Capítulo 2

Proyectos con el PIC16F84

- ***Conexión de LED y dipswitch***
- ***Manejo de un display de siete segmentos***
- ***Multiplexaje de teclados y displays***
- ***Conexión de memorias seriales al PIC***
- ***Manejo de un módulo LCD***
- ***Comunicación serial RS-232***
- ***Características especiales de los PIC***
  - *Interrupciones*
  - *Watchdog timer*
  - *EEPROM de datos del PIC16F84*
- ***Control de un motor paso a paso***

Proyecto N° 1: Conexión de LED y dipswitch

Como un ejercicio práctico que nos introduzca de manera rápida y sencilla en el manejo de los microcontroladores PIC, vamos a realizar un montaje simple, el cual consiste en conectar cuatro interruptores (dipswitch) como entradas del microcontrolador y cuatro LED como salidas. El programa que se escriba se debe encargar de verificar el estado de los dipswitch y de acuerdo a este, encender los LED. Este ejemplo aunque es muy simple, pero es fundamental para ejercitar el manejo de los puertos. La figura 2.1 muestra el diagrama esquemático del circuito.

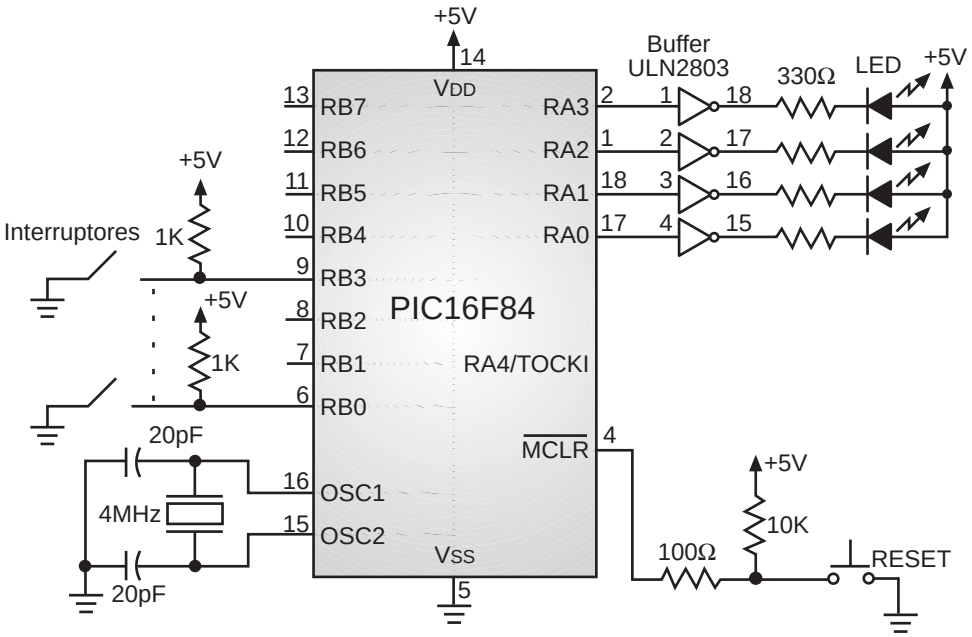


Figura 2.1. Conexión de los LED y dipswitch.

Debe notarse que los interruptores tienen resistencias conectadas a la fuente de alimentación, estas sirven para fijar un nivel alto cuando el dipswitch no está haciendo contacto. En este caso, cuando no se encuentra cerrado ningún interruptor el microcontrolador lee “unos” y cuando alguno se encuentre cerrado se leerá un “cero”. Por otra parte, para encender los LED se utiliza un circuito integrado ULN2803, el cual tiene un conjunto de transistores que invierten el pulso y amplifican la corriente. Por lo tanto, el pulso para encender un LED debe ser positivo.

Dado lo anterior, cuando se lee el estado de los dipswitch se debe invertir el valor leído, para asegurarse que el interruptor que esté cerrado se convierta en una señal positiva para encender el LED correspondiente. En la figura 2.2 se muestra el diagrama de flujo correspondiente al ejercicio y en la figura 2.3 el programa respectivo.

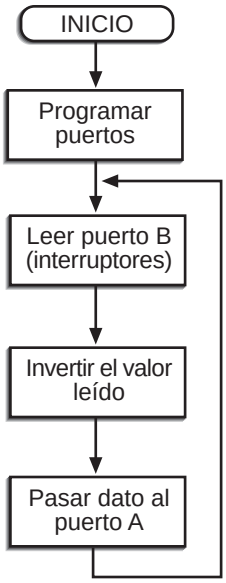


Figura 2.2. Diagrama de flujo para la conexión de los LED y dipswitch

```
;Este programa lee el estado de 4 interruptores y de acuerdo a ello enciende o
;no 4 LED
;En caso de que un número se escriba D'15': significa número decimal
;En caso de que el número se escriba B'00010101': significa número binario
;En caso de que un número se escriba 15H: significa número hexadecimal
;Si no se especifica nada, se supone numeración hexadecimal
;definición de registros
pc        equ       02h
status    equ       03h
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B
w         equ       00h       ;indica que el resultado se guarda en W

reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa

          org       5         ;el programa empieza en la dirección de memoria 5

inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f
          movwf     trisa     ;se programan los pines del puerto A como salidas
          movlw     0ffh      ;se carga el registro W con ff
          movwf     trisb     ;se programan los pines del puerto B como entradas
          bcf       status,5  ;se ubica en el primer banco de memoria RAM

ciclo     movf      ptob,w    ;el valor de puerto B lo pasa al registro W
          xorlw     0ffh      ;con una operación xor se invierte el valor
                              ;del dato leído del puerto B
          movwf     ptoa      ;pasa el valor de W al puerto A
          goto      ciclo

          end
;=========================================================================
;         Fusibles de programación
;         Osc                 XT
;         Watchdog            OFF
;         Code protect        OFF
;         Power-Up-Timer      ON
;         Micro.              PIC16F84
;=========================================================================
```

Figura 2.3. Programa de la conexión de LED y dipswitch

Proyecto N° 2: Manejo de un display de siete segmentos

Los displays de siete segmentos son un elemento muy útil en el diseño de aparatos electrónicos, por ejemplo, cuando se requiere visualizar el dato proveniente de un conteo, de una temporización, el estado de una máquina, etc. El ejercicio que vamos a realizar consiste en hacer un contador decimal (de 0 a 9), el cual lleva el conteo del número de veces que se oprime una tecla (pulsador). Para manejar el display utilizaremos un decodificador 9368, que es compatible con el tradicional 7448, pero decodifica de binario a hexadecimal, es decir que puede mostrar los caracteres de A hasta F. En el ejercicio el microcontrolador debe encargarse de verificar cuando el conteo llega a 9 para empezar nuevamente en 0.

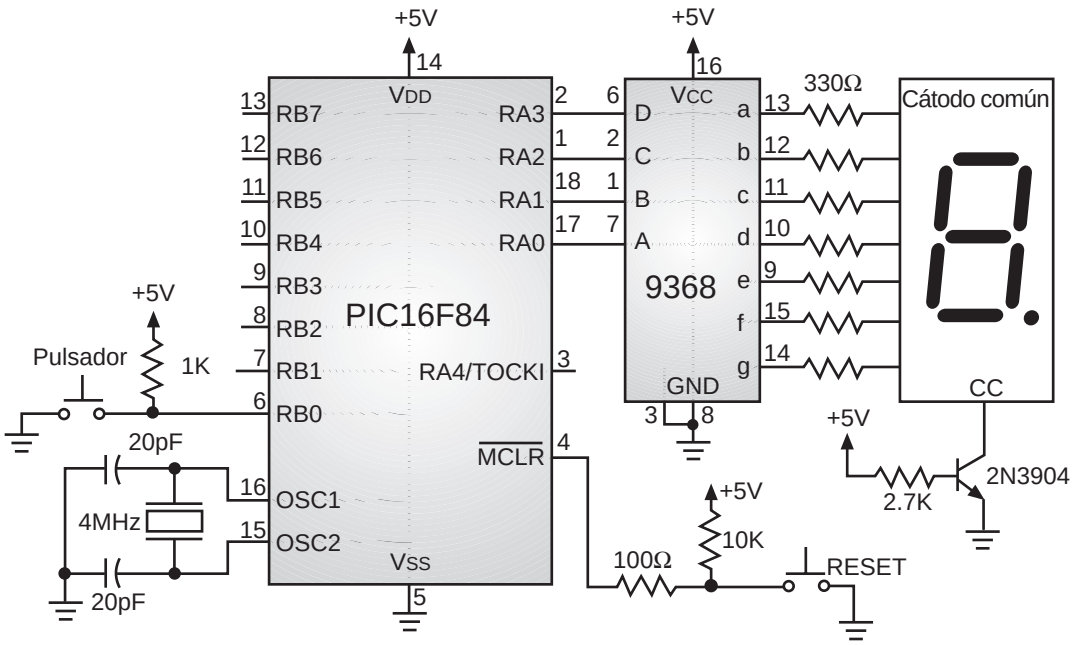


Figura 2.4. Manejo de un display de siete segmentos

El display utilizado es de cátodo común, para aumentar su visibilidad se conecta un transistor NPN que le entrega una buena corriente. En la figura 2.4 se muestra el diagrama correspondiente, en la figura 2.5 el diagrama de flujo y en la figura 2.6 el programa que realiza el control de las funciones.

> **Nota:** Si se usa el decodificador 7448 en lugar del 9368, el pin 3 se debe dejar al aire

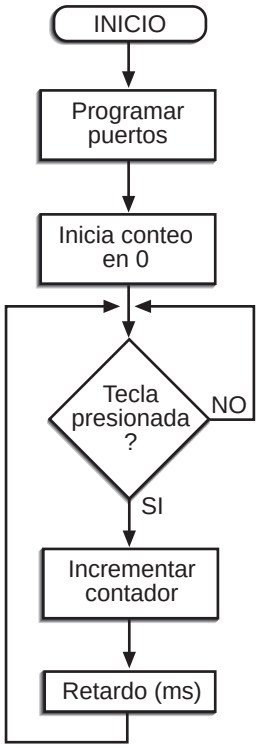


Figura 2.5. Diagrama de flujo del contador decimal

```
;Este programa hace un contador decimal en
;un display de 7 segmentos.
;En caso de que un número se escriba D'15': significa número decimal
;En caso de que el número se escriba B'00010101': significa número binario
;En caso de que un número se escriba 15H: significa número hexadecimal
;Si no se especifica nada, se supone numeración hexadecimal
;definición de registros
status    equ       03h       ;registro de estados
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
conta     equ       0ch       ;lleva el conteo de pulsaciones
loops     equ       0dh       ;utilizado en retardos (milisegundos)
loops2    equ       0eh       ;utilizado en retardos
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B
z         equ       02h       ;bandera de cero del registro de estados
reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa

          org       5         ;el programa empieza en la dirección de memoria 5
retardo   ;subrutina de retardo de 100 milisegundos
          movlw     D'100'    ;el registro loops contiene el número
          movwf     loops     ;de milisegundos del retardo
top2      movlw     D'110'    ;
          movwf     loops2    ;
top       nop
          nop
          nop
          nop
          nop
          nop
          decfsz    loops2    ;pregunta si terminó 1 ms
          goto      top
          decfsz    loops     ;pregunta si termina el retardo
          goto      top2
          retlw     0
```

```
inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f
          movwf     trisa     ;se programa el puerto A como salidas
          movlw     0ffh      ;se carga el registro W con ff
          movwf     trisb     ;se programa el puerto B como entradas
          bcf       status,5  ;se ubica en el primer banco de memoria RAM
          clrf      conta     ;inicia contador en cero
ciclo     movf      conta,w   ;el valor del contador pasa al registro W
          movwf     ptoa      ;pasa el valor de W al puerto A (display)
          call      retardo   ;retardo esperando que suelten la tecla

pulsa     btfsc     ptob,0    ;pregunta si el pulsador está oprimido
          goto      pulsa     ;si no lo está continúa revisándolo
          call      retardo   ;si está oprimido retarda 100 milisegundos
          btfsc     ptob,0    ;para comprobar
          goto      pulsa     ;si no lo está vuelve a revisar
          incf      conta     ;si lo confirma incrementa el contador
          movf      conta,w   ;carga el registro W con el valor del conteo
          xorlw     0ah       ;hace operación xor para ver si es igual a 0ah
          btfsc     status,z  ;prueba si el contador llegó a 0ah (diez)
          goto      inicio    ;si es igual el contador se pone en ceros
          goto      ciclo     ;si no llegó a diez incrementa normalmente
          end                 ;y actualiza el display

;=====================================================================
;         Fusibles de programación
;         Osc                 XT
;         Watchdog            OFF
;         Code protect        OFF
;         Power-Up-Timer      ON
;         Micro.              PIC16F84
;=====================================================================
```

Figura 2.6. Programa del contador decimal

Proyecto N° 3: Multiplexaje de teclados y displays

Uno de los problemas que con frecuencia enfrentan los diseñadores y experimentadores de los sistemas electrónicos es que algunas veces las líneas de entrada/salida que tienen disponibles en un dispositivo parecen no ser suficientes para una aplicación determinada; pero esto no siempre es verdad. En ocasiones, algunas técnicas y trucos pueden ayudarnos a optimizar las funciones de los microcontroladores, reduciendo el tamaño de los circuitos impresos y evitando la necesidad de conseguir circuitos integrados con mayor número de líneas I/O. Nuestro propósito en esta práctica, es proporcionar algunas técnicas que puedan ayudar a optimizar los diseños.

El multiplexaje, que se define como una forma de compartir secuencialmente el tiempo para que dos o más señales se puedan transmitir a la vez por un mismo medio conductor, es sin duda una gran herramienta (y en ocasiones la única) para conseguir un mejor aprovechamiento de un dispositivo. Nosotros la utilizaremos para la lectura de teclados y la visualización de información a través de displays de siete segmentos.

Manejo de teclados

Inicialmente consideremos la implementación de un teclado sencillo, el cual consta básicamente de 8 interruptores (dipswitch), tal como se muestra en la figura 2.7, en donde a cada pin del puerto B del microcontrolador corresponde una determinada tecla. Cuando estas teclas no están presionadas, el pin correspondiente estará conectado a un nivel lógico alto, en cambio cuando alguna de ellas se presiona, el pin correspondiente se conectará a un nivel lógico bajo; en este teclado por lo tanto se lee “ceros”. Un aspecto que vale la pena tener en cuenta es que si el microcontrolador tiene elementos *pull-ups* internos, las resistencias que se muestran pueden eliminarse, simplificando el circuito.

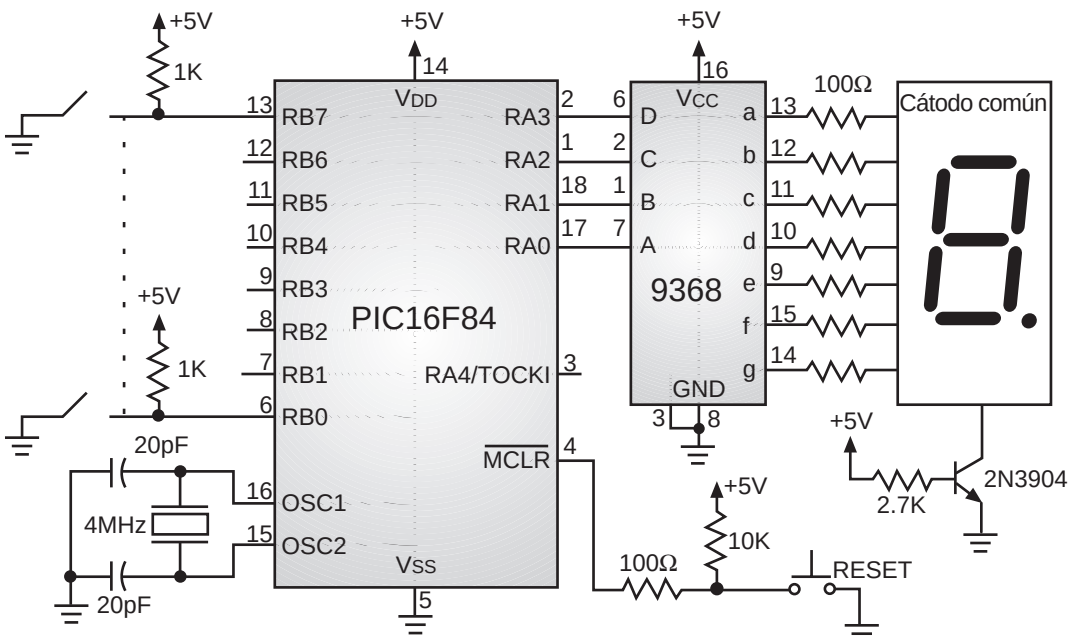


Figura 2.7. Lectura de un teclado sencillo

En la figura 2.8 se muestra el diagrama de flujo para la lectura de un teclado de esta naturaleza; observe como el proceso se queda enclavado mientras no detecta tecla presionada. En la figura 2.9 se muestra el programa respectivo, el cual asigna un valor numérico, comprendido entre 0 y 7, a la tecla presionada. El valor obtenido se lleva a un display de siete segmentos para comprobar que el programa funciona correctamente. El lector podrá determinar que pasa cuando dos o más teclas se presionan "simultáneamente", la prioridad que existe entre ellas y como puede modificarse ésta.

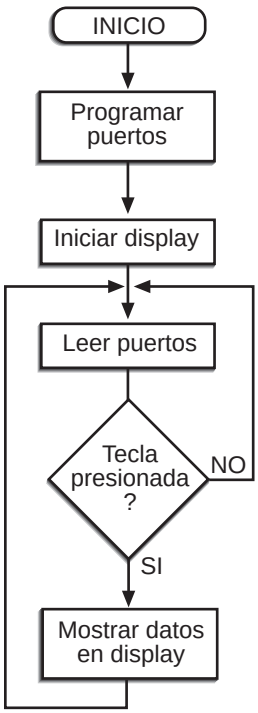


Figura 2.8. Diagrama de flujo para la lectura de un teclado sencillo

```
;Este programa lee un teclado sencillo compuesto por 8 interruptores y maneja
;un display de 7 segmentos.
;En caso de que un número se escriba D'15': significa número decimal
;En caso de que el número se escriba B'00010101': significa número binario
;En caso de que un número se escriba 15H: significa número hexadecimal
;Si no se especifica nada, se supone numeración hexadecimal
;definición de registros
status    equ       03h       ;registro de estados
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
conta     equ       0ch       ;contador de rotaciones para identificar la tecla
loops     equ       0dh       ;utilizado en retardos (milisegundos)
loops2    equ       0eh       ;utilizado en retardos
rota      equ       0fh       ;registro que se rota para encontrar la tecla
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B
z         equ       02h       ;bandera de cero del registro de estados
c         equ       00h       ;banderq de carry del registro de estados
w         equ       00h       ;indica que el resultado se guarda en W

reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa
          org       5         ;el programa empieza en la dirección de memoria 5

retardo   ;subrutina de retardo de 100 milisegundos
          movlw     D'100'    ;el registro loops contiene el número
```

```
          movwf     loops     ;de milisegundos del retardo
top2      movlw     D'110'    ;
          movwf     loops2    ;
top       nop
          nop
          nop
          nop
          nop
          nop
          decfsz    loops2    ;pregunta si terminó 1 ms
          goto      top
          decfsz    loops     ;pregunta si termina el retardo
          goto      top2
          retlw     0

inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f
          movwf     trisa     ;se programan los pines del puerto A como salidas
          movlw     0ffh      ;se carga el registro W con ff
          movwf     trisb     ;se programan los pines del puerto B como entradas
          bcf       status,5  ;se ubica en el primer banco de memoria RAM

          movlw     0ffh      ;si no hay tecla oprimida se muestra una F
          movwf     conta     ;en el display
ciclo     movf      conta,w   ;el valor del contador pasa al registro W
          movwf     ptoa      ;pasa el valor de W al puerto A (display)
          call      retardo   ;retardo

leer      movf      ptob,w    ;lee el puerto de los interruptores
          xorlw     0ffh      ;invierte el dato leído
          btfsc     status,z  ;pregunta si el resultado de la inversión dió cero
          goto      inicio    ;si no hay tecla oprimida borra display y
                              ;vuelve a leer
          movwf     rota      ;lleva valor de tecla oprimida a registro rota
          clrf      conta     ;inicializa el contador de rotaciones
sigue     rrf       rota      ;se rota el dato para buscar en que posición
                              ;se encuentra el interruptor activado
          btfsc     status,c  ;pregunta si el carry es 1 luego de la rotación
          goto      salir     ;si es uno esa es la tecla oprimida y va a indicar
                              ;en el display cual es su valor
          incf      conta     ;si el carry estaba en 0 luego de
                              ;rotar el registro
          goto      sigue     ;se vuelve a rotar y se vuelve a probar

salir     goto      ciclo     ;el valor de la tecla queda en el registro conta
                              ;y se pasa a W para mostrarlo en el display
          end

;=====================================================================
;         Fusibles de programación
;         Osc                 XT
;         Watchdog            OFF
;         Code protect        OFF
;         Power-Up-Timer      ON
;         Micro.              PIC16F84
;=====================================================================
```

Figura 2.9. Programa para la lectura de un teclado sencillo

Cuando el número de líneas I/O es suficiente, la configuración anterior simplifica el programa y podemos quedar satisfechos; pero cuando las líneas de entrada/salida empiezan a escasear, debemos pensar en rediseñar este teclado, optimizándolo. La figura 2.10 muestra una alternativa para este teclado, observe que para las mismas 8 teclas se tienen sólo 6 líneas de entrada/salida (nos hemos ahorrado 2 líneas, las cuales pueden ser aprovechadas para otros propósitos igualmente importantes); en la figura

2.11 se muestra un teclado de 16 elementos, precisando sólo 8 líneas de entrada/salida (necesitaríamos 8 líneas más si hubiésemos seguido el principio de diseño inicial). Estos dos últimos teclados tienen algo en común, están organizados matricialmente y para manejarlos se requiere el multiplexaje.

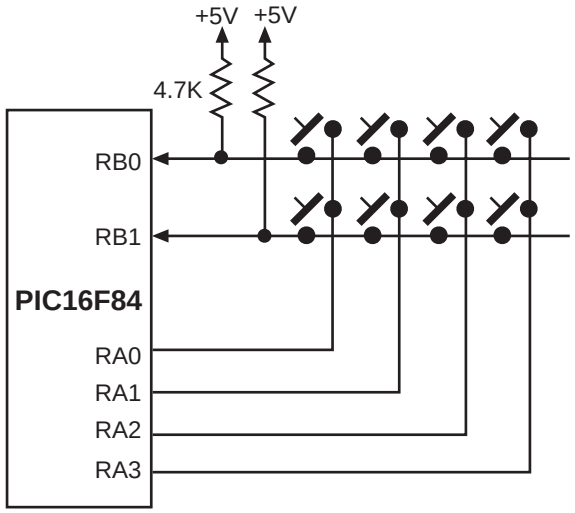


Figura 2.10. Teclado matricial de 2 filas x 4 columnas

Consideremos la figura 2.11, que muestra el teclado matricial de 4 filas por 4 columnas. En este caso, las líneas del microcontrolador correspondientes a las filas se han configurado como salidas y las correspondientes a las columnas como entradas.

