  movlw   0f          ;se carga el registro W con el valor 0f
        tris   ptoa        ;se programan los pines del puerto A como
                           ;entradas
        movlw   00         ;se carga el registro W con el valor 00
        tris   ptob        ;se programan los pines del puerto B como
                           ;salidas
        movlw   00         ;se carga el registro W con el valor 00
        movwf  leido       ;se dispone el dato en alto para encender los
                           ;LED's
hol     movf    leido,0     ;mueve el contenido del registro leido al
                           ;registro W
        movwf  ptob        ;enciende el LED correspondiente
        movf    ptoa,0     ;lee el puerto A y guarda el dato en el
                           ;registro W
        xorlw   03         ;invierte el dato que leyó
        andlw   03         ;deja solamente los dos bits de interés
        addwf   pc         ;el valor que se leyó se suma al contador de
                           ;programa
        goto   hol        ;si no oprimen tecla, regresa a leer
        nop
        nop
        movwf  leido       ;carga valor que leyó en el registro leido
        goto   hol        ;actualiza el dato de salida en el puerto B y
                           ;vuelve a leer

        end

;=====

```

Figura 8. Programa lector de teclas del ejercicio 3 (ejerc3.asm)

Programa avanzado utilizando el entrenador K-061

Este ejercicio es una pequeña muestra de las posibilidades que se tienen cuando se usa un microcontrolador PIC; además utiliza los recursos que brinda el entrenador K-061. Este ejemplo es un poco avanzado, pero se da para animar al lector a seguir adelante y a estudiarlo para comprender cada una de sus partes. Si el lector lo prefiere lo puede dejar para practicar al final del nivel básico.

```

=====
;Este programa realiza una secuencia luminosa en los LED's, al llegar al
;último se emite un sonido. Además, cuando se oprime una tecla se cambia
;el sentido de giro de la secuencia

list      p=16c84

;-----

;las que vienen a continuación son una serie de equivalencias, las
;cuales toma el ensamblador y las reemplaza por el número asociado, en
;aquellos sitios en los cuales se hace referencia

carry     equ      0           ;estas hacen referencia a posiciones de bits
cero      equ      2           ;
bandera   equ      6           ;
rtcc      equ      1           ;registros que se utilizan en el programa
pc        equ      2           ;
sta       equ      3           ;
port_a    equ      5           ;
port_b    equ      6           ;
rota      equ      18h         ;
cuenta    equ      19h         ;
tiempo    equ      1ah         ;
reg1      equ      1bh         ;
reg2      equ      1ch         ;
reg3      equ      1dh         ;

;-----

inicio    bcf      sta, bandera ;limpiar bandera desplazamiento
          movlw    0           ;cargar a w con cero
          tris     port_b      ;puerto b como salidas
          movlw    3           ;cargar a w con 3
          tris     port_a      ;dos líneas bajas como entradas
          movlw    0feh        ;disponer dato para la rotación
          movwf    rota        ;cargar en registro 18
          movlw    1           ;dato del led encendido
          movwf    tiempo      ;carga en registro 1a

ceros     movlw    2           ;cargar a w con número dos
          movwf    reg3        ;y luego al registro 1d

uno       movf     port_a,w     ;leer puerto a
          andlw    3           ;elimina la parte alta
          xorlw    3           ;invierte el dato
          addwf    pc          ;se suma al pc para salto
          goto     dos_a       ;si no hubo tecla
          goto     siete_a     ;si tecla derecha
          goto     siete_b     ;si tecla izquierda
          goto     siete_c     ;si las dos teclas

dos_a     decfsz   reg1        ;decrementa registro y salta si cero
          goto     uno         ;si no cero lee de nuevo teclado
dos_b     decfsz   reg2        ;decrementa registro y salta si cero
          goto     uno         ;si no cero lee de nuevo teclado
dos_c     decfsz   reg3        ;decrementa registro y salta si cero
          goto     uno         ;si no cero lee de nuevo teclado

```