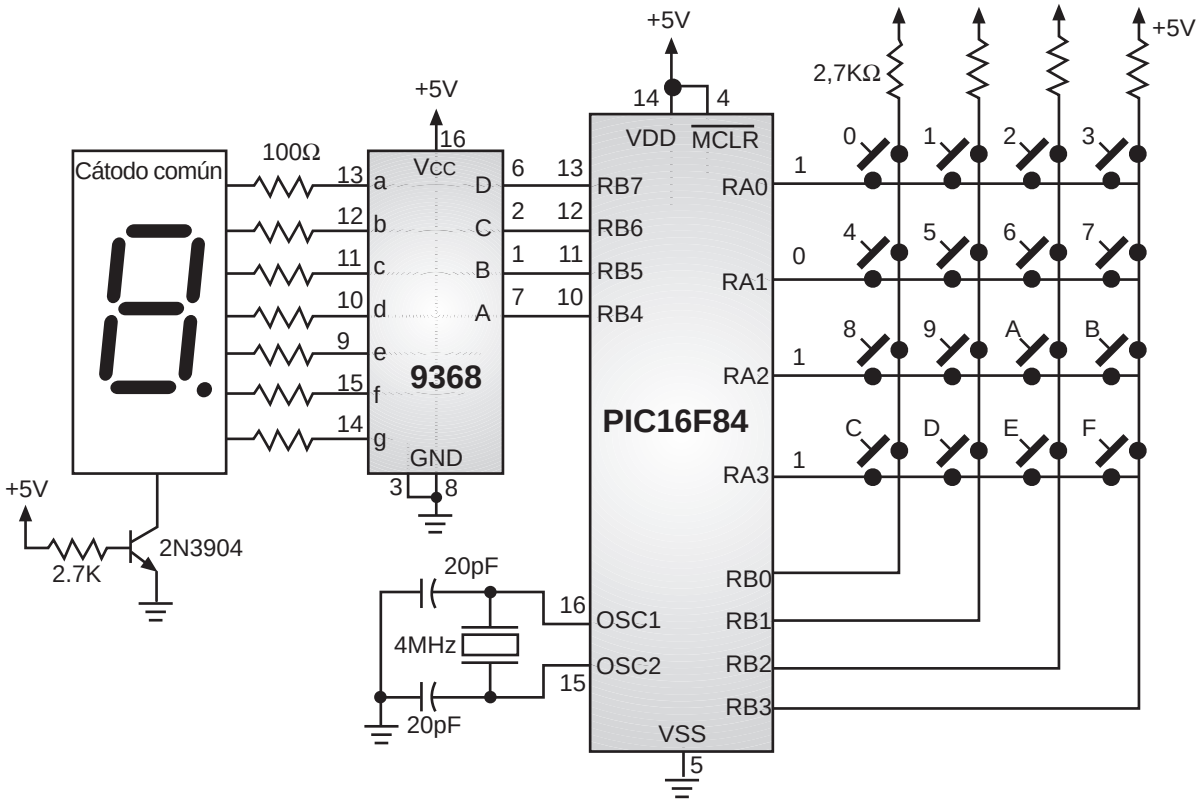


Figura 2.11. Teclado matricial de 4 filas x 4 columnas

Como se puede observar, normalmente, las líneas de entrada permanecen en un nivel lógico alto, gracias a los elementos *pull-up* (resistencias de 2.7K). La clave para manejar este tipo de teclados consiste en enviar por las líneas de salida sólo un cero por vez; por ejemplo si enviamos un cero por la línea RA1, cuando oprimimos una tecla de la segunda fila (el 4, 5, 6 ó 7), un nivel lógico bajo se reflejará en el pin correspondiente de las líneas de entrada (RB0, RB1, RB2 o RB3 respectivamente); así, si se encuentra un nivel lógico bajo en la línea RB3 podemos concluir que la tecla presionada fue el dígito 7.

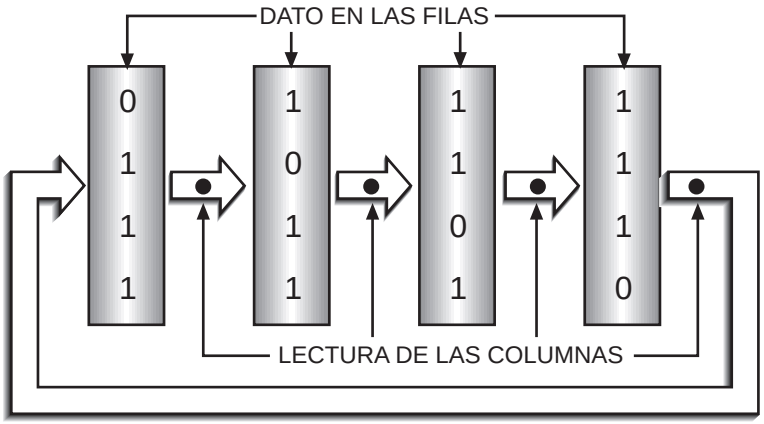


Figura 2.12. Secuencia para la lectura de un teclado matricial

Si queremos explorar todo este teclado, bastará con rotar el cero circularmente, de tal manera que solamente un cero se encuentre en las filas del teclado, cuando se realiza las lecturas de las líneas de entrada (las columnas) como se muestra en la figura 2.12. Cuando el cero llegue a la fila más significativa del teclado, debe reingresar en la próxima ocasión por la menos significativa, reiniciando la exploración del teclado. Un diagrama de flujo para este proceso, en donde el microcontrolador queda enclavado leyendo el teclado hasta que se detecta la presión de uno de sus elementos, se muestra en la figura 2.13; en la figura 2.14 se tiene el programa respectivo. Como resultado del programa, un valor comprendido entre 0 y 15 queda almacenado en un registro, dicho valor se muestra en un display de 7 segmentos que se ha conectado a los pines RB4 a RB7, para comprobar el funcionamiento del sistema.

El proceso se realiza a una gran velocidad, por lo que se tiene la sensación que todo el teclado se está sensando permanentemente. En el programa realizado, por ejemplo, la exploración total del teclado tarda menos de 60 μs, si consideramos que el oscilador es de 4 MHz.

Otro aspecto que no se puede olvidar, son los rebotes causados por la pulsación de una tecla. Cuando una tecla se oprime, sus contactos actúan como resortes, y la unión eléctrica no es estable; se generan una serie de uniones y desuniones mecánicas durante un intervalo significativo de tiempo. Estos rebotes pueden dar lugar a que, en una aplicación real, el programa los interprete como si se hubieran generado muchas pulsaciones, si es que no se toman los correctivos del caso. Para ello existen

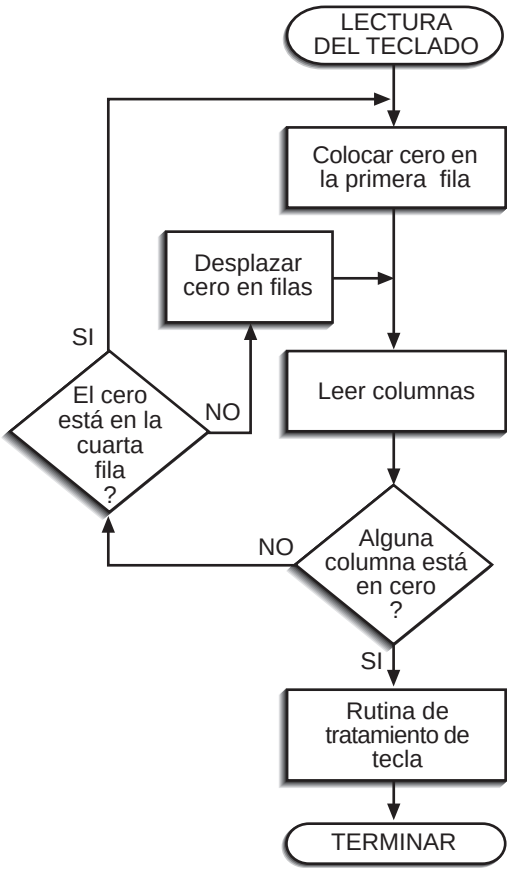


Figura 2.13. Diagrama de flujo para la lectura de un teclado matricial

soluciones de *hardware* y *software*, consideramos más interesantes las segundas ya que simplifican el diseño. Allí, la solución más obvia es que después de la detección de la tecla pulsada se genere un retardo en la lectura del teclado, de tal manera que se ignoren los contactos subsiguientes debidos a los rebotes.

Experimentalmente, se encuentra que un retardo aceptable tiene un valor comprendido entre los 100 a 125 ms; tiempos más pequeños pueden todavía interpretar los rebotes y tiempos más largos pueden tornar demasiado lento un teclado. En ocasiones, conviene también pensar en el tipo de usuarios de un sistema, ya que hay algunos de ellos que tienen la tendencia a mantener oprimida una tecla un tiempo más largo del común de las personas, lo que mal controlado en el programa puede dar lugar a un funcionamiento incorrecto del sistema.

```
;Este programa lee un teclado matricial de 4x4 y muestra la tecla
;oprimida en el display de 7 segmentos.
;definición de registros
pc        equ       02h       ;contador de programa
status    equ       03h       ;registro de estados
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
tecla     equ       0ch       ;contienen el valor de la tecla oprimida
loops     equ       0dh       ;utilizado en retardos (milisegundos)
loops2    equ       0eh       ;utilizado en retardos
rota      equ       0fh       ;registro que rota para enviar unos a las filas
filas     equ       10h       ;contiene el número de la fila a probar
```

```
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B
z         equ       02h       ;bandera de cero del registro de estados
c         equ       00h       ;bandera de carry del registro de estados
w         equ       00h       ;indica que el resultado se guarda en W

reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa
          org       5         ;el programa empieza en la dirección 5

retardo   ;subrutina de retardo de 100 milisegundos
          movlw     D'100'    ;el registro loops contiene el número
          movwf     loops     ;de milisegundos del retardo
top2      movlw     D'110'    ;
          movwf     loops2    ;
top       nop
          nop
          nop
          nop
          nop
          nop
          decfsz    loops2    ;pregunta si terminó 1 ms
          goto      top
          decfsz    loops     ;pregunta si termina el retardo
          goto      top2
          retlw     0

tabla     addwf     pc        ;sumar W al PC
          nop
          retlw     0         ;primera columna
          retlw     1         ;segunda columna
          nop
          retlw     2         ;tercera columna
          nop
          nop
          nop
          retlw     3         ;cuarta columna

inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f0h
          movwf     trisa     ;se programa el puerto A como salidas
          movlw     0fh       ;se carga el registro W con 0fh
          movwf     trisb     ;se programa el puerto B como entradas y salidas
          bcf       status,5  ;se ubica en el primer banco de memoria RAM
          movlw     00h       ;para empezar se muestra un 0 en el display
          movwf     tecla     ;
ciclo     swapf     tecla,w   ;intercambia 4 bits altos y bajos y quedan en W
          movwf     ptob      ;pasa el valor de W al puerto B (display)
          call      retardo   ;retardo
escan     clrf      filas
          movlw     b'1110'   ;se prepara para enviar ceros a las filas
          movwf     rota
probar    movf      rota,w    ;envia el dato a las filas
          movwf     ptoa
          nop                 ;tiempo para estabilidad de las líneas
leer      movf      ptob,w    ;leer las columnas conectadas al puerto B
          andlw     0fh       ;elimina la parte alta del byte leído
          xorlw     0fh       ;invierte el dato para ver si hay algún cero
          btfss     status,z  ;pregunta si el resultado es cero (alguna tecla)
          goto      salir     ;si hay tecla, mostrar en display
          btfss     rota,3    ;consulta si ya van 4 rotaciones
          goto      escan     ;si terminó, vuelve a empezar el escan de teclado
          bsf       status,c  ;coloca bit de carry en 1
          rlf       rota      ;para rotar el 0 que va a ir hacia las filas
          movlw     4         ;carga W con 4 para sumarlo al valor de filas
          addwf     filas,1
          goto      probar    ;va a hacer la próxima prueba con el 0 rotado
salir     call      tabla     ;para obtener valor de la columna
          addwf     filas,w   ;sumar columna y filas para obtener el dato real
```

```
          movwf     tecla     ;muestra el dato en display
          goto      ciclo
          end

;=====================================================================
;         Fusibles de programación
;         Osc                 XT
;         Watchdog            OFF
;         Code protect        OFF
;         Power-Up-Timer      ON
;         Micro.              PIC16F84
;=====================================================================
```

Figura 2.14. Programa para la lectura de un teclado matricial

Multiplexaje de displays de siete segmentos

Consideremos la estructura de la figura 2.15. Allí se tienen cuatro displays de siete segmentos, con punto decimal; un puerto de 4 bits (o la mitad de uno de 8) controla cuatro transistores NPN, que finalmente alimentan los cátodos de cada uno de los displays, en donde el bit menos significativo controla el display de menor peso. Un puerto de 8 bits maneja cada uno de los segmentos de los displays, en donde el bit menos significativo del puerto controla el segmento **a**, el siguiente el **b**, y así sucesivamente, hasta llegar al más significativo, que controla el punto decimal del display. Las resistencias tienen como objeto limitar la corriente que fluye a través de los segmentos y que ingresa a los pines del microcontrolador.

Por la configuración, es fácil deducir que cuando se tiene un nivel lógico bajo en la base del transistor éste se comporta como un interruptor abierto y no se presenta corriente entre emisor y colector; se necesita tener un uno lógico en

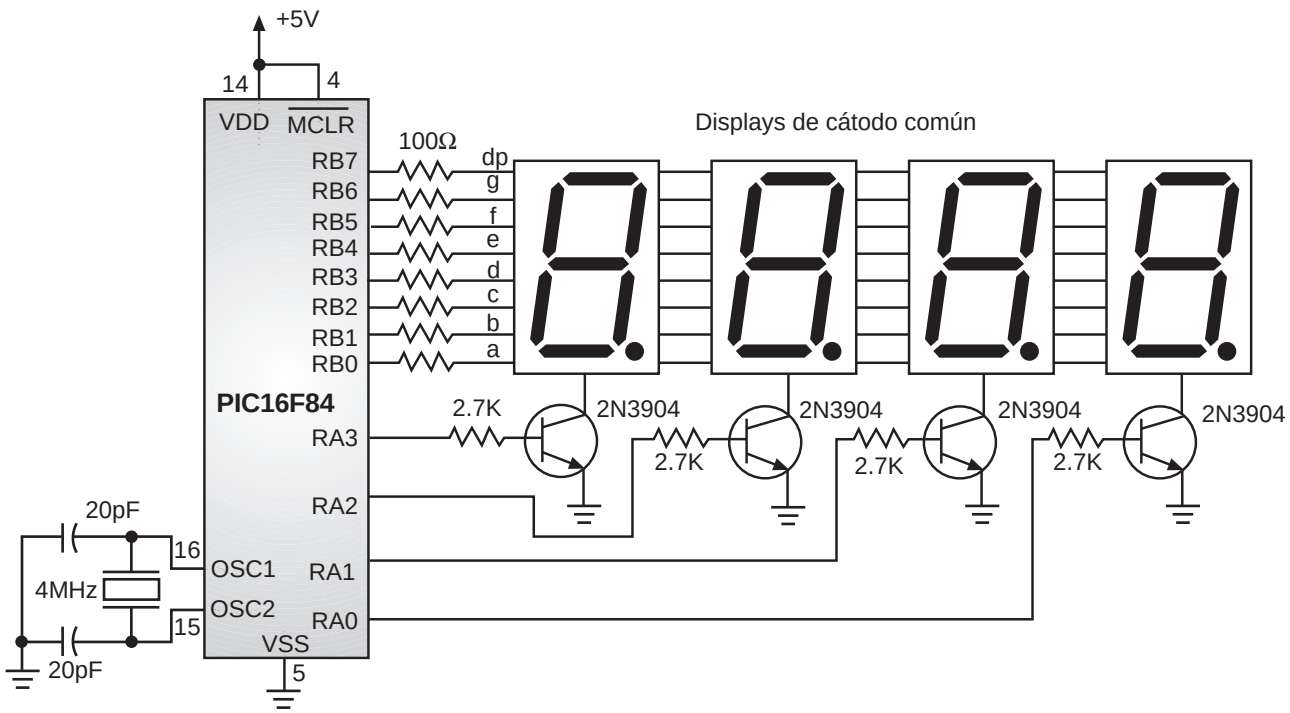


Figura 2.15. Configuración para manejo de displays de 7 segmentos

las bases de éstos para que el transistor se comporte como un interruptor cerrado y se presente dicha corriente. Pero aún cuando los transistores se comporten como interruptores cerrados, no se encenderá ningún display si la salida del puerto que maneja los segmentos tiene niveles lógicos bajos (ceros); para encenderlo, es necesario colocar un nivel lógico alto en el pin de salida correspondiente a cada segmento.

Así por ejemplo, si queremos mostrar un tres en el segundo display menos significativo debemos establecer básicamente los siguientes pasos:

- Colocar el número binario 0010 en el puerto que controla los transistores
- Enviar el número binario 01001111 por el puerto que controla los segmentos

Un buen observador encontrará que aunque los anteriores pasos consiguen el objetivo propuesto (mostrar en la pantalla el —3-, en donde la línea quiere decir espacios en blanco), este proceso puede conllevar efectos indeseados. Aquí, por ejemplo, si el puerto que controla los segmentos tenía un valor diferente del binario 00000000 o de 01001111, entre la ejecución de los dos pasos dados se mostrará en el display un número o un símbolo que es diferente del valor deseado. Por ello, es conveniente apagar momentáneamente los segmentos, de tal manera que nos permita seleccionar adecuadamente el display en cuestión y posterior a esto, enviar el dato correcto a los segmentos; por lo tanto, un paso 0 que se debe agregar a este proceso es colocar el número binario 00000000 en el puerto que controla los segmentos.

Otra opción que permite obtener el mismo resultado, con un proceso diferente, sería la siguiente:

0. Colocar el número binario 0000 en el puerto que controla los transistores
1. Enviar el número binario 01001111 por el puerto que controla los segmentos
2. Colocar el número binario 0010 en el puerto que controla los transistores

Ambos procesos conllevan al mismo objetivo propuesto, eliminando la posibilidad de los efectos indeseados.

Ahora, si pretendemos visualizar no uno sino los cuatro displays de siete segmentos, es necesario empezar a controlar los transistores secuencialmente a la vez que se envía por el puerto que controla los segmentos los datos correspondientes al display en cuestión, realizando este proceso a una velocidad tal que de nuevo parezca que el proceso se está realizando simultáneamente sobre todos los displays. El tiempo en que necesitamos sostener el dato en cada display puede variar significativamente, dependiendo fundamentalmente del valor de las resistencias limitadoras, del número de dígitos que se tengan por mostrar y de las características propias del display; experimentalmente se encuentra que mostrar cada dígito durante 3 milisegundos, cuando se tienen resistencias limitadoras de 100 ohm, proporcionan un brillo aceptable de un display “estándar” y una buena visualización a una distancia prudente.

El diagrama de flujo de la figura 2.16 muestra el proceso necesario para mostrar cuatro dígitos en un display y la figura 2.17 muestra el respectivo programa, el cual acude a tablas para consultar los segmentos que se deben encender en cada caso. La utilización de estas tablas permiten que se pueda alambrar de manera diferente las salidas del microcontrolador y las entradas de los segmentos del display; por ejemplo, se puede reorganizar la configuración de pines de tal manera que se simplifique el diseño del circuito impreso, bastando con reorganizar los valores de la tabla. La utilización de las tablas también nos permiten la generación de caracteres y signos especiales, ya que podemos controlar el encendido de todos y cada uno de los siete segmentos y el punto decimal.

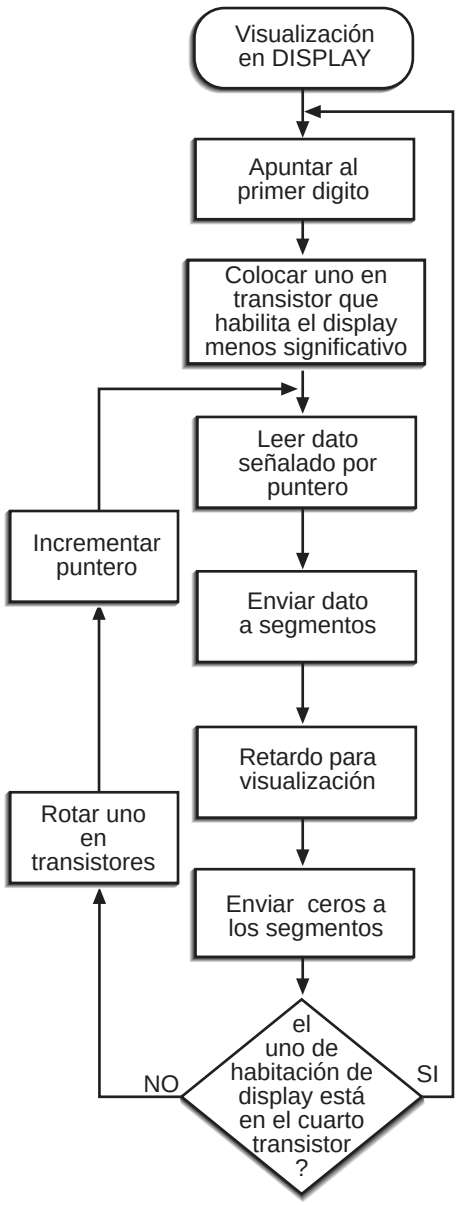


Figura 2.16. Diagrama de flujo para manejo de 4 displays sin decodificador

```
;Este programa maneja 4 displays de 7 segmentos
;en forma multiplexada.
;definición de registros
indo      equ       00h       ;registro de direccionamiento indirecto
pc        equ       02h       ;contador de programa
status    equ       03h       ;registro de estados
fsr       equ       04h       ;registro selector
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
loops     equ       0dh       ;utilizado en retardos (milisegundos)
loops2    equ       0eh       ;utilizado en retardos
rota      equ       0fh       ;rota el uno para habilitar displays
dig1      equ       10h       ;primer dígito a mostrar
dig2      equ       11h       ;segundo dígito a mostrar
dig3      equ       12h       ;tercer dígito a mostrar
dig4      equ       13h       ;cuarto dígito a mostrar
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B
z         equ       02h       ;bandera de cero del registro de estados
c         equ       00h       ;banderq de carry del registro de estados
w         equ       00h       ;indica que el resultado se guarda en W

reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa

          org       5         ;el programa empieza en la dirección de memoria 5

retardo   ;subrutina de retardo de 3 milisegundos
          movlw     03h       ;el registro loops contiene el número
          movwf     loops     ;de milisegundos del retardo
top2      movlw     D'110'    ;
          movwf     loops2    ;
top       nop
          nop
          nop
          nop
          nop
          nop
          decfsz    loops2    ;pregunta si termino 1 ms
          goto      top
          decfsz    loops     ;pregunta si termina el retardo
          goto      top2
          retlw     0

tabla     ;contiene los valores para encender segmentos en display de
          ;cátodo común (cuando no se utiliza decodificador 9368)
          addwf     pc                  ;sumar W al PC
          ;segmen.    .gfedcba          ;orden de los bits
          retlw     b'00111111'         ;0
          retlw     b'00000110'         ;1
          retlw     b'01011011'         ;2
          retlw     b'01001111'         ;3
          retlw     b'01100110'         ;4
          retlw     b'01101101'         ;5
          retlw     b'01111101'         ;6
          retlw     b'00000111'         ;7
          retlw     b'01111111'         ;8
          retlw     b'01101111'         ;9

inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     00h       ;se carga el registro W con 0f0h
          movwf     trisa     ;se programan los pines del puerto A como salidas
          movlw     00h      ;se carga el registro W con 00h
          movwf     trisb     ;se programa el puerto B como entradas y salidas
          bcf       status,5  ;se ubica en el primer banco de memoria RAM

          movlw     01        ;datos que se muestran en los displays
          movwf     dig1
          movlw     02
```

```
          movwf     dig2
          movlw     03
          movwf     dig3
          movlw     04
          movwf     dig4

          movlw     00        ;envía ceros a los transistores para apagarlos
          movwf     ptoa
empe      movlw     08h       ;iniciar un 1 en el registro de rotación
          movwf     rota
          movlw     dig1      ;con el registro selector (fsr) se apunta
          movwf     fsr       ;al primer dato que se va a mostrar
disp      movlw     00h       ;colocar en cero el dato del display
          movwf     ptob      ;para apagarlos
          movf      rota,w    ;pasa rotación del 1 al registro W
          movwf     ptoa
          movf      indo,w     ;lee el dato del registro apuntado por el fsr
          ;call     tabla     ;se utilizaría si no hubiera decodificador 9368
          movwf     ptob      ;envia dato leído al display
          call      retardo   ;retardo de 3 milisegundos para visualización
          btfsc     rota,0    ;pregunta si terminaron las 4 rotaciones
          goto      empe      ;si ya rotaron todos, vuelve a empezar
          bcf       status,c  ;pone el carry en 0 para que no afecte rotaciones
          rrf       rota      ;rota el 1 habilitador de displays
          incf      fsr       ;apunta al próximo dígito a mostrar
          goto      disp

          end

;=====================================================================
;         Fusibles de programación
;         Osc                 XT
;         Watchdog            OFF
;         Code protect        OFF
;         Power-Up-Timer      ON
;         Micro.              PIC16F84
;=====================================================================
```

Figura 2.17. Programa para manejo de 4 displays sin decodificador

Una variante para este sistema consistiría en utilizar un decodificador 9368 para manejar los displays de 7 segmentos, en este caso no se requiere la lectura de tablas y el dato correspondiente se puede llevar al puerto directamente. Debe recordarse que con este sistema se pueden mostrar números hexadecimales (entre 0 y F). El diagrama de conexión se muestra en la figura 2.18. El programa es el mismo de la figura 2.17, sólo se debe omitir la subrutina llamada TABLA y el sitio donde se hace el llamado (call TABLA), esto debido a que el dato se puede mostrar directamente.

Manejo simultáneo de teclados y displays

Una estructura más completa es la que se muestra en la figura 2.19. Observe como se utilizan doce líneas para implementar un teclado y la visualización en siete segmentos. Aquí, se está utilizando el decodificador 7447; de esta manera se ahorran al menos cuatro líneas (sólo tres si utilizamos una adicional para manejar el punto decimal); un puerto de 8 bits se comporta totalmente como salida, controlando simultáneamente los datos que se mostrarán por los displays, los que saldrán por las filas y las bases de los transistores, mientras que un puerto de cuatro bits se comporta como entrada, para leer las columnas del teclado. En este caso se utilizan displays de ánodo común y transistores PNP, por lo tanto, el cero que se rota para leer el teclado matricial sirve también para encender los displays.

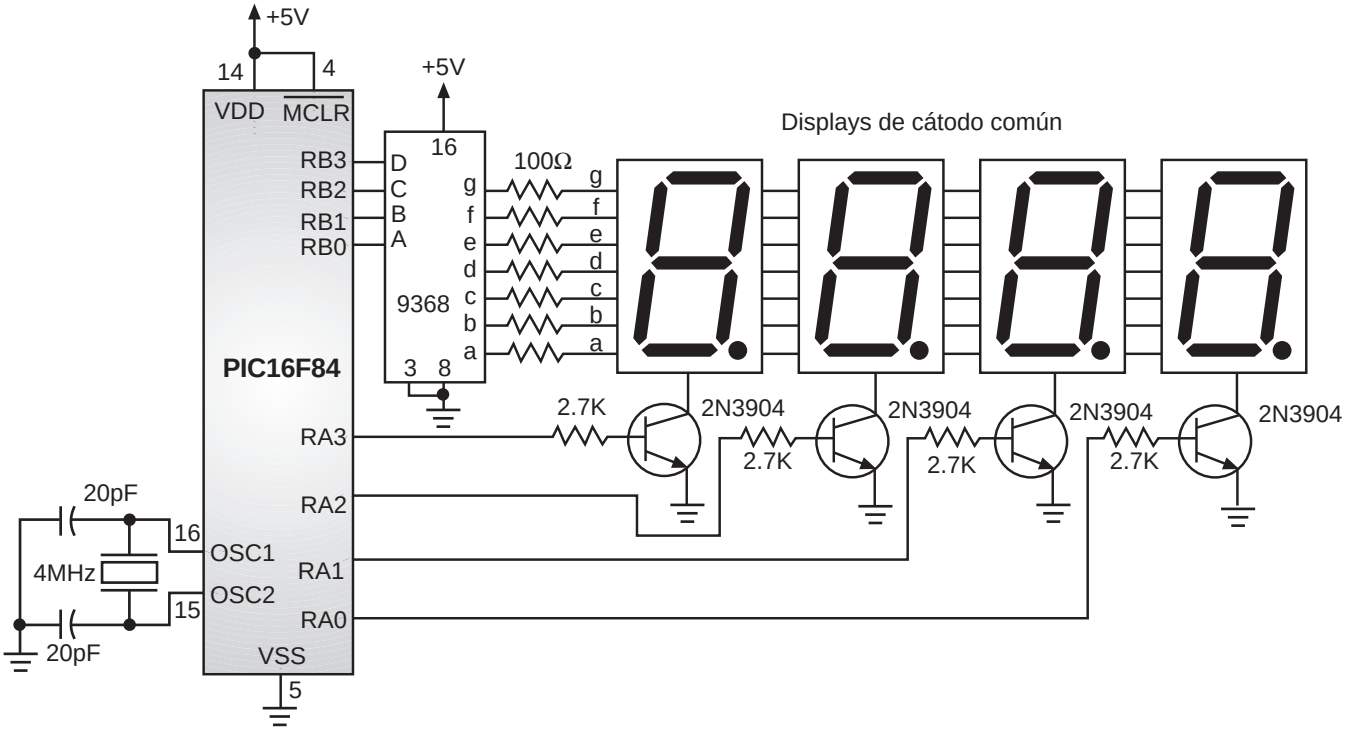


Figura 2.18. Conexión para el manejo de 4 displays con decodificador

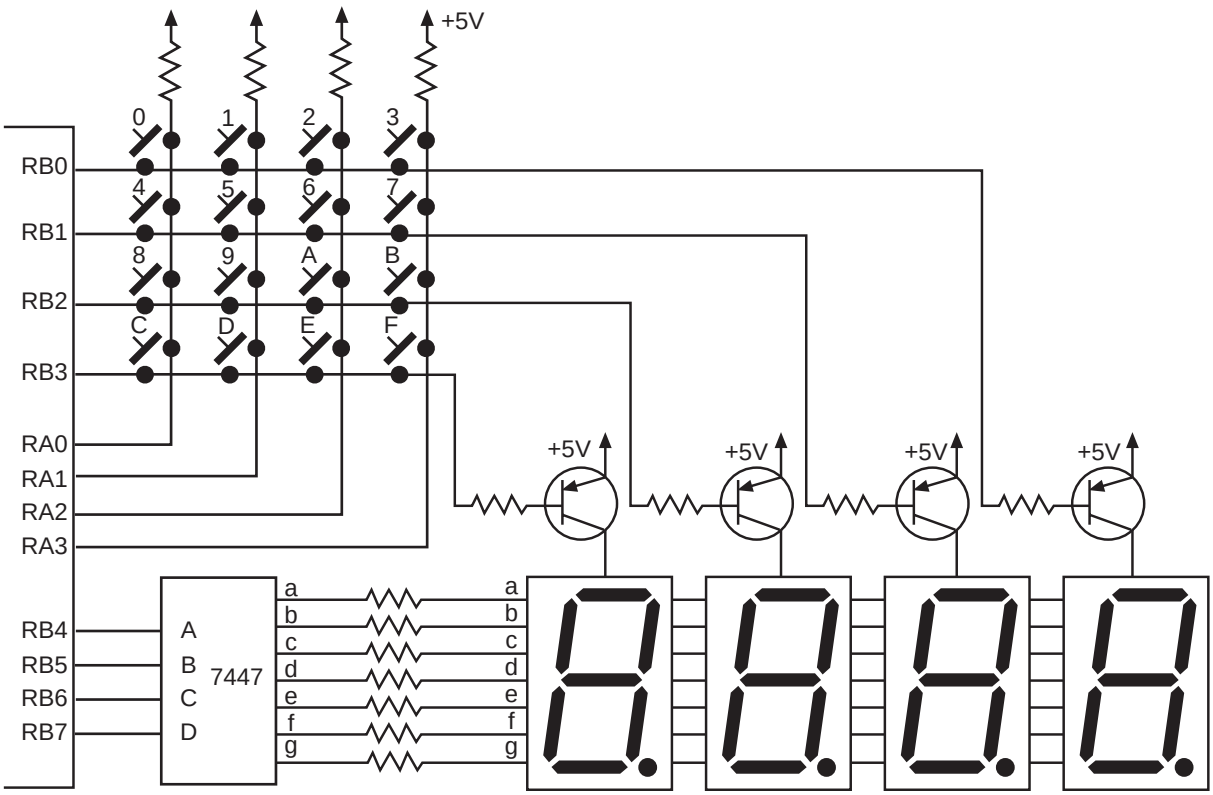


Figura 2.19. Multiplexaje de teclado y display al mismo tiempo

Aquí se resumen los anteriores ejemplos para crear uno sólo. El disparo del transistor es simultáneo con la salida del dato para activar cada segmento dentro del display, así que no es necesario preocuparse con apagar previamente los LED, antes de rotar el cero en las bases de los transistores.

Aquí dejamos al experimentador el desarrollo de los programas necesarios para leer el teclado y visualizar de manera “simultánea” la información en el display; puede tomar los programas anteriores como base. Con la práctica encontrará que puede mejorar estas rutinas y configuraciones, logrando minimizar el diseño y los costos, y maximizar el rendimiento.

Proyecto N° 4: Conexión de memorias seriales al PIC

Las técnicas para almacenar información en medios electrónicos se perfeccionan más cada día. A diario vemos ejemplos de su utilización en nuestros hogares y oficinas, por ejemplo, en receptores de televisión, reproductores de *compact disc*, sistemas de control remoto, impresoras, fotocopiadoras, teléfonos celulares, etc. Una de estas tecnologías corresponde a las llamadas memorias EEPROM seriales, las cuales tienen grandes ventajas si se comparan con otras posibilidades. Entre sus principales características se cuentan:

- Se pueden conectar fácilmente con microprocesadores o microcontroladores, inclusive algunos de ellos tienen pines dedicados para esta labor.
- Transferencia de datos de manera serial, lo que permite ahorrar pines del micro para dedicarlos a otras funciones.
- Ocupan la décima parte del espacio de las memorias que trabajan en paralelo, esto permite ahorrar dinero debido al menor tamaño del circuito impreso.
- El consumo de corriente es mucho menor que en las memorias que trabajan en paralelo, esto las hace ideales para sistemas portátiles que funcionan con baterías.

El objetivo de esta práctica es mostrar los aspectos más importantes de su tecnología y enseñar conceptos básicos para su utilización en circuitos reales, se basa en las memorias que tienen comunicación a 2 hilos empleando la interface I^2C, cuyas referencias más conocidas son 24LC01/02/04/16. La velocidad de transferencia de información para estos dispositivos es de 100 ó 400 kHz (aunque el límite lo impone el protocolo I^2C más no la tecnología del dispositivo). Como característica importante de este elemento se tiene la inmunidad al ruido, dado que este integrado tiene filtros en los pines de comunicación.

Memorias 24XX

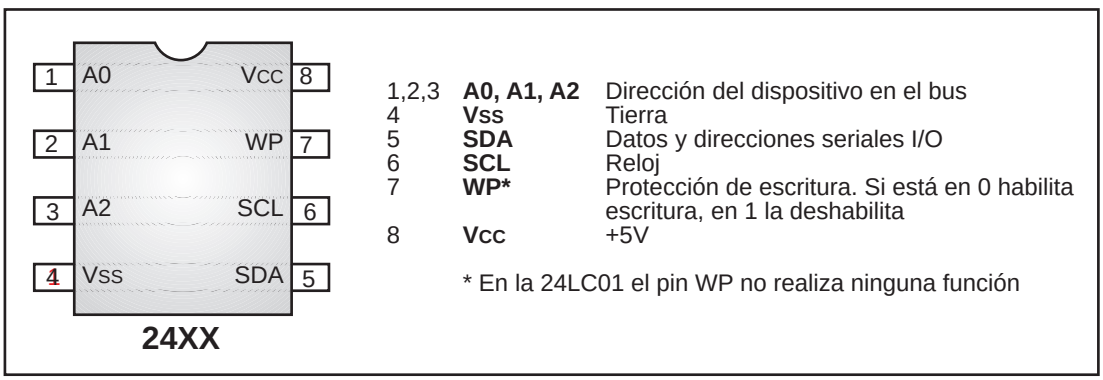


Figura 2.20. Configuración de pines de la memoria 24LCXX

Estas memorias utilizan el bus de 2 hilos para comunicarse con otros dispositivos. Dado que cumplen con el protocolo I^2C, tiene un pin llamado SCL que recibe los pulsos generados por el dispositivo maestro (o sea el microcontrolador) y otro llamado SDA que maneja el flujo de datos de forma bidireccional (entrada/salida). En la figura 2.20 se muestra el diagrama de pines correspondiente a estas memorias.

Este dispositivo no requiere de un pin habilitador o *chip select*, ya que en este esquema la transferencia de información solo se puede iniciar cuando el bus esté libre. En este caso, como cada dispositivo tiene su dirección determinada mediante los pines A0, A1 y A2; solamente responderá la memoria cuya dirección coincida con la dirección que va encabezando la trama de información. En la figura 2.21 se muestra la capacidad de almacenamiento de estos dispositivos y las posibilidades de direccionamiento que tienen.

Referencia	Capacidad en K bits	Bloques internos	A0	A1	A2	Dispositivos en el bus
24LC01B, 24C01	1	1	1 ó 0	1 ó 0	1 ó 0	8
24LC02B, 24C02	2	1	1 ó 0	1 ó 0	1 ó 0	8
24LC04B, 24C04	4	2	X	1 ó 0	1 ó 0	4
24LC08B	8	4	X	X	1 ó 0	2
24LC16B	16	8	X	X	X	1

Figura 2.21. Capacidad de memoria y direccionamiento de las memorias 24LCXX

Transferencia de la información. Cuando el microcontrolador desea entablar comunicación con la memoria, debe enviarle una serie de bits que llevan la siguiente información:

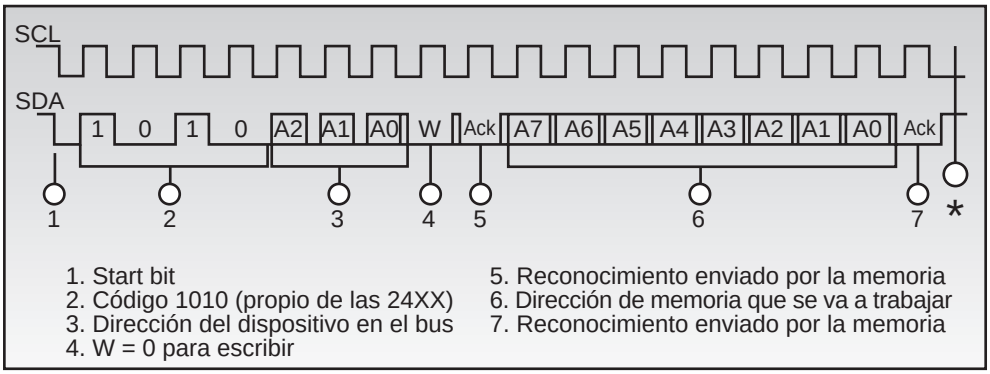
1. Se envia el bit de arranque o *start bit*
2. El código 1010 (propio de estas memorias)
3. La dirección del dispositivo (A2, A1, A0)
4. Un bit que indica que se desea escribir (¨0¨) en la memoria

Luego de esto,la memoria debe enviar un reconocimiento para informarle al microcontrolador que recibió la información. Dicho asentimiento, llamado *ACK*, consiste en poner el bus en un nivel bajo (lo hace la memoria). Después el microcontrolador debe enviar los bits que corresponden a la posición de memoria que se quiere leer o escribir; nuevamente la memoria envia un reconocimiento. El paso siguiente depende de la operación que se vaya a ejecutar.

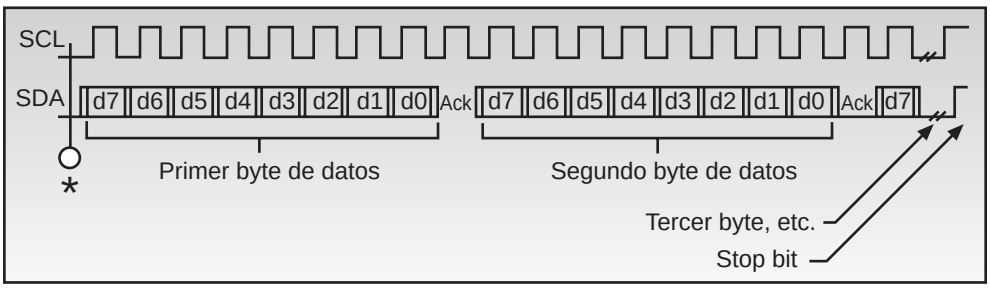
Si se trata de un proceso de escritura, el microcontrolador solo debe enviar el dato a ser almacenado y esperar el asentimiento por parte de la memoria para confirmar que llegó correctamente. Si se trata de una lectura, nuevamente se debe repetir los primeros cuatro pasos, solo que en lugar de un “0” que indica escritura, se debe enviar un “1” que indica lectura. Después se espera el asentimiento y acto seguido se puede leer el byte con el dato que estaba en la posición de memoria que se indicó anteriormente. Cuando se termina la operación, el microcontrolador debe enviar una señal de parada o *stop bit*. En la figura 2.22 se muestra el diagrama de tiempos correspondiente a todo el proceso descrito anteriormente.

Ejemplo de aplicación

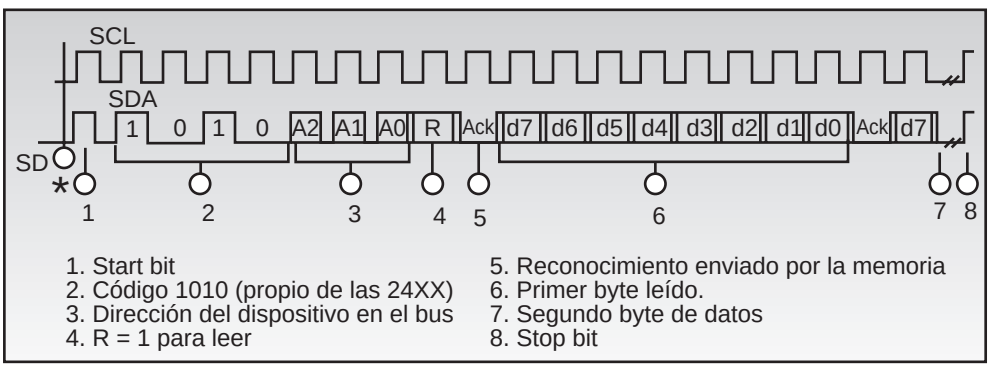
El ejercicio consiste en hacer un contador de 0 a 9 con un interruptor pulsador y un display de siete segmentos, similar al ejercicio de la figura 2.4. La diferencia radica en que el número que se muestra en el display se va a almacenar simultáneamente en una memoria 24LC01 (LC quiere decir que puede trabajar desde 2 voltios). Se va a utilizar un microcontrolador PIC16F84 (aunque se puede utilizar un 16C61 o 16C71).



A. Forma de direccionar la memoria 24XX



B. Escritura de byte



C. Lectura de byte

Figura 2.22. Diagrama de tiempos para la lectura y la escritura en una memoria 24LCXX

En la figura 2.23 se muestra el diagrama esquemático del circuito. En este caso los pines de dirección de la memoria se conectaron a tierra, al igual que el pin WP. La resistencia de 4.7 kohm conectada al pin SDA es necesaria dado que dicho pin tiene salida de colector abierto (*open collector*). El display se conecta al puerto A y el pulsador al pin RB0.

El programa que se escribe en el microcontrolador se muestra en la figura 2.24, su función principal es llevar el control del conteo decimal y almacenar en la memoria el mismo dato que se envía al display.