```

dos_d   bsf      sta,carry    ;activa o coloca carry en 1
        btfsc   sta, bandera ;salta si bandera está desactivada
        goto    cuatro      ;si activada bifurca a cuatro

tres     rlf      rota       ;rota a la izquierda registro de leds
        btfss   sta,carry    ;salta si carry está en 1
        goto    tres_b      ;si es cero -límite rotación- va a tres_b
tres_a   incf     tiempo     ;incrementa registro del led encendido
        goto    seis        ;bifurca a para sacar dato

tres_b   rlf      rota       ;rota una vez más registro de leds
        movlw   1           ;carga a w con 1
        movwf   tiempo     ;para indicar primer led encendido
        call    pito        ;señal sonora por límite rotación
        goto    seis        ;bifurca para sacar dato

cuatro   rrf      rota       ;rota a la derecha registro de leds
        btfss   sta,carry    ;salta si carry está en uno
        goto    cuatro_b    ;si cero -límite rotación- va a cuatro_b
cuatro_a decf     tiempo     ;decrementa registro led encendido
        goto    seis        ;bifurca para sacar dato

cuatro_b rrf      rota       ;rota una vez más registro de leds
        movlw   8           ;carga a w con 8
        movwf   tiempo     ;para indicar último led encendido
cinco    call     pito        ;señal sonora por límite rotación
        goto    seis        ;bifurca para sacar dato

seis     movf     rota,w      ;carga a w con dato led a encenderse
        movwf   port_b      ;escribe dato en el puerto b
        goto    ceros       ;reinicia para ir por siguiente dato

siete_a  bsf      sta,bandera ;activa bandera -rotación derecha-
        goto    dos_a       ;bifurca a leer el teclado

siete_b  bcf      sta,bandera ;desactiva bandera -rotación izquierda-
        goto    dos_a       ;bifurca a leer el teclado

siete_c  movf     tiempo,w    ;carga valor led encendido en w
        rlf     tiempo       ;multiplica por dos (valor x 2)
        rlf     tiempo       ;multiplica por dos (valor x 4)
        rlf     tiempo       ;multiplica por dos (valor x 8)
        addwf   tiempo       ;se suma una vez más (valor x 9)
        addwf   tiempo       ;se suma una vez más (valor x 10)

ocho     comf     tiempo,w    ;se invierte el dato y se carga en w
        movwf   port_b      ;se saca tiempo faltante en los leds
        clrf    cuenta      ;se limpia registro
        call    base        ;se llama base de un segundo
        decfsz  tiempo      ;decrementa el tiempo y salta si cero
        goto    ocho        ;si no cero vuelve a mostrar tiempo
        movlw   0ffh        ;carga a w con b'11111111'
        movwf   port_b      ;para apagar todos los leds
nueve    call     pito        ;se llama a señal sonora
        movlw   4           ;b'0100' carga a w con 4
        movwf   port_a      ;para activar el relé
        sleep                    ;ir al modo de reposo

```

```

pito    movlw    8            ;b'1000'  cargar a w
        movwf    port_a      ;para activar el pito
        movlw    d'110'      ;cargar con el decimal 110
        movwf    cuenta      ;para que el pito
        call     base         ;dure 1 segundo/(125-110)

        clrf     port_a      ;dejar el puerto en b'0000'
        retlw    3           ;regresar de la interrupción

base    clrf     rtcc         ;dejar en cero el rtcc
        movlw    5           ;cargar a w con b'101', para cargar el
        option   rtcc        ;rtcc interno con una preescala de 64

bas     movf     rtcc,w       ;leer el rtcc
        xorlw    d'125'      ;comparar con el decimal 125
        btfss    sta,cero    ;si es igual salta
        goto     bas         ;si no igual, vuelve a leer
        incf     cuenta      ;incrementa la cuenta
        movf     cuenta,w    ;carga a w con valor de cuenta
        xorlw    d'125'      ;comparar con el decimal 125
        btfss    sta,cero    ;si es igual salta
        goto     base        ;si no igual, vuelve a configurar cuenta
        retlw    4           ;al terminar regresa

end

;-----
;      Fusibles de programación

;      Osc                XT
;      Watchdog            OFF
;      CP                  OFF
;      Power-Up-Timer      ON
;=====
;=====

```

Figura 9. Programa avanzado utilizando el entrenador K-061(avanzado.asm)

Los ejercicios descritos en este capítulo se encuentran en el disquete que acompaña al curso, tanto el listado con extensión *.asm*, como el archivo ensamblado con extensión *.hex*. El lector puede, como ejercicio, utilizarlos para comprobar su funcionamiento y, después, hacerles los cambios que desee para modificar los tiempos de retardo, los pines de salida, etc., con el fin de profundizar en su estudio.