En el programa, la subrutina *WAIT* produce un retardo en milisegundos, la cantidad de milisegundos deseada debe escribirse en el registro *loops* antes de hacer el llamado correspondiente. Se utiliza principalmente para hacer un retardo de 10 mi-

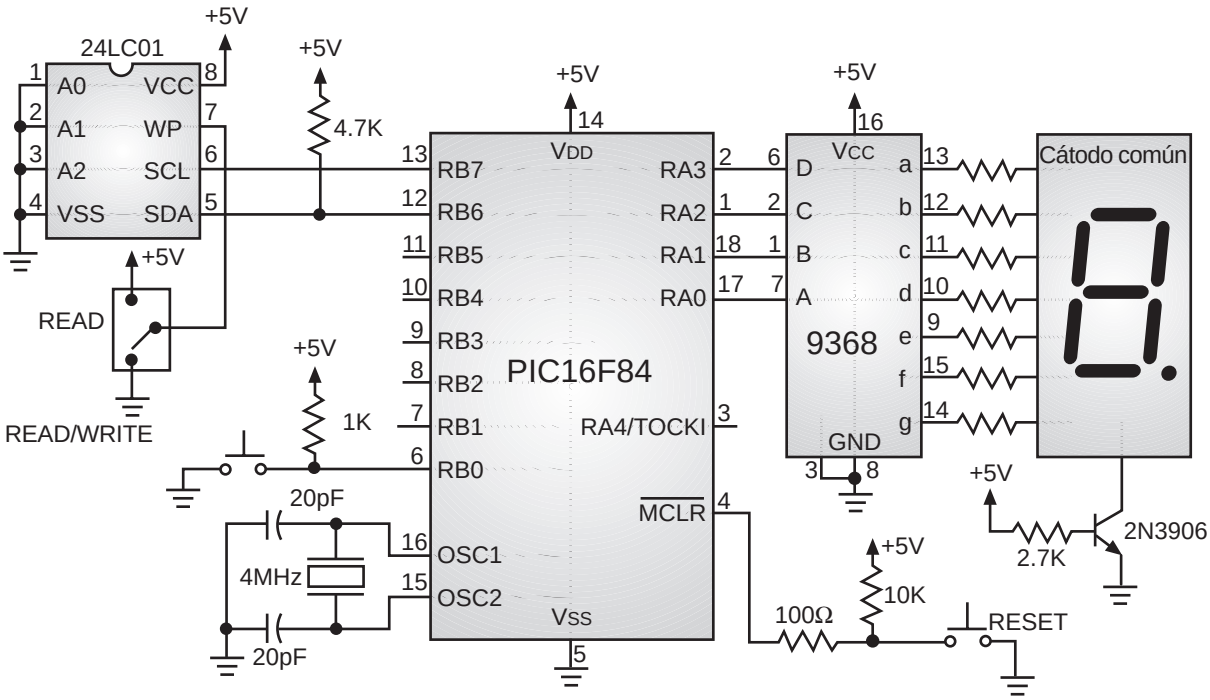


Figura 2.23. Diagrama esquemático del contador con PIC y memoria 24LC01

```
;Este programa realiza un contador decimal con un pulsador y un display de 7
;segmentos, el valor del conteo se guarda en la memoria serial 24LC01
;definición de bits
status  equ     3h      ;registro de estados
ptoa    equ     5h      ;
ptob    equ     6h      ;
addr    equ    0dh      ;posición de memoria que se lee o escribe
datao   equ    0eh      ;registro para escribir datos en la memoria
slave   equ    0fh      ;dirección del dispositivo en el bus I2C (1010xxx0)
txbuf   equ    10h      ;buffer de transmision
count   equ    11h      ;contador de bits
eeprom  equ    12h      ;buffer de bits
rxbuf   equ    13h      ;buffer de recepción
loops   equ    15h      ;se utilizan en retardos
loops2  equ    16h      ;
di      equ    7        ;bit de entrada desde eeprom
do      equ    6        ;bit de salida para eeprom
sdata   equ    6        ;linea de datos seriales (pin RB6)
sclk    equ    7        ;reloj serial (pin RB7)
conta   equ    17h      ;lleva el conteo de pulsaciones
conta2  equ    18h
trisa   equ    85h      ;registro de configuración del puerto A
trisb   equ    86h      ;registro de configuración del puerto B
z       equ    02h      ;bandera de cero del registro de estados
w       equ    00h      ;indica que el resultado se guarda en W
c       equ    00h      ;bandera de carry

        org     00h     ;Vector de reset
        goto    INICIO
        org     03h

WAIT                    ;subrutina de retardo en milisegundos
top2    movlw   .110    ;el numero de milisegundos llega
        movwf   loops2  ;cargado en el registro loops
top     nop
        nop
```

```
        nop
        nop
        nop
        nop
        decfsz  loops2  ;pregunta si termino 1 ms
        goto    top
        decfsz  loops   ;pregunta si termina el retardo
        goto    top2
        retlw   0

retardo movlw   .100    ;retardo de 100 milisegundos
        movwf   loops
        call    WAIT
        retlw   0

BSTART   ;Esta rutina genera el start bit para la comunicacion serial
         movlw   b'00111111'
         tris    ptob         ;programar datos y reloj como salidas
         bcf     ptob,sclk    ;linea de reloj en nivel bajo
         bsf     ptob,sdata   ;se asegura línea de datos en alto
         nop
         bsf     ptob,sclk    ;línea de reloj en alto
         nop
         nop                    ;ajuste de tiempo
         nop
         nop
         nop
         bcf     ptob,sdata   ;se baja la línea de datos
         nop                  ;mientras el reloj está alto
         nop
         nop                  ;ajuste de tiempo
         nop
         nop
         bcf     ptob,sclk    ;se baja la línea de reloj
         nop                  ;para terminar el pulso
         nop
         retlw   0

BSTOP    ;Esta rutina genera el stop bit para la comunicación serial
         movlw   b'00111111'  ;
         tris    ptob         ;programa reloj y datos como salidas
         bcf     ptob,sdata   ;asegura línea de datos en bajo
         nop
         nop
         nop
         bsf     ptob,sclk    ;línea de reloj en nivel alto
         nop
         nop
         nop
         bsf     ptob,sdata   ;la línea de datos pasa a nivel alto
         nop                  ;mientras el reloj está alto
         nop
         bcf     ptob,sclk    ;la línea de reloj baja nuevamente
         nop                  ;para completar el pulso
         nop
         nop
         retlw   0

BITOUT   ;Esta rutina toma el bit que se debe transmitir y lo saca al puerto
         movlw   b'00111111'  ;además genera el pulso de reloj
         tris    ptob         ;programa reloj y datos como salidas
         bsf     ptob,sdata   ;asume que el bit es alto
         btfss   eeprom,do    ;pregunta estado del bit a transmitir
         bcf     ptob,sdata   ;si el bit es bajo pone la salida en bajo
clkout   nop
         nop
         bsf     ptob,sclk    ;sube el nivel de la línea de reloj
         nop                  ;para formar el pulso
         nop
```

```
        nop
        nop
        bcf     ptob,sclk    ;termina pulso de reloj
        retlw   0

BITIN   ;Esta rutina lee un bit de la memoria y lo pone en un registro
        bsf     eeprom,di    ;asume que el bit es de nivel alto
        movlw   b'01111111'  ;programa pin de datos como entrada
        tris    ptob
        bsf     ptob,sclk    ;sube la linea del reloj
        nop                  ;
        nop
        nop
        nop
        nop
        nop
        nop
        nop                  ;
        btfss   ptob,sdata   ;pregunta por el estado del pin de datos
        bcf     eeprom,di    ;si es bajo lo pone en ese nivel
        bcf     ptob,sclk    ;si es alto lo deja como se asumió antes
        retlw   0            ;

TX      ;Esta rutina se encarga de transmitir un byte hacia la memoria
        movlw   .8
        movwf   count          ;el número de bits es 8
TXLP    bcf     eeprom,do      ;asume que el bit a enviar es bajo
        btfsc   txbuf,7        ;consulta el estado real del bit
        bsf     eeprom,do      ;si era alto lo deja con dicho nivel
        call    BITOUT         ;saca el bit por el puerto
        rlf     txbuf,1        ;rota el byte que se está transmitiendo
        decfsz  count          ;pregunta si ya pasaron los 8 bits
        goto    TXLP           ;si no ha terminado sigue transmitiendo
        call    BITIN          ;espera el reconocimiento enviado por la
        retlw   0              ;memoria (ACK)

RX      ;Esta rutina recibe un byte y lo entrega en el registro rxbuf
        clrf    rxbuf          ;borra el buffer de entrada
        movlw   .8             ;indica que recibe 8 bits
        movwf   count
        bcf     status,0       ;Borra el carry
RXLP    rlf     rxbuf, F       ;rota a la izquierda
        call    BITIN          ;lee un bit
        btfsc   eeprom,di
        bsf     rxbuf,0        ;si es necesario pone el bit en uno
        decfsz  count          ;pregunta si completo 8 bits
        goto    RXLP           ;sino, recibe otro bit
        bsf     eeprom,do      ;envia el ACK de asentimiento
        call    BITOUT         ;para terminar
        retlw   0

LEER    ;Esta rutina recibe la dirección que se
        ;desea LEER y devuelve el dato que tiene grabado
        call    BSTART      ;genera el start bit
        nop                 ;
        nop                 ;
        bcf     slave,0     ;selecciona la memoria
        movf    slave,w     ;y selecciona modo de escritura
        movwf   txbuf       ;
        call    TX          ;envía esos datos a la memoria
        movf    addr,w      ;
        movwf   txbuf       ;envía la posicion de memoria a ser leida
        call    TX          ;
        nop                 ;ahora se selecciona nuevamente la memoria
        nop                 ;y se le indica modo de lectura
        call    BSTART      ; genera start bit
        nop
        nop
        bsf     slave,0     ;indica que se va a LEER
```

```
          movf    slave,w     ;selecciona el dispositivo
          movwf   txbuf       ;
          call    TX          ;envia esa información a la memoria
          nop
          call    RX          ;la memoria entrega el byte de esa dirección
          bsf     eeprom,do   ;envia el ACK de reconocimiento
          call    BITOUT
          call    BSTOP       ;se envia el stop bit para finalizar comunicación
          retlw   0

ESCRIB    ;Esta rutina escribe un dato en la posición
          ;de memoria que se le indique en el registro addr
          call    BSTART      ;genera el start bit
          nop                 ;
          nop                 ;
          bcf     slave,0     ;selecciona la memoria
          movf    slave,w     ;y selecciona modo de escritura
          movwf   txbuf       ;
          call    TX          ;envía esos datos a la memoria
          movf    addr,w      ;
          movwf   txbuf       ;envía la posición de memoria a ser grabada
          call    TX          ;ahora se selecciona nuevamente la memoria
          nop                 ;y se le indica modo de lectura
          nop
          movf    datao,w     ;toma el dato que va a ser grabado
          movwf   txbuf       ;y lo envía
          call    TX
          call    BSTOP
          movlw   .10         ;retardo de 10 ms al escribir
          movwf   loops       ;cada dato
          call    WAIT
          retlw   0

INICIO    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f
          movwf     trisa     ;se programa el puerto A como salidas
          movlw     07fh      ;se carga el registro W con 00
          movwf     trisb     ;se programa el puerto B como entradas
          bcf       status,5  ;se ubica en el primer banco de memoria RAM
          movlw  b'10100000'   ;La dirección A0, A1, y A2 de la memoria
          movwf  slave         ;en el bus I2C es 000
          clrf      addr      ;cuando se enciende el sistema se verifica que
          call      LEER      ;el dato guardado en memoria esté entre 0 y 9
          movlw     0ah       ;la prueba se hace porque la primera vez que
          subwf     rxbuf,w   ;se encienda el sistema se puede tener un
          btfss     status,c  ;número fuera del rango
          goto      ciclo     ;para las ocasiones posteriores no importa
ini2      clrf      conta     ;inicia contador en cero
          clrf      datao
          call      ESCRIB    ;inicia dato de memoria en 0
ciclo     call      LEER      ;LEER memoria, devuelve dato en W
          movf      rxbuf,w   ;pasa el valor de W al puerto A (display)
          movwf     conta
          movwf     ptoa
          call      retardo   ;retardo esperando que suelten la tecla
pulsa     btfsc     ptob,0    ;pregunta si el pulsador está oprimido
          goto      pulsa     ;si no lo está continúa revisándolo
          call      retardo   ;si está oprimido retarda 100 milisegundos
          btfsc     ptob,0    ;para comprobar
          goto      pulsa     ;si no lo está vuelve a revisar
          incf      conta     ;si lo confirma incrementa el contador
          movf      conta,w   ;carga el registro W con el valor del conteo
          movwf     datao     ;el dato del conteo lo guarda en memoria
          call      ESCRIB    ;para recuperarlo en caso de un apagón
          movf      conta,w
          xorlw     0ah       ;hace operación xor para ver si es igual a 0ah
          btfss     status,z  ;prueba si el contador llegó a 0ah (diez)
          goto      ciclo     ;si no es igual se incrementa normalmente
          goto      ini2      ;
```

```
          end                 ;
;=========================================================================
;         Fusibles de programación
;         Osc                 XT
;         Watchdog            OFF
;         Code protect        OFF
;         Power-Up-Timer      ON
;         Micro.              PIC16F84
;=========================================================================
```

Figura 2.24. Programa del contador con PIC y memoria 24LC01

lisegundos luego de escribir un dato en la memoria. Se debe tener en cuenta que los retardos están calculados para un oscilador de 4 MHz en el microcontrolador.

La subrutina *BSTART* genera el bit de inicio de la comunicación, con la temporización y estado de los pines adecuados. Por su parte, la subrutina *BSTOP* hace lo mismo con el bit de parada o de fin de la comunicación. La subrutina *BITOUT* toma el bit de dato que se debe transmitir y lo envia hacia la memoria, se encarga de programar el pin del microcontrolador como salida y de generar el pulso de reloj necesario para la sincronización. La rutina *BITIN* hace su parte cuando se está leyendo un bit enviado por la memoria, genera el pulso de reloj y pasa el bit leído al registro o buffer de entrada. Las rutinas *TX* y *RX* se encargan de transmitir y recibir un byte completo de datos, cada una hace 8 llamados seguidos a las rutinas *BITOUT* y *BITIN* respectivamente.

La rutina *LEER* recibe en el registro *addr* la posición de memoria que se debe leer y genera todas las señales necesarias (incluyendo el start y el stop bit) para obtener el dato que en ella se encuentra grabado, al final devuelve el dato que recibió de la memoria en el registro *rxbuf*. La rutina *ESCRIB* toma el dato contenido en el registro *datao* y lo escribe en la posición de la memoria que está direccionada en el registro *addr*.

Cada vez que se enciende el sistema el microcontrolador lee el dato que se encuentra en la primera posición de memoria y lo pasa al display. Cuando el pulsador sea oprimido se debe incrementar dicho dato y se actualiza el display al tiempo que se vuelve a almacenar ese número en la memoria. Un caso especial ocurre cuando se enciende el sistema por primera vez, como el dato que se encuentra grabado en la memoria es desconocido y podría ser superior a 9, este se debe probar y si se encuentra que es mayor, se borra y se empieza el conteo en 0.

Las rutinas que permiten leer y escribir en la memoria 24LC01se pueden utilizar como parte de cualquier programa sin que se tengan contratiempos, sólo se debe tener en cuenta que las temporizaciones están calculadas para un oscilador de 4 MHz. Con las rutinas LEER y ESCRIB se tiene una velocidad de transferencia de información de aproximadamente 60 kHz.

Una prueba que es muy interesante consiste en cambiar de posición el interruptor que selecciona la protección de escritura, cuando está en la posición READ/WRITE se puede incrementar el contador normalmente, cuando se encuentra en la posición READ el contador no se incrementa debido a que la memoria está protegida contra escritura.

Proyecto N° 5: Manejo de un módulo LCD

Cuando se trabaja en diseño de circuitos electrónicos es frecuente encontrarse con la necesidad de visualizar un mensaje, que tiene que ver con el estado de la máquina a controlar, con instrucciones para el operario, o si es un instrumento de medida, mostrar el valor registrado. En la mayoría de los casos, recurrimos a los displays de siete segmentos, pero estos además de no mostrar caracteres alfanuméricos ni ASCII, tienen un elevado consumo de corriente y son un poco dispendiosos de manejar, cuando se requiere hacer multiplexaje.

Los módulos de cristal líquido o LCD, solucionan estos inconvenientes y presentan algunas ventajas, como un menor consumo de corriente, no hay que preocuparse por hacer multiplexaje, no hay que hacer tablas especiales con los caracteres que se desea mostrar, se pueden conectar fácilmente con microprocesadores o microcontroladores y además, los proyectos adquieren una óptima presentación y funcionalidad. En principio, vamos a conocer las características más importantes de los módulos, luego se muestra la forma de conectarlos con el microcontrolador y se hacen programas simples para escribir mensajes en la pantalla.

Módulos de cristal líquido o LCD

Antes de mostrar la forma de conectar estos módulos con el microcontrolador, haremos un pequeño recuento de las principales características que ellos tienen, las cuales nos servirán para entender mejor los programas y los diagramas que se muestran más adelante:

- Los módulos LCD se encuentran en diferentes presentaciones, por ejemplo (2 líneas por 16 caracteres), 2x20, 4x20, 4x40, etc. La forma de utilizarlos y sus interfaces son similares, por eso, los conceptos vistos aquí se pueden emplear en cualquiera de ellos. En nuestro caso, trabajaremos con un display de 2x16, ya que es de bajo costo, se consigue fácilmente en el comercio y tiene un tamaño suficiente para la mayoría de las aplicaciones.

- La figura 2.25 muestra dos tipos de configuración de pines que se encuentran comúnmente, aunque cambian su ubicación, estos conservan las mismas funciones. Algunos módulos LCD tienen luz posterior o “backlight”, para mejorar su visualiza-

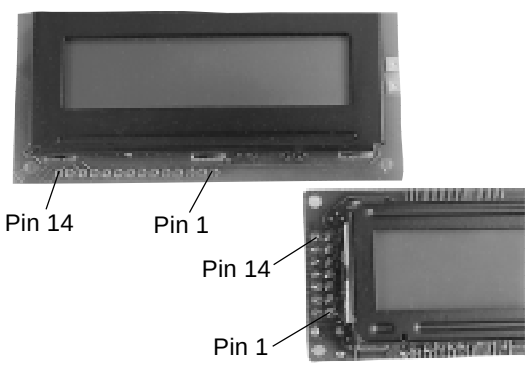


Figura 2.25. Configuración de pines de los módulos LCD

ción, ésta se maneja a través de dos pines que normalmente se conectan a +5V y a tierra. Para evitar que se presenten altas temperaturas, debido a la luz posterior, estos pines se deben manejar de manera pulsante (encendiendo y apagando), con una frecuencia de aproximadamente 60 Hz. Otra opción mucho más sencilla es utilizar una resistencia de 10 ohmios (a 1/2W) para alimentar el positivo del *backlight*.

• Los pines de conexión de estos módulos incluyen un bus de datos de 8 bits, un pin de habilitación (E), un pin de selección, que indica que el dato es una instrucción o un caracter del mensaje (RS) y un pin que indica si se va a escribir o leer en el módulo LCD (R/W). La figura 2.26 describe la función de cada uno de ellos.

Terminal	Símbolo	Nombre y Función
1	Vss	Tierra, 0V
2	Vdd	Alimentación +5V
3	Vo	Ajuste de Voltaje de contraste
4	$\overline{RS}$	Selección Dato/Control
5	R/$\overline{W}$	Lectura/escritura en LCD
6	E	Habilitación
7	D0	D0 Bit menos significativo
8	D1	D1
9	D2	D2
10	D3	D3
11	D4	D4
12	D5	D5
13	D6	D6
14	D7	D7 Bit más significativo

Figura 2.26. Función de los pines del módulo LCD

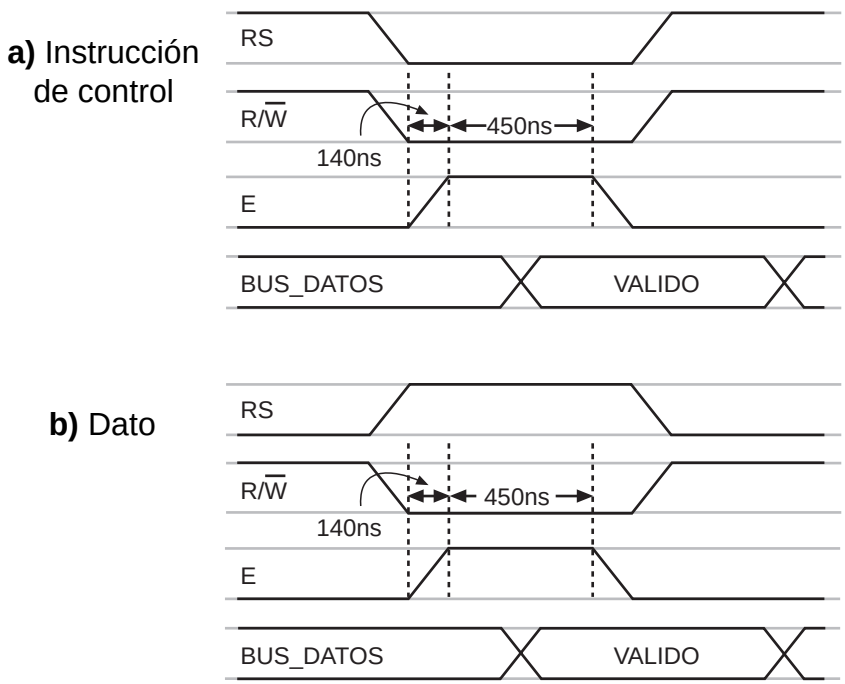


Figura 2.27. Diagrama de tiempo del módulo LCD

• Según la operación que se desee realizar sobre el módulo de cristal líquido, los pines de control E, RS y R/W deben tener un estado determinado. Además, debe tener en el bus de datos un código que indique un caracter para mostrar en la pantalla o una instrucción de control. En la figura 2.27 se muestra el diagrama de tiempos que se debe cumplir para manejar el módulo.

• El módulo LCD responde a un conjunto especial de instrucciones, estas deben ser enviadas por el microcontrolador o sistema de control al display, según la operación que se requiera. Estas instrucciones se emplean en los ejemplos que realizaremos más adelante, en ellos se explica la forma de utilizarlas. En la figura 2.28 se muestran las instrucciones del módulo.

• La interface entre el microcontrolador y el display de cristal líquido se puede hacer con el bus de datos trabajando a 4 u 8 bits. Las señales de control trabajan de la misma forma en cualquiera de los dos casos, la diferencia se establece en el momento de iniciar el sistema, ya que existe una instrucción que permite establecer dicha configuración. Estas conexiones se explican más adelante en forma detallada.

Control y dato / INSTRUCCIONES	Señal de control		DATO/DIRECCION								DESCRIPCION
	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	
Borrar pantalla	0	0	0	0	0	0	0	0	0	1	Limpia todo display y retorna el cursor a la posición de inicio (dirección 0)
Cursor a casa	0	0	0	0	0	0	0	0	1	*	Retorna el cursor a la posición de inicio (dirección 0). También retorna el display, desplazándolo a la posición original. Los contenidos de la RAM DD permanecen sin cambio.
Seleccionar modo	0	0	0	0	0	0	0	1	1/D	0	Configura la dirección de movimiento del cursor y si se desplaza o no el display. Esta operación es realizada durante operaciones de lectura y escritura.
Encender/apagar pantalla	0	0	0	0	0	0	1	D	C	B	Configura el estado ON/OFF de todo el display (D), el cursor (C) y el parpadeo del caracter en la posición del cursor.
Desplazar Cursor/Pantalla	0	0	0	0	0	1	S/C	R/L	*	*	Mueve el cursor y desplaza el display sin cambiar los contenidos de la RAM DD.
Acticar función	0	0	0	0	1	D/L	N	F	*	*	Configura el tamaño de la interface (DL), el número de líneas del display (N) y la fuente del caracter (F). N=0, 1 línea N=1, 2 líneas
CG RAM	0	0	0	1	Dirección generador de RAM						Ajusta la dirección del generador de caracteres. El dato CG RAM es enviado y recibido después de este ajuste.
DD RAM	0	0	1	Direcciónes de datos RAM							Ajusta la dirección de la RAM DD. La dirección es enviada y recibida después de este ajuste.
Bandera de ocupado	0	0	BF	AC							Lectura de la bandera Busy Flag, indicando que operaciones internas son realizadas, y lectura de los contenidos del contador de direcciones.
Escritura CG RAM/DD RAM	1	0	Escritura de Dato								Escribe datos en la RAM DD o en la RAM CG.
LECTURA CG RAM/DD RAM	1	1	Lectura de Dato								Lectura de datos desde la RAM DD o la RAM CG

Figura 2.28. Conjunto de instrucciones de los módulos LCD

Significado de las abreviaturas

I/D = 1 Incrementa
= 0 Decrementa
S = 1 Desplaza el mensaje en la pantalla
= 0 Mensaje fijo en la pantalla
D = 1 Encender (activar) la pantalla
= 0 Apagar la pantalla (desactivar)

C = 1 Activar cursor
= 0 Desactivar cursor
B = 1 Parpadea caracter señalado por el cursor
= 0 No parpadea el caracter
S/C = 1 Desplaza pantalla
= 0 Mueve cursor

RL = 1 Desplazamiento a la derecha
= 0 Desplazamiento a izquierda
DL = 1 Datos de ocho bits
= 0 Datos de cuatro bits
BF = 1 Durante operación interna del módulo
= 0 Finalizada la operación interna

• Los caracteres que se envían al display se almacenan en la memoria RAM del módulo. Existen posiciones de memoria RAM, cuyos datos son visibles en la pantalla y otras que no son visibles, estas últimas se pueden utilizar para guardar caracteres que luego se desplazan hacia la parte visible. En la figura 2.29 se muestran las direcciones de memoria visibles y no visibles, que conforman las dos líneas de caracteres del módulo.

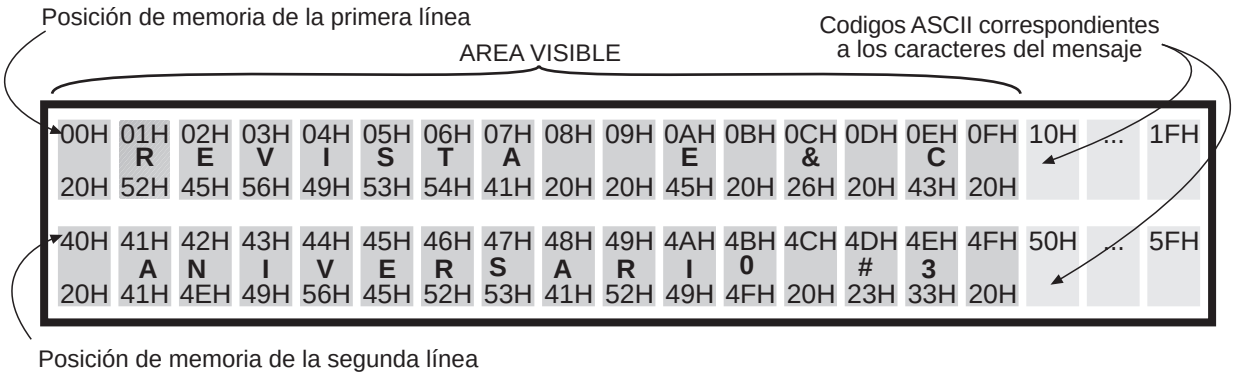


Figura 2.29. Mapa de memoria del módulo LCD

• Es importante anotar que sólo se pueden mostrar caracteres ASCII de 7 bits, por lo tanto algunos caracteres especiales no se pueden ver (se debe tener a la mano una tabla de los caracteres ASCII para conocer los datos que son prohibidos) . Por otra parte, se tiene la opción de crear caracteres especiales (creados por el programador), y almacenarlos en la memoria RAM que posee el módulo.

Interface con microcontrolador a 8 bits

El proyecto consiste en conectar el módulo de cristal líquido a un microcontrolador PIC16F84, utilizando el bus de datos a 8 bits. En este caso se emplea el PIC16F84, aunque se puede implementar con otros microcontroladores que sean compatibles. En la figura 2.30 se muestra el diagrama de conexiones para este caso.

Para estos ejercicios en particular, sólo nos interesa escribir datos en la pantalla (no hacer lectura); por lo tanto el pin de selección de lectura/escritura (R/W) en el display, se conecta a tierra. El puerto B del microcontrolador se utiliza como bus de datos, y el puerto A se encarga de generar las señales de control.

En el oscilador del PIC16F84 se emplea un cristal de 4 MHz, por lo tanto tenemos ciclos de instrucción de un microsegundo. Para el módulo LCD, se emplea un potenciómetro de 5Kohm, conectado entre +5V y tierra, para controlar el contraste de la pantalla.

En la figura 2.31 se muestra el listado del programa. Para este caso, el ejemplo consiste en hacer circular un mensaje en la línea superior de la pantalla. La explicación de los pasos contenidos en él es la siguiente:

1. Se programan los puertos según las conexiones que se tienen en el circuito.
2. Se debe inicializar el módulo LCD. El primer dato que se envía (30H) le dice al módulo que la comunicación es a 8 bits y que se empleará solo una línea de carac-

teres. El dato se puede deducir con la lista de las instrucciones que se muestra en la figura 3. La rutina llamada CONTROL, se encarga de generar las señales y los tiempos necesarios para que exista una correcta comunicación.

3. El segundo dato (07H), le dice al módulo que el mensaje se va a desplazar en la pantalla.
4. El tercer dato (0CH), hace que se encienda el display.
5. El siguiente paso es entrar en un ciclo que hace una lectura de la tabla donde se encuentra el mensaje y lo lleva a la memoria del módulo LCD. Cuando se termina de enviar todos los caracteres, se inicia el ciclo nuevamente.

Las rutinas CONTROL y DATO emplean las mismas instrucciones. La única diferencia es que cada una le da el nivel lógico adecuado al pin RS, que indica si el dato enviado es un caracter del mensaje (un dato) o una instrucción de control.

En el modo que desplaza el mensaje en la pantalla se tiene un tiempo de espera un poco largo antes de que aparezcan los caracteres en la pantalla, esto se debe a que primero se llena o se carga la memoria de datos de la parte no visible.

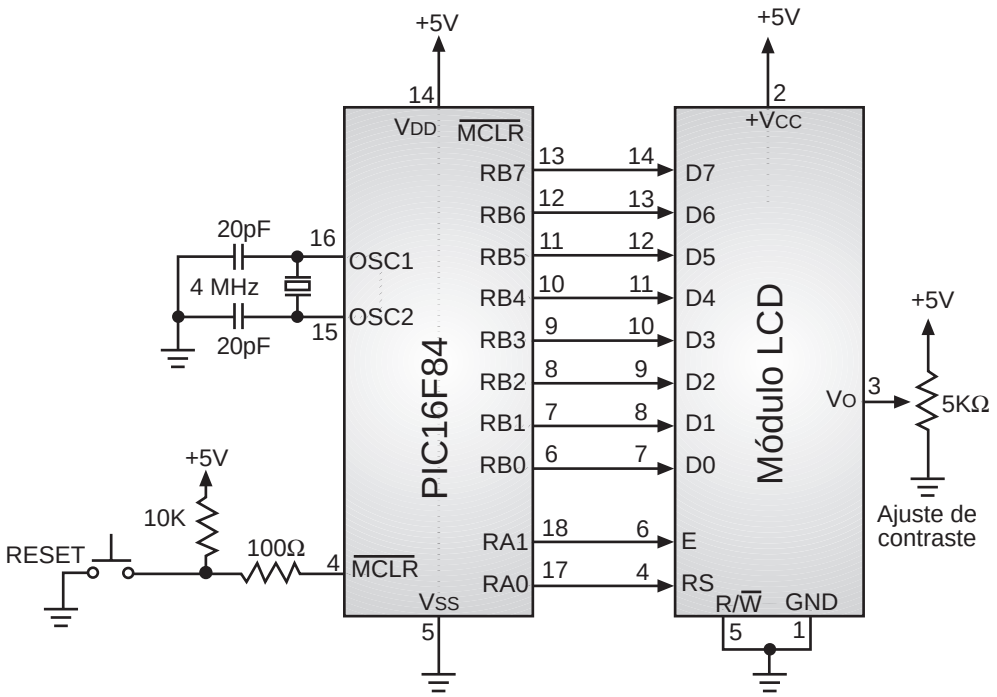


Figura 2.30. Diagrama esquemático de la conexión a 8 bits entre el microcontrolador y el módulo LCD

```
;Este programa hace que un mensaje circule en la pantalla
;de un modulo lcd        ; p=16f84, osc= xt, wdt = off
indf    equ     0h      ;para direccionamiento indirecto
tmro    equ     1h      ;contador de tiempo real
pc      equ     2h      ;contador de programa
status  equ     3h      ;registro de estados y bits de control
fsr     equ     4h      ;seleccción de bancos de memoria y registros
ptoa    equ     5h      ;puertos
ptob    equ     6h
r0c     equ     0ch     ;
r0d     equ     0dh     ;
```

```
r13     equ     13h     ;
z       equ     2h      ;bandera de cero
c       equ     0h      ;bandera de carry
w       equ     0h      ;para almacenar en w
r       equ     1h      ;para almacenar en el mismo registro
e       equ     1h
rs      equ     0h

        org     00       ;vector de reset
        goto    inicio
        org     05h

retardo movlw   0ffh
        movwf   r13
decre   decfsz  r13,r
        goto    decre
        retlw   0

control bcf     ptoa,rs         ;esta rutina genera las señales de control
        goto    dato2           ;y entrega el dato correspondiente al módulo
dato    bsf     ptoa,rs         ;utiliza interface a 8 bits
dato2   bsf     ptoa,e
        movwf   ptob
        call    retardo
        bcf     ptoa,e
        call    retardo
        retlw   0

tabla2  addwf   pc,r             ;mensaje a ser rotado
        retlw   "C"
        retlw   "U"
        retlw   "R"
        retlw   "S"
        retlw   "O"
        retlw   " "
        retlw   "D"
        retlw   "E"
        retlw   " "
        retlw   "M"
        retlw   "I"
        retlw   "C"
        retlw   "R"
        retlw   "O"
        retlw   "C"
        retlw   "O"
        retlw   "N"
        retlw   "T"
        retlw   "R"
        retlw   "O"
        retlw   "L"
        retlw   "A"
        retlw   "D"
        retlw   "O"
        retlw   "R"
        retlw   "E"
        retlw   "S"
        retlw   " "
        retlw   "P"
        retlw   "I"
        retlw   "C"
        retlw   " "
        retlw   "C"
        retlw   "E"
        retlw   "K"
        retlw   "I"
        retlw   "T"
        retlw   " "
        retlw   " "
        retlw   " "
```

Nota: Las comillas que posee cada letra le indican al ensamblador que el dato requerido es el valor ASCII del caracter

```
        retlw   " "
        retlw   " "
        retlw   0

inicio  movlw   0fch            ;programación de puertos
        tris    ptoa            ;segun el circuito
        movlw   00h             ;
        tris    ptob            ;
begin   movlw   30h              ;inicia display a 8 bits y 1 línea
        call    control
        movlw   07h              ;selecciona el modo de desplazamiento
        call    control
        movlw   0ch              ;activa display
        call    control
muestra movlw   0               ;inicia el envio de caracteres
        movwf   r0c             ;al módulo
ciclo   movf    r0c,w           ;hace barrido de la tabla
        call    tabla2
        call    dato
        movlw   09fh            ;retardo entre caracteres
        movwf   r0d
reta1   call    retardo
        call    retardo
        decfsz  r0d,r
        goto    reta1
        incf    r0c,r           ;sigue con el próximo caracter del mensaje
        movlw   28h
        xorwf   r0c,w           ;pregunta si terminó el mensaje para volver
        btfss   status,z        ;a empezar
        goto    ciclo
        goto    muestra
        end
```

Figura 2.31. Programa para la conexión a 8 bits

Interface con microcontrolador a 4 bits

La conexión entre el PIC y el módulo LCD se hará empleando un bus de datos de 4 bits, en este caso, se utilizarán los 4 pines de mayor peso del puerto B (RB4-RB7) del microcontrolador. Las señales de control (RS y E), se generarán con los dos pines de menor peso del puerto B (RB0 y RB1). Con esto, se libera todo el puerto A y los pines RB2 y RB3 del microcontrolador, los cuales se podrían emplear en otras funciones, figura 2.32.

El software que se implementará en el microcontrolador, se encarga de mostrar un mensaje en el display. Dicho mensaje ocupa las dos líneas de la pantalla y permanece fijo, a diferencia del ejemplo anterior, en el cual el mensaje se desplazaba.

En la figura 2.33 se muestra el listado del programa que realiza la tarea descrita anteriormente, este sufre unas modificaciones respecto al ejemplo anterior. Los pasos más importantes son:

1. Se programan los puertos según el circuito.
2. Se debe inicializar el módulo LCD. El primer dato que se envía (02H) le dice al módulo que la comunicación se va a realizar a 4 bits.
3. El segundo dato (28H) ratifica que la comunicación es a 4 bits y que se emplearán las dos líneas de caracteres del display. El dato se puede deducir con la lista de las instrucciones que se muestra en la figura 3. La rutina llamada CONTROL, se

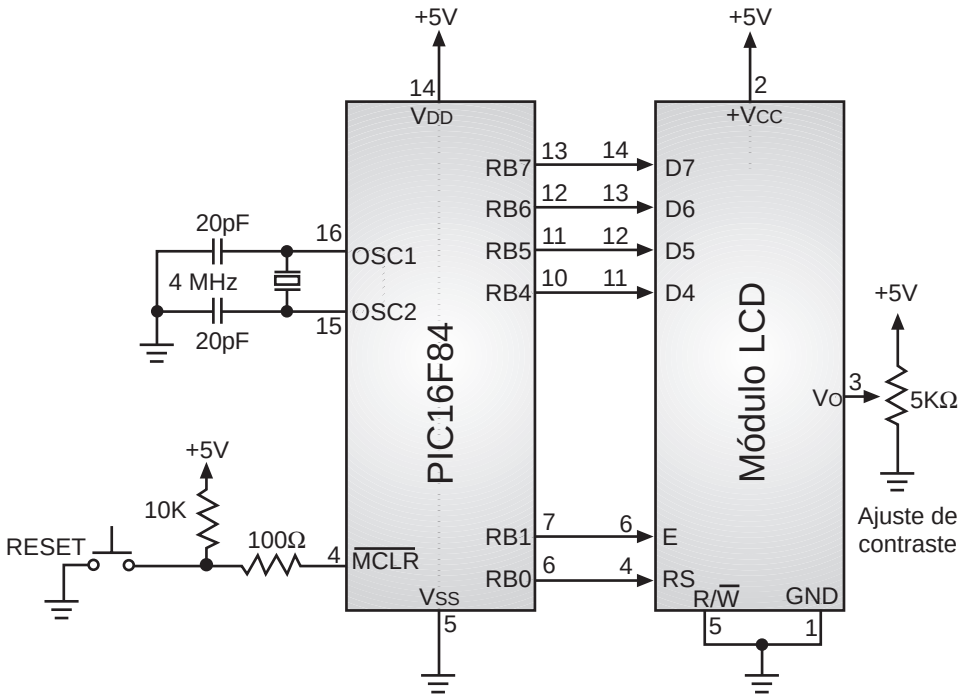


Figura 2.32. Diagrama esquemático de la conexión a 4 bits entre el microcontrolador y el módulo LCD

```
;este programa hace que un mensaje se repita indefinidamente
;en un modulo lcd de 2 lineas con 16 caracteres
indf    equ     0h      ;para direccionamiento indirecto
tmro    equ     1h      ;contador de tiempo real
pc      equ     2h      ;contador de programa
status  equ     3h      ;registro de estados y bits de control
fsr     equ     4h      ;selecccion de bancos de memoria y registros
ptoa    equ     5h      ;puertos
ptob    equ     6h
r0c     equ     0ch     ;
r0d     equ     0dh     ;
r0e     equ     0eh     ;
r13     equ     13h     ;
z       equ     2h      ;bandera de cero
c       equ     0h      ;bandera de carry
w       equ     0h      ;para almacenar en w
r       equ     1h      ;para almacenar en el mismo registro
e       equ     1h              ;
rs      equ     0h              ;

        org     00             ;vector de reset
        goto    inicio          ;va a iniciar programa principal
        org     05h

retardo movlw   0ffh
        movwf   r13
decre   decfsz  r13,r
        goto    decre
        retlw   0

limpia  clrf    r0c
limpi   movlw   " "
        call    dato
        incf    r0c,r
        movlw   50h
        xorwf   r0c,w
```

```
        btfss   status,z
        goto    limpi
        retlw   0

control bcf     ptob,rs         ;esta rutina genera las señales de control
        goto    dato2           ;para escribir en el modulo lcd y
dato    bsf     ptob,rs         ;entrega el dato a ser mostrado en la pantalla
dato2   bsf     ptob,e          ;utiliza la interface a 4 bits
        movwf   r0e
        movlw   0fh
        andwf   ptob,r
        movf    r0e,w
        andlw   0f0h
        iorwf   ptob,r
        call    retardo
        bcf     ptob,e
        call    retardo
        bsf     ptob,e
        movlw   0fh
        andwf   ptob,r
        swapf   r0e,w
        andlw   0f0h
        iorwf   ptob,r
        call    retardo
        bcf     ptob,e
        call    retardo
        retlw   0

tabla   addwf   pc,r            ;mensaje que se muestra
        retlw   " "
        retlw   " "
        retlw   "R"
        retlw   "E"
        retlw   "V"
        retlw   "I"
        retlw   "S"
        retlw   "T"
        retlw   "A"
        retlw   " "
        retlw   "E"
        retlw   "&"
        retlw   "C"
        retlw   " "
        retlw   " "
        retlw   " "
        retlw   " "           ;mensaje de la segunda línea
        retlw   "C"
        retlw   "E"
        retlw   "K"
        retlw   "I"
        retlw   "T"
        retlw   " "
        retlw   "-"
        retlw   " "
        retlw   "P"
        retlw   "E"
        retlw   "R"
        retlw   "E"
        retlw   "I"
        retlw   "R"
        retlw   "A"
        retlw   " "
        retlw   0

inicio  movlw   0ffh            ;programación de puertos
        tris    ptoa            ;segun el circuito
        movlw   0ch             ;
        tris    ptob            ;
begin   movlw   02h               ;inicia display a 4 bits
```

```
         call    control
         movlw   28h               ;display a 4 bits y 2 lineas
         call    control
         movlw   0ch               ;activa display
         call    control
         movlw   06h               ;hace que el mensaje permanezca fijo
         call    control

blank    call    limpia          ;borra display
muestra  clrf    r0c             ;inicia contador de caracteres
ciclo    movf    r0c,w           ;hace barrido de la tabla
         call    tabla
         call    dato
         movlw   0ffh            ;retardo entre caracteres
         movwf   r0d
reta1    call    retardo
         decfsz  r0d,r
         goto    reta1
         incf    r0c,r           ;sigue con la tabla
         movlw   11h
         subwf   r0c,w           ;pregunta si está mostrando el mensaje de la
         btfss   status,c        ;segunda linea
         goto    ciclo           ;
         movlw   11h             ;pregunta si es la primera vez que entra
         xorwf   r0c,w           ;a la segunda linea para ir a iniciar
         btfss   status,z        ;el puntero de la ram del modulo lcd
         goto    line2
linea2   movlw   0c0h            ;ubica puntero de la ram del módulo lcd
         call    control         ;en la segunda línea
line2    movlw   21h             ;pregunta si terminó la segunda línea
         xorwf   r0c,w           ;para ir a iniciar de nuevo el mensaje o
         btfss   status,z        ;para continuar en la segunda parte del
mensaje
         goto    ciclo           ;
         movlw   080h            ;ubica puntero de ram en la primera fila
         call    control
         goto    blank           ;va a reiniciar el mensaje en blank
         end

;      ******  pic16F84   ********
;      ******  wdt = off  *******
;      ******  osc = xt   *******
;      ******   cp = off   ******
```

Figura 2.33. Programa para la conexión a 4 bits

encarga de generar las señales y los tiempos necesarios para que exista una correcta comunicación. Debe notarse que esta rutina ha cambiado respecto al primer ejemplo.

4. El tercer dato (0CH), hace que se encienda el display.
5. El cuarto dato (06H), le indica al módulo LCD que el mensaje debe permanecer fijo en la pantalla. Recuerde que en el ejemplo anterior, para desplazarlo se usó el dato 07H.
6. El siguiente paso, es entrar en un ciclo que hace un borrado de la pantalla (con la subrutina BLANK) y luego una lectura de la tabla donde se encuentra el mensaje, para llevarlo a la memoria del módulo LCD. Esta lectura incluye una verificación del momento en que se llena la primera línea de caracteres, con el fin de ubicar el puntero de la RAM del módulo LCD, en la segunda línea de caracteres. Para esto se envía el comando C0H al display (este dato se puede deducir observando la lista de instrucciones de la figura 3).

Cuando se está escribiendo en la segunda línea de caracteres, se verifica en que momento se llenó el área visible del display (se compara el puntero de la tabla con 21H), en ese momento el programa vuelve a iniciar el ciclo de borrar el display y escribir el mensaje en la pantalla.

Las rutinas CONTROL y DATO han cambiado de manera sustancial, como en este caso la interface es a cuatro bits, se debe enviar primero el nibble alto (4 bits de mayor peso) del dato y luego el nibble bajo (4 bits de menor peso). En el momento de enviar cada uno de los datos de 4 bits, las señales de control deben comportarse de la misma forma como si fuera un dato completo. Por eso en la rutina se ve que la señal RS conserva el nivel lógico adecuado y la señal E genera los dos pulsos que se requieren (el primero para el nibble alto y el segundo para el bajo).

Los módulos LCD tienen una instrucción especial para borrar la pantalla (comando 01H), pero en las pruebas que se realizaron en diferentes tipos de módulo se encontró que algunos no la aceptaban y otros requerían retardos diferentes. Por lo tanto, se optó por hacer el borrado con una subrutina llamada LIMPIA, la cual se encarga de mandar el dato correspondiente a un espacio en blanco (20H) a cada posición de memoria del módulo.

Proyecto N° 6: Comunicación serial RS-232

Interface serial RS-232

El puerto serial de las computadoras, conocido también como puerto RS-232, es muy útil ya que permite la comunicación no sólo con otras computadoras, sino también con otros dispositivos tales como el mouse, impresoras y por supuesto, microcontroladores.

Existen dos formas de intercambiar información binaria: la paralela y la serial. La comunicación paralela transmite todos los bits de un dato de manera simultánea y tiene la ventaja que la transferencia es rápida, pero la desventaja de necesitar una gran cantidad de hilos o líneas, situación que encarece los costos y se agrava cuando las distancias que separan los equipos entre los cuales se hace el intercambio es muy grande, debido a las capacitancias entre los conductores, la cual limita el correcto intercambio de datos a unos pocos metros.

La comunicación serial por su parte, transmite un bit a la vez, por lo cual es mucho más lenta, pero posee la ventaja de necesitar un menor número de líneas para la transferencia de la información y las distancias a las cuales se puede realizar el intercambio es mayor; a esto se suma que mediante dispositivos como los modem, la comunicación se pueda extender prácticamente a cualquier lugar del planeta.

Existen dos formas de comunicación serial: la sincrónica y la asincrónica. En la comunicación sincrónica, además de una línea sobre la que transfieren los datos, se necesita otra que contenga pulsos de reloj que indiquen cuando un dato es válido; la duración del bit está determinada por la duración del pulso de sincronismo. En la comunicación asincrónica, los pulsos de reloj no son necesarios y se acude a otros mecanismos para realizar la lectura/escritura de los datos; la duración de cada bit está determinada por la velocidad con la cual se realiza la transferencia de datos. En esta práctica sólo trataremos la comunicación asincrónica o asíncrona.

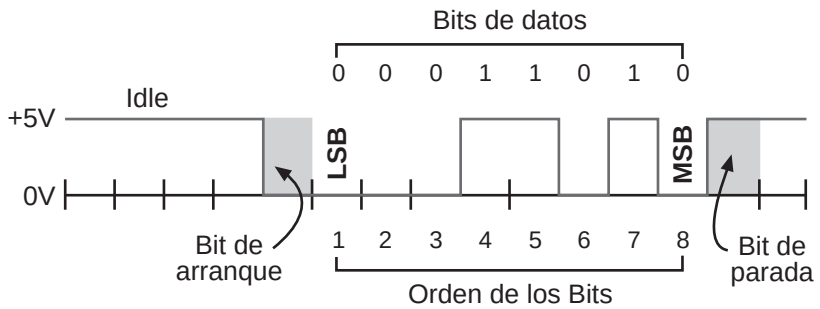


Figura 2.34. Estructura de un caracter que se transmite serialmente

La figura 2.34 muestra la estructura de un caracter que se transmite de forma asíncrona. Normalmente, cuando no se realiza ninguna transferencia de datos, la línea del transmisor es pasiva (*idle*) y permanece en un estado alto. Para empezar a transmitir datos, el transmisor coloca esta línea en bajo durante el tiempo de un bit, lo cual se conoce como bit de arranque (*start bit*) y a continuación, empieza a transmitir con el mismo intervalo de tiempo los bits correspondientes al dato (que pueden ser 7 u 8 bits), empezando por el menos significativo (*LSB*), y terminando con el

más significativo (*MSB*). Al finalizar se agrega el bit de paridad (*Parity*), si es que está activada esta opción, y los bits de parada (*Stop*) que pueden ser 1 ó 2, en los cuales la línea regresa a un estado alto. Al concluir esta operación el transmisor estará preparado para transmitir el siguiente dato.

El receptor no está sincronizado con el transmisor y desconoce cuando va a recibir datos. La transición de alto a bajo de la línea del transmisor activa al receptor y éste genera un conteo de tiempo de tal manera que realiza una lectura de la línea medio bit después del evento; si la lectura realizada es un estado alto, asume que la transición ocurrida fue ocasionada por ruido en la línea; si por el contrario, la lectura es un estado bajo, considera como válida la transición y empieza a realizar lecturas secuenciales a intervalos de un bit hasta conformar el dato transmitido. El receptor puede tomar el bit de paridad para determinar la existencia o no de errores y realizar las acciones correspondientes, al igual que los bits de parada para situaciones similares. Lógicamente, tanto el transmisor como el receptor deberán tener los mismos parámetros de velocidad, paridad, número de bits del dato transmitido y de bits de parada.

Dentro de los microcontroladores hay algunos que poseen funciones y registros especiales para las comunicaciones seriales, tales como la familia PIC16C63 o PIC16C73 de Microchip, los cuales se encargan de manejar todos los aspectos relacionados con las comunicaciones asíncronas, si previamente se han definido todos sus parámetros. Aún si el microcontrolador o microprocesador no posee la opción de las comunicaciones seriales, esta se puede implementar siempre y cuando se tenga presente la duración de cada uno de los bits en la línea. El elemento clave es detectar el bit de arranque, bien sea a través de interrupciones, o bien a través de la lectura frecuente de la línea que contiene los datos. En ambos casos, lo recomendable es que después de detectado el bit de arranque, la lectura de los bits restantes se realice en la mitad del bit, con un error permitido en cada uno de ellos del 3% del tiempo (aunque se podría extender hasta el 4%), sin que se presenten errores de lectura.

En los circuitos digitales, cuyas distancias son relativamente cortas, se pueden manejar transmisiones en niveles lógicos TTL (0 - 5V), pero cuando las distancias aumentan, estas señales tienden a degradarse debido al efecto capacitivo de los conductores y su resistencia eléctrica. El efecto se incrementa a medida que se incrementa la velocidad de la transmisión. Todo esto origina que los datos recibidos no sean iguales a los transmitidos, lo que no se puede permitir en una transferencia de datos. Una de las soluciones más inmediatas en este tipo de situaciones es aumentar los márgenes de voltaje con que se transmiten los datos, de tal manera que las perturbaciones causadas se puedan minimizar e incluso ignorar.

Ante la gran variedad de equipos, sistemas y protocolos que existen surgió la necesidad de un acuerdo que permitiera que los equipos de varios fabricantes pudieran comunicarse entre sí. A principios de los años sesenta se desarrollaron varias normas que pretendían hacer compatibles los equipos, pero en 1962 se publicó la que se convirtió en la más popular: la norma RS-232. Esta norma define la interface mecánica, las características, los pines, las señales y los protocolos que debía cumplir la comunicación serial. La norma ha sufrido algunas revisiones, como la RS-232C en 1969 y la la EIA/TIA-232E en 1991.

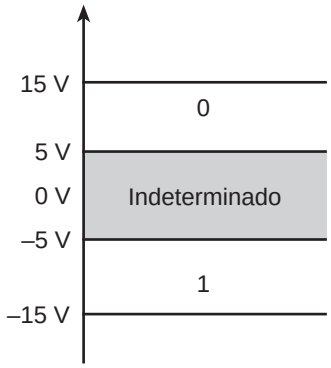


Figura 2.35. Niveles de voltaje RS-232

De todas maneras, todas las normas RS-232 cumplen básicamente con los mismos niveles de voltaje, como se puede observar en la figura 2.35:

- Un uno lógico es un voltaje comprendido entre -5V y -15V en el transmisor y entre -3V y -25V en el receptor.
- Un cero lógico es un voltaje comprendido entre 5V y 15V en el transmisor y entre 3V y 25V en el receptor.

Por lo tanto, deben existir dispositivos que permitan convertir niveles TTL a niveles RS-232 y viceversa. Los primeros dispositivos utilizados fueron los *drivers* MC1488 y los *receivers* MC1489 de Motorola, de los que se desarrollaron versiones mejoradas como los SN75188, SN75189 de *Texas Instruments* y algunos similares de otros fabricantes. Todos los dispositivos nombrados anteriormente necesitan tres voltajes diferentes para su operación cuando el equipo actúa como transmisor y receptor, lo cual no representa ningún problema en computadores tipo PC, ya que se disponen de estos voltajes en la fuente. Pero cuando se trata de sistemas de microcontroladores, en las cuales el espacio es importante y no se puede disponer de voltajes diferentes a 5 voltios, estos circuitos integrados no se pueden utilizar. Para esto se han desarrollado alternativas muy útiles, como el integrado MAX232 que describiremos más adelante.

Se debe tener presente que la norma RS-232 fue desarrollada hace más de 30 años, época en la cual los requirimientos y las capacidades de los equipos eran diferentes. En la actualidad esta norma es un poco limitada, tanto para la distancia a la cual se puede transmitir, como para la velocidad y número de transmisores y receptores que pueden estar simultáneamente conectados. Existen otras normas para la comunicación serial, en la cual se incrementa el número de trasmisores o receptores, la velocidad de transmisión, la distancia, etc. Pero a pesar de esto, los principios rectores siguen siendo los mismos de la comunicación asincrónica y de la interface RS-232.

Aspectos prácticos de una comunicación serial

El envío de niveles lógicos (bits) a través de cables o líneas de transmisión necesita la conversión a voltajes apropiados. En un circuito lógico o con microprocesador se trabaja con niveles de voltaje inferiores a 0.8 para representar el valor

lógico 0 y voltajes mayores a 2.0 para representar el valor lógico 1. Por lo general, cuando se trabaja con familias TTL y CMOS se asume que un “0” es igual a cero voltios y un “1” a +5 V.

Cuando la comunicación que se pretende hacer es muy corta, se pueden conectar directamente el transmisor y el receptor para hacer la transferencia de bits usando los mismos niveles lógicos tradicionales de 0 y 5 V. Pero cuando la distancia es mayor a los dos metros, la información digital se afecta notablemente por acción de la atenuación en el cable, el ancho de banda del mismo y la velocidad con que se transmita. La interface RS-232C es una de las diferentes soluciones que hay para esta situación. Básicamente consiste en cambiar los niveles lógicos de la salida o envío de 0 y 5V a dos niveles de voltaje de magnitud mayor: uno positivo (+V) para representar el cero lógico y uno negativo (-V) para representar el uno. En el equipo receptor de la información se realiza el proceso contrario, los niveles positivos y negativos que lleguen se convierten a los niveles lógicos tradicionales de 0 y 5V, figura 2.36. Los niveles de voltaje son simétricos con respecto a tierra y son al menos de +3V para el "0" binario y -3V para el "1". En la figura 2.37 se muestra un ejemplo de la transmisión de un caracter sobre una línea RS-232, incluyendo sus respectivos niveles de voltaje.

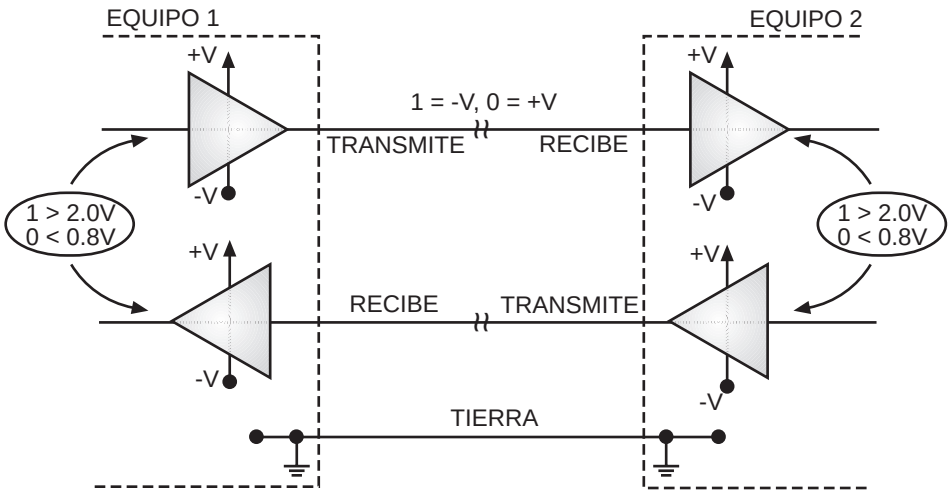


Figura 2.36. Representación de la interface RS-232

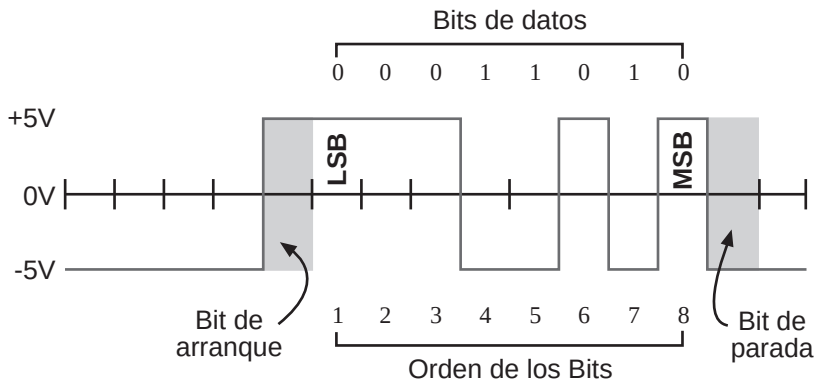
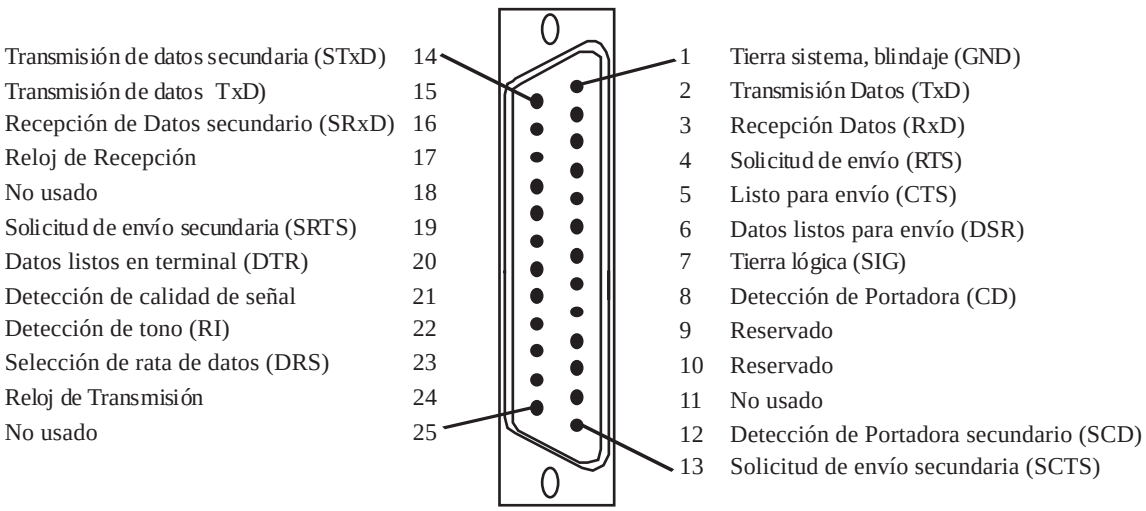


Figura 2.37. Señal presente sobre una línea RS-232

En la práctica, los niveles de voltaje los determinan las fuentes de alimentación que se apliquen a los circuitos de la interface; los niveles más comunes son desde ±12V hasta ±15V. Una interface RS-232 está compuesta por el circuito **transmisor** que convierte la señal de bajo voltaje del equipo lógico a los niveles de voltaje alto que se necesitan en la línea de transmisión y un **receptor** que realiza la función inversa. En los manuales de circuitos integrados se llama *line drivers* y *line receivers*, respectivamente, a los circuitos que ejecutan esta conversión de niveles de voltaje.

Por lo general, se utiliza con las interfaces RS-232 cable multipar o cable *ribbon* con un solo conductor como referencia de tierra. El ruido que se capta a través de la línea aún puede originar problemas. Para reducir el efecto se suele conectar un condensador en paralelo con la salida del circuito transmisor. Según las reglamentación, los estándares de la interface RS-232 permiten una separación máxima de 15 metros a una velocidad de transmisión no mayor a 9.6 kbps (kilo bits por segundo). Sin embargo, se realizan conexiones a distancias mayores sin problema alguno. En la figura 2.38 se muestran los conectores de la interface RS-232.



Nº Pin	Nombre de la señal
1	Detector de portadora (CD)
2	Recepción de Datos (RxD)
3	Transmisión de Datos (TxD)
4	Datos listos en terminal (DTR)
5	Tierra (GND)
6	Datos listos para enviar (DSR)
7	Solicitud de envío (RTS)
8	Listo para envío (CTS)
9	Detector de tono (RI)

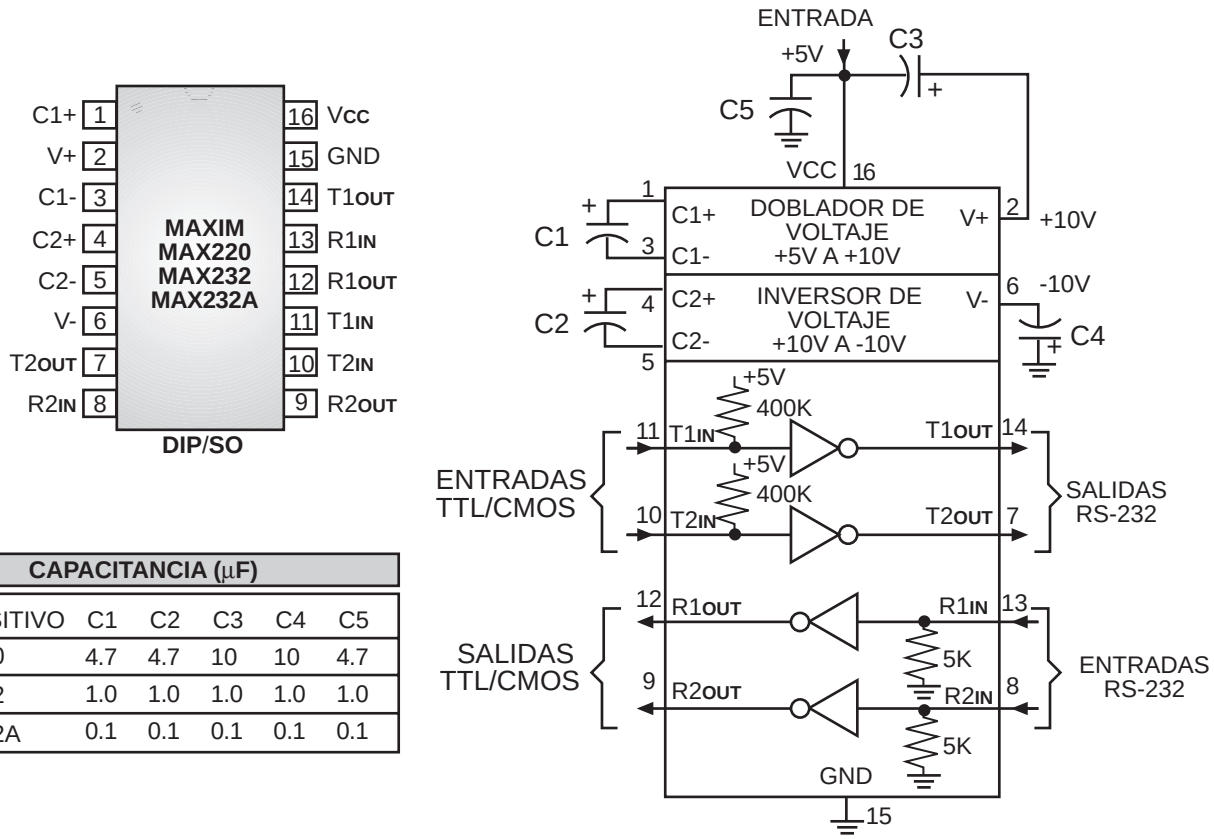
Figura 2.38. Conectores RS-232 con sus respectivos pines

El MAX232

Este circuito integrado soluciona los problemas de niveles de voltaje cuando se requiere enviar señales digitales sobre una línea RS-232. El MAX232 se usa en aquellas aplicaciones donde no se dispone de fuentes dobles de ±12 voltios; por

ejemplo, en aplicaciones alimentadas con baterías de una sola polaridad. El MAX232 necesita solamente una fuente de +5V para su operación; un elevador de voltaje interno convierte el voltaje de +5V al de doble polaridad de ±12V.

Como la mayoría de las aplicaciones de RS-232 necesitan de un receptor y un emisor, el MAX232 incluye en un solo empaque 2 parejas completas de *driver* y *receiver*, como lo ilustra la estructura interna del integrado que se muestra en la figura 2.39. El MAX232 tiene un doblador de voltaje de +5V a +10 voltios y un inversor de voltaje para obtener la polaridad de -10V. El primer convertidor utiliza el condensador C1 para doblar los +5V de entrada a +10V sobre el condensador C3 en la salida positiva V+. El segundo convertidor usa el condensador C2 para invertir +10V a -10V en el condensador C4 de la salida V-. El valor mínimo de estos condensadores los sugiere el fabricante en el recuadro de la misma figura, aunque en la práctica casi siempre se utilizan condensadores de Tantalio de 10 µF. En la tabla de la figura 2.40 se presentan algunas características de funcionamiento de este circuito integrado.



CAPACITANCIA (µF)					
DISPOSITIVO	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figura 2.39. Diagrama de pines y estructura interna del MAX232

Una aplicación clásica consiste en conectar las salidas para trasmisión serial TX y RX de un microcontrolador a una interface RS-232 con el fin de intercambiar información con una computadora. La mayoría de los sistemas concentradores de datos están compuestos por sensores conectados a microcontroladores que, a su vez, vía RS-232 le comunican los datos recolectados a un computador central. El MAX232 implementa la interface con la misma fuente de alimentación de +5 voltios. En la figura 2.41 se ilustra la conexión serial de un microcontrolador a través del MAX232.

LIMITES:			
Fuente de alimentación	-0.3 a +6V		
Voltajes de entrada:			
Tin	-0.3V a (VCC - 0.3V)		
Rin	± 30V		
Voltajes de salida:			
Tout	±15V		
Rout	-0.3V a (Vcc + 0.3V)		
Protección Corto	Continua		
Disipación de Potencia	842 mW		
CARACTERISTICAS a Vcc = +5V, C1-C4 = 0.1 μF			
	Min.	**Típ.**	**Máx.**
TRANSMISOR			
Voltaje de salida (carga 3KΩ)	±5V	±8V	
Entrada BAJA		1.4V	0.8V
Entrada ALTA	2V	1.4V	
Velocidad		200 Kb/seg.	
RECEPTOR			
Rango de entrada			±30V
Entrada BAJA	0.8V	1.3V	
Entrada ALTA		1.8V	2.4V
Resistencia de Entrada	3KΩ	5KΩ	7KΩ

Figura 2.40. Características del MAX232

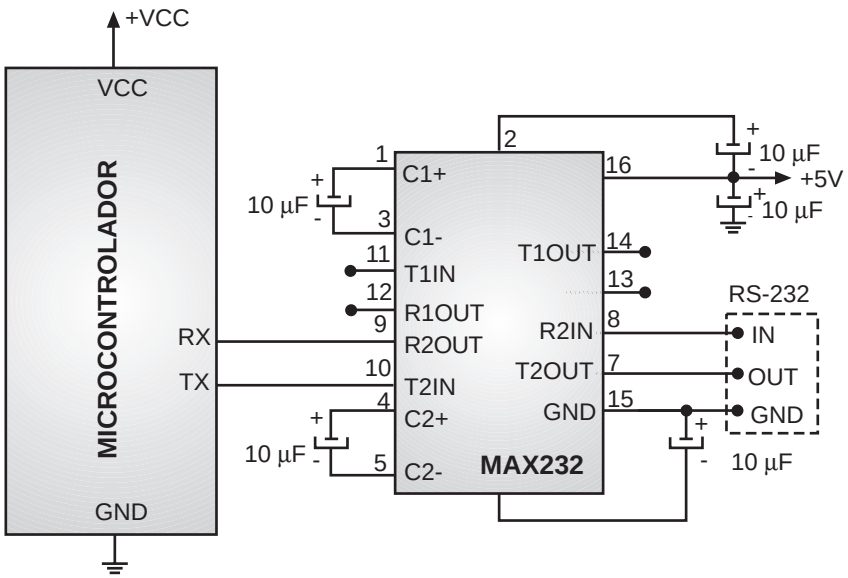


Figura 2.41. Aplicación típica del MAX232

Envío de datos seriales desde el microcontrolador hacia la computadora

El ejercicio que vamos a realizar tiene por objeto practicar la comunicación serial y entender los principios básicos que la rigen. Consiste en hacer un contador decimal (0 a 9), el cual se incrementa cada vez que se oprime un pulsador y muestra el dato del conteo en un display de 7 segmentos, a la vez que lo envía hacia la computadora para que sea mostrado en la pantalla. La comunicación entre el microcontrolador y la computadora se da en un solo sentido (del primero hacia el segundo), por lo tanto se utiliza sólo una línea de datos y el cable de tierra. En la figura 2.42 se muestra el diagrama esquemático del circuito.

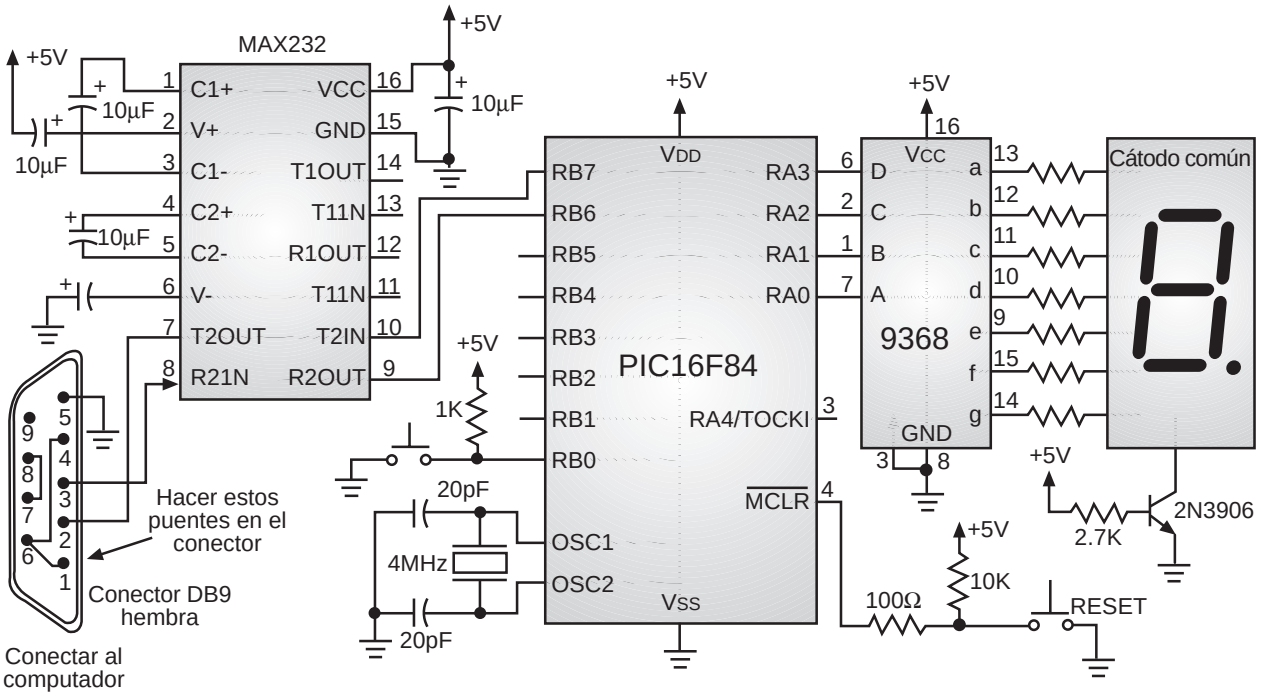


Figura 2.42. Diagrama del contador decimal que envía los datos serialmente hacia la computadora

El microcontrolador se encarga de enviar los datos serialmente con una velocidad de 1200 bps (bits por segundo), datos 8 bits, sin paridad y con un *stop bit*, esta configuración se representa como "1200, 8, N, 1". El integrado MAX232 se encarga de convertir los datos a niveles de voltaje adecuados para la línea RS-232. Debe notarse que el pin de salida del MAX232 llamado T2OUT va a conectarse al pin de recepción del puerto serial de la computadora (comparar con la figura 2.38).

Dado que la conexión hacia la computadora se realiza con un conector de 9 pines, esta se puede hacer directamente al puerto COM1 (de 9 pines), donde normalmente se conecta el *mouse*. Si se desea que este permanezca en su sitio, se requiere un adaptador RS-232 de 9 a 25 pines para que se pueda hacer la conexión al COM2 (de 25 pines).

Programa del microcontrolador. En la figura 2.43 se muestra el listado completo del programa, es muy similar al del segundo ejercicio (contador decimal) que se encuentra en la figura 2.6, la diferencia radica en que se han agregado dos rutinas.

La primera de ellas, llamada ENVIAR, se encarga de tomar el dato del registro W y transmitirlo serialmente por el pin RB7 del microcontrolador. La rutina llamada DELAY1 se encarga de hacer el retardo de tiempo necesario para sostener cada bit transmitido en la línea; ese retardo está calculado para un oscilador de 4 MHz. El cálculo es muy sencillo: si se transmiten 1200 bits en un segundo, el tiempo de cada bit es de 833µs (1/1200=833µs); como la rutina de retardo tiene un ciclo que toma cinco períodos del reloj (5 µs), se divide 833/5 y se obtiene la constante de retardo 166. Un caso especial es la rutina que transmite un bit y medio para leer el primer bit, incluyendo el retardo del *start bit*, en este caso la constante es 1.5 veces la de un bit, es decir 249. Las otras partes del programa se encargan de llevar el conteo del número de veces que se oprima el pulsador y de actualizar el display.

```
; este programa envia datos al computador via rs-232
; velocidad = 1200 ,  datos de 8 bits , sin paridad , un stop bit
indf    equ     0h      ;para direccionamiento indirecto
tmro    equ     1h      ;contador de tiempo real
pc      equ     2h      ;contador de programa
status  equ     3h      ;registro de estados y bits de control
fsr     equ     4h      ;selecccion de bancos de memoria y registros
ptoa    equ     5h      ;puertos
ptob    equ     6h
trisa   equ     85h     ;programación de los registros
trisb   equ     86h
trans   equ     0ch     ;
r0d     equ     0dh     ;
r0e     equ     0eh     ;
unidad  equ     10h     ;
decena  equ     11h     ;
centena equ     12h     ;
r14     equ     14h     ;
r1b     equ     1bh     ;
loops   equ     13h
loops2  equ     14h
conta   equ     15h
z       equ      2
rp0     equ     5h      ;selección de página
z       equ     2h      ;bandera de cero
c       equ     0h      ;bandera de carry
w       equ     0h      ;para almacenar en w
r       equ     1h      ;para almacenar en el mismo registro
tx      equ     7h

        org     00      ;vector de reset
        goto    inicio  ;va a iniciar programa principal
        org     05h
delay1  movlw   .166    ;carga para 833 µs aproximadamente
startup movwf   r0e     ;llevar valor de carga al retardo
redo    nop             ;limpiar circuito de vigilancia
        nop
        decfsz  r0e     ;decrementar retardo, saltar si cero
        goto    redo    ;repetir hasta terminar
        retlw   0       ;retornar
retardo ;subrutina de retardo de 100 milisegundos
        movlw     d'100' ;el registro loops contiene el n£mero
        movwf     loops  ;de milisegundos del retardo
top2    movlw     d'110' ;
        movwf     loops2 ;
top     nop
        nop
        nop
        nop
        nop
        nop
        decfsz    loops2    ;pregunta si terminó 1 ms
        goto      top
        decfsz    loops     ;pregunta si termina el retardo
        goto      top2
        retlw     0
enviar  ;rutina para enviar dato
        movwf   trans           ;llevar el contenido de w a transmisión
xmrt    movlw   8               ;cargar con número de bits
        movwf   r0d             ;el contador
        bcf     ptob,tx         ;colocar línea de transmisión en bajo
        call    delay1          ;para generar bit de  arranque
xnext   bcf     ptob,tx         ;colocar línea de transmisión en bajo
        bcf     status,c        ;limpiar carry
        rrf     trans           ;rotar registro de transmisión
        btfsc   status,c        ;preguntar por el carry
        bsf     ptob,tx         ;si es uno, colocar línea en alto
        call    delay1          ;llamar retardo de 1 bit
```

```
        decfsz  r0d             ;decrementar contador, saltar si cero
        goto    xnext           ;repetir hasta transmitir todo el dato
        bsf     ptob,tx         ;colocar línea de transmisión en alto
        call    delay1          ;llamar retardo 1 bit -bit de parada-
        retlw   0               ;retornar
inicio  bsf     status,rp0
        movlw   00h
        movwf   trisa
        movlw   07fh
        movwf   trisb
        bcf     status,rp0
        bsf     ptob,tx
        clrf     conta     ;inicia contador en cero
ciclo   movf      conta,w   ;el valor del contador pasa al registro w
        movwf     ptoa
        addlw     30h
        call      enviar
        call      retardo   ;retardo esperando que suelten la tecla
pulsa   btfsc     ptob,0    ;pregunta si el pulsador est  oprimido
        goto      pulsa     ;si no lo está continua revisándolo
        call      retardo   ;si está oprimido retarda 100 milisegundos
        btfsc     ptob,0    ;para comprobar
        goto      pulsa     ;si no lo está vuelve a revisar
        incf      conta     ;si lo confirma incrementa el contador
        movf      conta,w   ;carga el registro w con el valor del conteo
        xorlw     0ah       ;hace operación xor para ver si es igual a 0ah
        btfsc     status,z  ;prueba si el contador llegó a 0ah (diez)
        goto      inicio    ;si es igual el contador se pone en ceros
        goto      ciclo     ;si no llegó a diez incrementa normalmente
        end                 ;y actualiza el display
;     ******  pic16f84   ********
;     ******  wdt = off  *******
;     ******  osc = xt   *******
;     ******   cp = on   ******
```

Figura 2.43. Programa que contiene la transmisión serial hacia la computadora

Una característica bien importante del programa es que al dato que se envía hacia la computadora se le ha sumado la constante 30h (48 en decimal), esto se hace para convertir el dato decimal en su caracter ASCII equivalente. Se hace la conversión porque en transmisiones seriales es más común trabajar con caracteres de este tipo, además cuando la computadora lo reciba, se puede pasar directamente a la pantalla.

Programa de la computadora. En la computadora se requiere un programa que se encargue de configurar el puerto con los valores adecuados (1200, 8, N, 1) y de recibir el dato para pasarlo a la pantalla. En este caso utilizamos un programa en lenguaje C, debido a que es el más utilizado en aplicaciones electrónicas y permite configurar fácilmente los puertos.

En lenguaje C, existe una instrucción especial para manejar las comunicaciones seriales. Esta instrucción posee la siguiente sintaxis:

bioscom(cmd, char abyte, int port);

En realidad, esta instrucción acude a la interrupción 14H para permitir la comunicación serial sobre un puerto. Para este caso, cada uno de los parámetros tiene el

siguiente significado:

cmd	Especifica la operación a realizar
abyte	Es un caracter que se envia o recibe por el puerto serial
port	Es la identificación del puerto serial (desde 0 para COM1 hasta 3 para COM4)

El parámetro cmd puede tener los siguientes valores y significados:

0	Inicializa el puerto port con los valores dados por abyte
1	Envía el caracter abyte por el puerto port
2	Lee el caracter recibido por el puerto port
3	Retorna el estado del puerto port

Para la inicialización del puerto, el caracter abyte toma los siguientes valores y los suma para obtener el dato correspondiente:

0x02	7 bits de datos
0x03	8 bits de datos

0x00	1 bit de parada
0x04	2 bits de parada

0x00	Sin paridad
0x08	Paridad impar
0x18	Paridad par

0x00	110 bps
0x20	150 bps
0x40	300 bpss
0x60	600 bps
0x80	1200 bps
0xA0	2400 bps
0xC0	4800 bps
0xE0	9600 baudios

(0x es la notación en lenguaje C para los números hexadecimales)

Para configurar el puerto con algunos parámetros, bastará con realizar una operación OR con los deseados, por ejemplo, para el mismo ejemplo anterior, bastará con seleccionar la palabra dada por:

```
abyte = 0x80 | 0x00 | 0x00 | 0x03
```

o lo que es equivalente,

```
abyte = 0x83
```

Para configurar el puerto COM1 con los parámetros del ejemplo dado anteriormente, bastará con la instrucción:

bioscom(0,0x83,0); /* (inicializar, parámetros, COM1) */

Los programas en lenguaje C tienen su programa fuente con extensión **.C**, en este caso el programa que recibe los datos del microcontrolador por el puerto COM1 se llama RECIBE1.C. El programa que puede ejecutar el usuario se llama RECIBE1.EXE y se puede correr desde el sistema operativo DOS. El listado completo se muestra en la figura 2.44. Si se desea recibir los datos por el COM2 se debe usar la versión del programa llamada RECIBE2.C y RECIBE2.EXE que también van en el disquete que acompaña el curso. En la figura 2.45 se muestra un pantallazo del programa.

```
/* LA COMPUTADORA RECIBE LOS DATOS SERIALES ENVIADOS POR EL PIC  */
#include <conio.h>
#include <stdio.h>
#include <dos.h>
#include <math.h>
#include <bios.h>
  int puerto,COM1,COM2;
  int k,j,dato;          /*definición de variables*/
  int config;
  int COM1,COM2;
  char lectura[1];
  char dato1[2];
char leer()
{
    do{
        dato=bioscom(2,0x83,puerto);              /*leer dato recibido*/
      } while (((dato<31)|(dato>127))&(!kbhit()));
    return(dato);
 }
void main(void)
{
  COM1=0;
  COM2=1;
  puerto=COM1;         /* definir cual puerto se utiliza */
  clrscr();             /*limpiar pantalla*/
  config=0x83;  /*configurar puerto: 1200 baudios,dato de 8 bits,
                 no paridad, 1 bit de parada*/
  bioscom(0,config,puerto);        /*configuracion de los puertos*/
  gotoxy(14,4);
  printf("Curso de Microcontroladores PIC - CEKIT");
  gotoxy(8,6);
  printf("La computadora recibe los datos enviados por el micro - COM1");
  gotoxy(29,8);
  printf("Escape = Salir");
  gotoxy(23,10);
  printf("El dato del contador es:");
do{
    if(!kbhit()) dato1[0]=leer();
    if(!kbhit())
    {
      gotoxy(40,12);
      printf("%1s",dato1);
    }
    }while(!kbhit());
  clrscr();
  printf("Elaborado por: Edison Duque");
}
```

Figura 2.44. Programa en lenguaje C que recibe los datos enviados por el PIC (recibe 1.C)

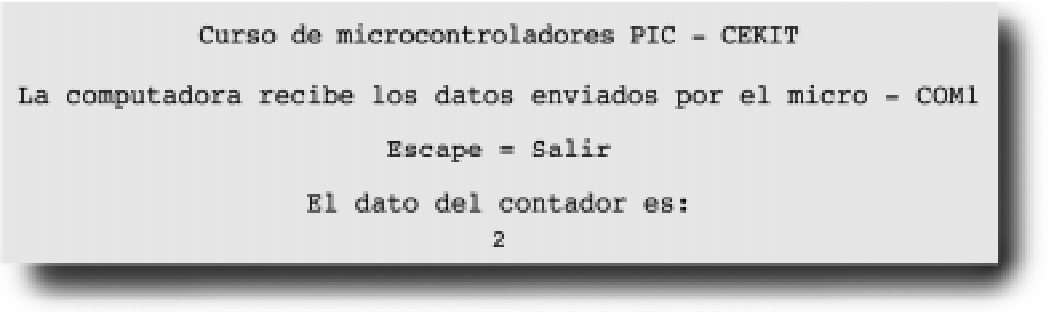


Figura 2.45. Pantallazo del programa RECIBE1

Envío de datos seriales desde la computadora hacia el microcontrolador

Este ejercicio es similar al anterior, solo que en esta ocasión el conteo decimal se lleva a cabo en la computadora (se incrementa el contador cada vez que se oprime una tecla), mostrando en la pantalla el valor del conteo. A su vez, dicho dato es enviado serialmente hacia el microcontrolador PIC, el cual los recibe luego que el integrado MAX232 adecua los niveles de voltaje. El número también es mostrado en un display de siete segmentos. El circuito es el mismo de la figura 2.42.

Programa del microcontrolador. En este caso el pin RB6 del microcontrolador se debe programar como entrada para leer los datos seriales. El principal cambio respecto al ejercicio anterior consiste en que la rutina ENVIAR se ha cambiado por la rutina RECIBIR, la cual tiene las temporizaciones y el orden exacto para recibir el dato serial. Nótese que los datos salen del computador por el pin de transmisión y se reciben en el pin R2IN del MAX232, el cual a su vez los entrega al PIC.

Aquí son válidas todas las consideraciones que se hicieron en el ejemplo anterior, por lo tanto dejamos al lector la tarea de estudiar el programa. En la figura 2.46 se muestra el listado completo con sus respectivos comentarios. Se debe notar que al dato recibido se le resta el valor 30h (48 decimal) porque se supone que la computadora envía es el caracter ASCII del número.

```
;este programa recibe datos enviados por la computadora via rs-232
;velocidad = 1200 ,  datos de 8 bits , sin paridad , un stop bit
indf    equ     0h      ;para direccionamiento indirecto
tmro    equ     1h      ;contador de tiempo real
pc      equ     2h      ;contador de programa
status  equ     3h      ;registro de estados y bits de control
fsr     equ     4h      ;selecccion de bancos de memoria y registros
ptoa    equ     5h      ;puertos
ptob    equ     6h
trisa   equ     85h     ;programación de los registros
trisb   equ     86h
r0d     equ     0dh     ;
r0e     equ     0eh     ;
conta   equ     10h
recep   equ     11h
z       equ      2
c       equ      0
rp0     equ     5h      ;selección de página
z       equ     2h      ;bandera de cero
c       equ     0h      ;bandera de carry
```

```
w       equ     0h      ;para almacenar en w
r       equ     1h      ;para almacenar en el mismo registro
rx      equ     6h
        org     00      ;vector de reset
        goto    inicio  ;va a iniciar programa principal
        org     05h

unoymedio     ;rutina para retardar bit y medio con un cristal de 4.00 mhz.
        movlw   .249            ;carga para 1250 µseg. aproximadamente
        goto    startup         ;ir a ejecutar el tiempo
delay1  movlw   .166            ;carga para 833 µs aproximadamente
startup movwf   r0e             ;llevar valor de carga al retardo
redo    nop                     ;limpiar circuito de vigilancia
        nop
        decfsz  r0e             ;decrementar retardo, saltar si cero
        goto    redo            ;repetir hasta terminar
        retlw   0               ;retornar

recibir nop
        clrf    recep     ;limpiar registro de recepción
        btfsc   ptob,rx   ;l¡nea de recepción est  en bajo?
        goto    recibir   ;si no lo est , volver a leer
        call    unoymedio ;llamar rutina uno y medio bits
rcvr    movlw   8         ;cargar contador con
        movwf   conta     ;el número de bits
rnext   bcf     status,c  ;limpiar carry
        btfsc   ptob,rx   ;preguntar por el estado de la l¡nea
        bsf     status,c  ;activar carry si está en alto
        rrf     recep     ;rotar registro de recepción
        call    delay1    ;llamar rutina de un bit
        decfsz  conta     ;decrementar contador, saltar si cero
        goto    rnext     ;repetir hasta completar dato
        retlw   0         ;retornar

inicio  bsf     status,rp0
        movlw   00h
        movwf   trisa
        movlw   0ffh
        movwf   trisb
        bcf     status,rp0
        clrf    recep
        clrf    ptoa
ciclo   call    recibir
        movlw   30h
        subwf   recep,w
        movwf   ptoa
        goto    ciclo
        end
```

Figura 2.46. Programa que recibe los datos seriales enviados por la computadora

Programa de la computadora. En este caso el programa se encarga de realizar el conteo del número de veces que se pulsa una tecla y de mostrar ese número en la pantalla, a la vez que lo envía por el puerto serial hacia el microcontrolador. En la figura 2.47 se muestra el listado completo del programa. En este caso también se tienen dos versiones, una llamada ENVIA1 para trabajar por el COM1 y otra llamada ENVIA2 para trabajar con el COM2. En la figura 2.48 se muestra un pantallazo del mismo.

En la instrucción que envía el dato del conteo se suma el valor 30h (48 decimal), para convertir el número del contador en su equivalente ASCII.

```
/* LA COMPUTADORA ENVIA DATOS SERIALES AL PIC  */

#include <conio.h>
#include <stdio.h>
#include <bios.h>

int COM1,COM2,Puerto;   /*definición de variables*/
int j,envio,configuracion;
int contador;
char tecla;

void main(void)
{
  clrscr();       /*limpiar pantalla*/
  COM1=0;         /*constantes de los puertos del PC*/
  COM2=1;
  Puerto=COM1;                    /*Indicar si es COM1 o COM2*/

  configuracion=0x83;                /*conf.puerto: 1200,8,N,1*/
  bioscom(0,configuracion,Puerto);     /*inicializa el COM del PC*/

  gotoxy(20,2);
  printf(« Curso de Microcontroladores PIC  «);

  gotoxy(8,5);
  printf("Envio de datos seriales hacia el microcontrolador - COM1");

  gotoxy(10,7);
  printf("Enter = Incremento del contador     Escape = Salir");

  gotoxy(24,10);
  printf("El dato del contador es:");
  contador=0;

do{         /*ciclo de lectura de medida*/
      tecla=getch();
      contador++;
      if(contador==10) contador=0;
      gotoxy(34,12);
      printf("%d",contador);         /*Obtiene tecla oprimida*/
      envio=bioscom(1,contador+0x30,Puerto);    /*envía caracter al micro*/
   }while(tecla!=27);  /*Hasta que se oprima ESC*/

  while(!kbhit());
  clrscr();
  printf("Elaborado por: Edison Duque");

}
```

Figura 2.47. Programa en lenguaje C para enviar datos seriales al PIC (envía1.C)

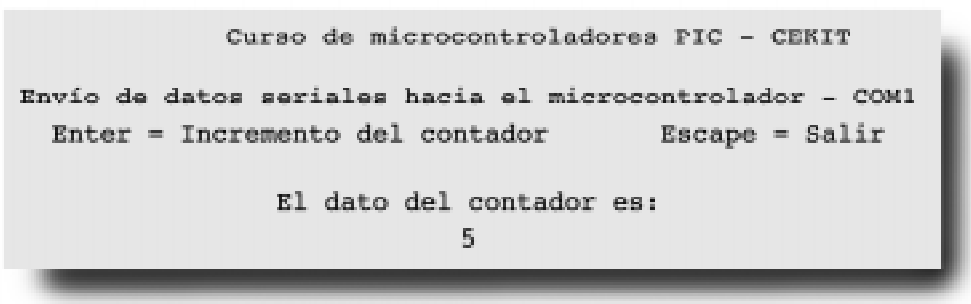


Figura 2.48. Pantallazo del programa ENVIA

Proyecto N° 7: Características especiales de los PIC

Los microcontroladores poseen muchas herramientas especiales, tales como temporizadores, convertidores A/D, interrupciones, comunicaciones seriales, etc. Pero en la mayoría de las ocasiones no se utilizan porque el diseñador desconoce la forma de emplearlos o porque el tiempo de desarrollo es corto y no hay tiempo de estudiarlas o practicar con ellas. En estos casos recurrimos a soluciones de *software* y *hardware* que nos sean familiares y de fácil uso para resolver el problema, aunque esto implique sacrificar el carácter óptimo que deben llevar los diseños.

La familia de microcontroladores PIC no es la excepción, ellos poseen muchas cualidades que la mayoría de las personas no utilizan. En esta práctica vamos a explicar algunas de ellas, de una manera clara y simple, para que el lector las pueda utilizar en sus propios diseños. Para esto vamos a tomar como base el circuito simple que se muestra en la figura 2.49, el cual consta de un microcontrolador PIC16F84 (se puede uilizar un 16C61 o un 16C71), un decodificador y un display, con ellos se pretende construir un contador de década (de 0 a 9) que nos permitirá estudiar todos los casos que veremos.

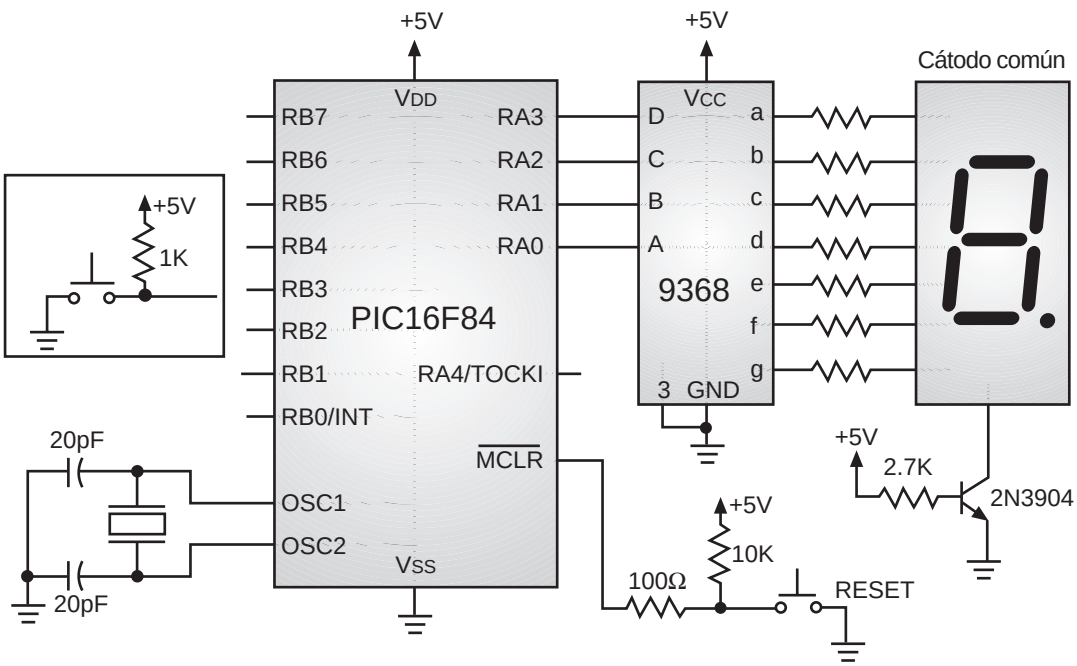


Figura 2.49. Circuito del contador decimal que se utiliza en los ejercicios

Uso de las interrupciones

Estos microcontroladores poseen varias fuentes de interrupción: interrupción externa, finalización del temporizador/contador, cambio en las líneas RB4 a RB7 y finalización de escritura en la EEPROM de datos. El registro 0Bh o INTCON contiene los bits que corresponden a las banderas de estado de las interrupciones y los bits de habilitación para cada una de ellas, figura 1.14. Sólo la bandera de finalización de la escritura reside en el registro 88h o EECON1.

Si el bit GIE (*Global Interrupt Enable*) se coloca en 0 deshabilita todas las interrupciones. Cuando una interrupción es atendida el bit GIE se coloca en 0 automáticamente para deshabilitar otras interrupciones que se puedan presentar, la dirección actual del PC se carga en la pila o stack y el PC se carga con el valor 04h, por lo tanto, la parte del programa que atiende las interrupciones se debe esribir a partir de dicha dirección. Una vez en la rutina de servicio, la fuente de interrupción se puede determinar examinando las banderas. Como la bandera que indica la causa se encuentra en "1", el programa debe encargarse de ponerla en "0" nuevamente antes de salir de la rutina que atiende las interrupciones, esto es para evitar que se vuelva a atender la misma interrupción en repetidas veces.

La instrucción RETFIE permite al usuario retornar de la interrupción, a la vez que habilita de nuevo las interrupciones, ya que pone el bit GIE en "1" automáticamente. A continuación veremos ejemplos de como trabajan algunas de las interrupciones:

Interrupción externa. Actúa en el pin RB0/INT y puede ser configurada para activarse con el flanco de subida o el de bajada, de acuerdo al bit INTEDG del registro OPTION, figura 1.15. Cuando se presenta un flanco válido en el pin INT, la bandera INTF (INTCON,1) se coloca en "1" y el contador de programa o PC salta a la dirección 04h. En ese momento se puede hacer la verificación de la causa de la interrupción consultando el estado de la bandera, si es del caso también se debe probar el estado de alguno de los pines del microcontrolador para confirmar. La interrupción se puede habilitar o deshabilitar utilizando el bit de control INTE (INTCON,4) en "0". Cuando se atiende la interrupción en la rutina de servicio, el programa debe poner la bandera INTF en "0", antes de regresar al programa principal.

El ejercicio que se ha tomado como ejemplo consiste en un contador decimal (0 a 9), el cual se incrementa cada vez que se presenta una interrupción a través del pin INT, provocada por un pulsador externo conectado a dicho pin y que tiene una resistencia de 1 kohm conectada a la fuente de alimentación para fijar un nivel lógico alto en estado de reposo, en la figura 2.49 se tiene un recuadro con el pulsador que se ha conectado al pin RB0/INT.

El programa que se graba en el microcontrolador se muestra en la figura 2.50 y tiene las siguientes características:

- Una subrutina de retardo de 100 milisegundos que sirve para comprobar que el pulsador si fue oprimido y descartar los pulsos de rebote.
- Al inicio se programan los puertos en los registros TRISA y TRISB, al igual que se habilita la interrupción externa con los bits respectivos en registro INTCON y OPTION.
- El ciclo en que se queda enclavado el programa principal no hace nada, solamente esperar a que se presente la interrupción para atenderla.
- La rutina que atiende la interrupción comprueba que el pulsador esté oprimido, además de probar que se haya activado la bandera correspondiente a la interrupción externa. Si las condiciones son favorables, se incrementa el contador y por lo tanto el número que se muestra en el display.

```
;Este programa hace un contador decimal en
;un display de 7 segmentos, se incrementa cada vez que el
;microcontrolador tienen una interrupción por el pin RB0/INT
status    equ       03h       ;registro de estados
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
intcon    equ       0bh       ;registro de control y banderas de interrupción
conta     equ       0ch       ;lleva el conteo de pulsaciones
loops     equ       0dh       ;utilizado en retardos (milisegundos)
loops2    equ       0eh       ;utilizado en retardos
opcion    equ       81h       ;configuración del flanco de interrupción
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B
z         equ       02h       ;bandera de cero del registro de estados
w         equ       00h       ;indica que el resultado se guarda en W

reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa
          org       4         ;aquí se atiende la interrupción

          call      retardo   ;retardo para confirmar pulsador
          btfsc     ptob,0    ;pregunta por el pin RB0
          goto      sale      ;si no está oprimido regresa
          btfss     intcon,1  ;confirma si la interrupción fue por el pin INT
          goto      sale      ;si no lo es sale
          incf      conta     ;incrementa el contador
          movf      conta,w   ;carga el registro W con el valor del conteo
          xorlw     0ah       ;hace operación xor para ver si es igual a 0ah
          btfsc     status,z  ;prueba si el contador llegó a 0ah (diez)
          clrf      conta     ;si llegó a diez pasa el conteo a 0
          movf      conta,w   ;pasa el dato al display
          movwf     ptoa      ;
          call      retardo   ;retardo de 100 milisegundos
sale      bcf       intcon,1  ;la bandera de la interrupción se debe
                              ;poner en 0 antes de regresar (INTF)
          retfie              ;regresa al programa principal y habilita otra vez
                              ;la interrupción al poner el bit GIE en 1

retardo   ;subrutina de retardo de 100 milisegundos
          movlw     D'100'    ;el registro loops contiene el número
          movwf     loops     ;de milisegundos del retardo
top2      movlw     D'110'    ;
          movwf     loops2    ;
top       nop
          nop
          nop
          nop
          nop
          nop
          decfsz    loops2    ;pregunta si terminó 1 ms
          goto      top
          decfsz    loops     ;pregunta si termina el retardo
          goto      top2
          retlw     0

inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f0
          movwf     trisa     ;se programan los pines del puerto A como salidas
          movlw     0ffh      ;se carga el registro W con ff
          movwf     trisb     ;se programan los pines del puerto B como entradas
          movlw     80h       ;en el registro OPTION sólo se programa
          movwf     opcion    ;el flanco de bajada para el pin INT
          bcf       status,5  ;se ubica en el primer banco de memoria RAM
          movlw     90h       ;en el registro INTCON se habilita la
          movwf     intcon    ;interrupción por el pin INT

          clrf      conta     ;inicia contador en cero
          movf      conta,w   ;el valor del contador pasa al registro W
          movwf     ptoa      ;pasa el valor de W al puerto A (display)
```

```
ciclo     nop                ;espera que se presente una interrupción
          nop
          goto      ciclo
          end

;=====================================================================
;         Fusibles de programación
;         Osc                XT
;         Watchdog           OFF
;         Code protect       OFF
;         Power-Up-Timer     ON
;         Micro.             PIC16F84
;=====================================================================
```

Figura 2.50. Programa para probar la interrupción externa

Interrupción por cambio en el puerto B. El funcionamiento de esta interrupción es muy particular, se presenta cuando ocurre un cambio de nivel en alguno de los pines RB4 a RB7 del microcontrolador. En este caso la bandera RBIF (INTCON,0) se colocará en "1". El bit de control respectivo es RBIE (INTCON,3). El funcionamiento es similar al que se explicó para la interrupción externa, solamente que se deben cambiar los bits del registro INTCON que se utilizan.

El ejemplo que se utiliza emplea nuevamente el contador decimal, la diferencia es que en lugar del pulsador conectado al pin RB0/INT se emplean cuatro interruptores (dipswitch) conectados a los pines RB4 a RB7, como los que se muestran en el recuadro de la figura 2.49. Como se puede ver en estado normal los pines del microcontrolador tienen un estado lógico alto determinado por las resistencias de 1 kohm conectadas a los dipswitch, si el sistema se enciende en estas condiciones no ocurre nada especial, pero si luego de encendido se cambia la posición de alguno de los interruptores, el microcontrolador sufrirá una interrupción y al atenderla hará que el contador decimal se incremente como en el ejercicio anterior. Si el interruptor se deja en ese estado, el microcontrolador tendrá repetidas interrupciones, hasta el momento en que se regrese a su estado inicial.

El programa que se escribe en el microcontroaldor se muestra en la figura 2.51, es muy similar al del primer ejercicio. La diferencia está en los bits que habilitan las interrupciones y que se graban en el registro INTCON.

Interrupción por finalización de la temporización. El temporizador/contador es una de las herramientas más valiosas que se tienen en el microcontrolador, se encuentra en la posición de memoria 01h. Su conteo se puede incrementar con una señal externa aplicada al pin RA4/TOCKI o con la señal del reloj interno del microcontrolador (en este caso cada microsegundo). Su rata de incremento se puede afectar (prolongar) mediante una preescala o divisor de frecuencia que se programa en el registro OPTION. Allí también se selecciona si el incremento es con flanco de subida o de bajada y si la fuente de pulsos es externa (pin RA4/TOCKI) o interna (oscilador).

Cuando el conteo del temporizador llega a 0FFh y pasa a 00h se genera una interrupción (siempre y cuando esté habilitada), el bit TOIF (INTCON,2) se pondrá en "1". El bit de control respectivo es TOIE (INTCON,5).

```
;Este programa hace un contador decimal en
;un display de 7 segmentos, se incrementa cada vez que el
;microcontrolador tienen una interrupción por cambio en los pines RB4 a RB7
status    equ       03h       ;registro de estados
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
intcon    equ       0bh       ;registro de control y banderas de interrupción
conta     equ       0ch       ;lleva el conteo de pulsaciones
loops     equ       0dh       ;utilizado en retardos (milisegundos)
loops2    equ       0eh       ;utilizado en retardos
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B
z         equ       02h       ;bandera de cero del registro de estados
w         equ       00h       ;indica que el resultado se guarda en W

reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa
          org       4         ;aquí se atiende la interrupción

          call      retardo   ;retardo
          btfss     intcon,0  ;confirma interrupción por cambio en RB<7:4>
          goto      sale      ;si no lo es sale
          incf      conta     ;incrementa el contador
          movf      conta,w   ;carga el registro W con el valor del conteo
          xorlw     0ah       ;hace operación xor para ver si es igual a 0ah
          btfsc     status,z  ;prueba si el contador llegó a 0ah (diez)
          clrf      conta     ;si llegó a diez pasa el conteo a 0
          movf      conta,w   ;pasa el dato al display
          movwf     ptoa      ;
          call      retardo   ;retardo de 100 milisegundos
sale      bcf       intcon,0  ;la bandera de la interrupción se debe
                              ;poner en 0 antes de regresar (RBIF)
          retfie              ;regresa al programa principal y habilita otra
                              ;vez la interrupción al poner el bit GIE en 1

retardo   ;subrutina de retardo de 100 milisegundos
          movlw     D'100'    ;el registro loops contiene el número
          movwf     loops     ;de milisegundos del retardo
top2      movlw     D'110'    ;
          movwf     loops2    ;
top       nop
          nop
          nop
          nop
          nop
          nop
          decfsz    loops2    ;pregunta si termino 1 ms
          goto      top
          decfsz    loops     ;pregunta si termina el retardo
          goto      top2
          retlw     0

inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f0
          movwf     trisa     ;se programa el puerto A como salidas
          movlw     0ffh      ;se carga el registro W con ff
          movwf     trisb     ;se programa el puerto B como entradas
          bcf       status,5  ;se ubica en el primer banco de memoria RAM
          movlw     88h       ;en el registro INTCON se habilita la
          movwf     intcon    ;interrupción cambio en los pines RB4 a RB7
          clrf      conta     ;inicia contador en cero
          movf      conta,w   ;el valor del contador pasa al registro W
          movwf     ptoa      ;pasa el valor de W al puerto A (display)
ciclo     nop                 ;espera que se presente una interrupción
          nop
          goto      ciclo
          end
```

Figura 2.51. Prueba de la interrupción por cambio en puerto B

El ejercicio que realizamos para esta prueba consiste en el incremento del temporizador por medio del reloj interno del microcontrolador, dicho conteo está afectado por la máxima preescala que se puede seleccionar (división por 256). Como el oscilador es un cristal de 4 MHz se tiene un ciclo de instrucción de 1 microsegundo, por lo tanto, dado que el registro TMR0 es de 8 bits y que la preescala es de 256 , el tiempo que se tarda en contar hasta 0FFh es de 256x256 microsegundos, es decir 65.536 microsegundos.

El tiempo logrado es muy pequeño para ser percibido , por lo que recurrimos a establecer un contador dentro de la rutina que atiende las interrupciones, este se incrementa cada vez que el programa pase por ese sitio, es decir cada 65,5 milisegundos. Cuando llegue a 10 hará que se incremente el contador decimal que se ha venido tratando en los otros ejercicios. De esta manera el experimentador verá que el display se incrementa cada 0.65 segundos aproximadamente.

En el programa que se muestra en la figura 2.52 se explica cada uno de los pasos que se llevan a cabo. Los valores con que al inicio se programan el registro INTCON y el registro OPTION, se pueden comparar con las tablas que se muestran en las figuras 1.14 y 1.15 para entender mejor la función de cada uno de ellos.

Consideraciones especiales. Se debe tener en cuenta que el registro de trabajo W y el registro de estados no se guardan cuando se atiende una interrupción, por lo tanto se recomienda almacenar su contenido en algún registro temporal ya que estos pueden sufrir cambios en la rutina que atiende dichas interrupciones, lo que puede causar errores cuando se regresa al programa principal.

Uso del circuito de vigilancia o watchdog timer

El *watchdog* es una herramienta que se utiliza para restablecer el programa del microcontrolador cuando este se ha salido de su flujo normal o se ha perdido por un ruido o interferencia electromagnética. Su funcionamiento se habilita por medio de un fusible de configuración que se selecciona (ON/OFF) cada vez que se graba el microcontrolador.

Funciona como un oscilador interno, independiente del oscilador principal del micro, con un período nominal de 18 milisegundos. Cada vez que se cumple el tiempo del *watchdog* el microcontrolador sufre un reset, por lo tanto, se debe usar una instrucción especial que reinicie dicho conteo antes de que se termine. Esa instrucción es CLRWDT. El período de 18 milisegundos se puede ampliar hasta casi 2.3 segundos utilizando la preescala del registro OPTION, en el cual existe también un bit que permite seleccionar si la preescala se asigna al *watchdog* o al temporizador/contador. El uso de la preescala se permite sólo para uno de los dos a la vez.

En el programa que se muestra en la figura 2.53 se tiene un ejemplo muy sencillo que ilustra el funcionamiento del *watchdog*. Consiste en hacer que el microcontrolador sufra reset continuamente, esto se logra habilitando el circuito de vigilancia en el momento de grabar el microcontrolador (seleccionando *watchdog timer* ON). Cada vez que el programa se reinicia hace que se conmute el dato que se muestra en

```
;Este programa hace un contador decimal en
;un display de 7 segmentos, se incrementa cada vez que el
;microcontrolador tiene una interrupción por el tmer TMR0
status    equ       03h       ;registro de estados
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
intcon    equ       0bh       ;registro de control y banderas de interrupción
conta     equ       0ch       ;lleva el conteo de pulsaciones
conta2    equ       0fh       ;cuenta 10 interrupciones de TMR0
opcion    equ       81h       ;configuración del temporizador TMR0
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B
z         equ       02h       ;bandera de cero del registro de estados
w         equ       00h       ;indica que el resultado se guarda en W

reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa

          org       4         ;aquí se atiende la interrupción

          btfss     intcon,2  ;confirma si la interrupción fue por el TMR0
          goto      sale      ;si no lo es sale
          incf      conta2    ;incrementa el contador de diez interrupciones
          movf      conta2,w  ;carga el registro W con el valor del conteo
          xorlw     0ah       ;hace operación xor para ver si es igual a 0ah
          btfss     status,z  ;prueba si el contador llegó a 0ah (diez)
          goto      sale
          clrf      conta2    ;si llegó a diez pasa el conteo a 0
          incf      conta     ;incrementa el contador
          movf      conta,w   ;carga el registro W con el valor del conteo
          xorlw     0ah       ;hace operación xor para ver si es igual a 0ah
          btfsc     status,z  ;prueba si el contador llegó a 0ah (diez)
          clrf      conta     ;si llegó a diez pasa el conteo a 0
          movf      conta,w   ;pasa el dato al display
          movwf     ptoa      ;
sale      bcf       intcon,2  ;la bandera de la interrupción se debe
                              ;poner en 0 antes de regresar (TOIF)
          retfie              ;regresa al programa principal
                              ;la interrupción al poner el bit GIE en 1
inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f0
          movwf     trisa     ;se programan los pines del puerto A como salidas
          movlw     0ffh      ;se carga el registro W con ff
          movwf     trisb     ;se programan los pines del puerto B como entradas
          movlw     b'11000111' ;en el registro OPTION sólo se programa
          movwf     opcion    ;el flanco de bajada para el pin INT
          bcf       status,5  ;se ubica en el primer banco de memoria RAM
          movlw     b'10100000' ;en el registro INTCON se habilita la
          movwf     intcon    ;interrupción por el TMR0
          clrf      conta2
          clrf      conta     ;inicia contador en cero
          movf      conta,w   ;el valor del contador pasa al registro W
          movwf     ptoa      ;pasa el valor de W al puerto A (display)
ciclo     nop                 ;espera que se presente una interrupción
          nop
          goto      ciclo
          end
;=====================================================================
;         Fusibles de programación
;         Osc                 XT
;         Watchdog            OFF
;         Code protect        OFF
;         Power-Up-Timer      ON
;         Micro.              PIC16F84
;=====================================================================
```

Figura 2.52. Programa para comprobar la interrupción por fin de la temporización

```
;Este programa hace que el microcontrolador sufra un reset
;cada 2 segundos aproximadamente, cada vez que se resetea cambia
;el dato del display, el reset lo causa el watchdog timer
status    equ       03h       ;registro de estados
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
cambia    equ       0fh       ;registro con el bit que hace cambiar el display
opcion    equ       81h       ;configuración del flanco de interrupción
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B

reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa

inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f
          movwf     trisa     ;se programa el puerto A como salidas
          movlw     0ffh      ;se carga el registro W con 00
          movwf     trisb     ;se programa el puerto B como entradas
          movlw     b'11001111' ;en el registro OPTION se programa
          movwf     opcion    ;la preescala del watchdog (en este caso 128)
          bcf       status,5  ;se ubica en el primer banco de memoria RAM
          btfss     cambia,0  ;pregunta el estado de la bandera de prueba
          goto      uno

cero      clrf      ptoa      ;pone en 0 el dato del display
          bcf       cambia,0  ;conmuta la bandera de prueba
          ;clrwdt             ;borrar el conteo del watchdog timer
          goto      cero      ;se queda en este ciclo esperando el reset

uno       movlw     01        ;dato para el display
          movwf     ptoa
          bsf       cambia,0  ;conmuta la bandera de prueba
          ;clrwdt             ;borrar el conteo del watchdog timer
          goto      uno       ;se queda en este ciclo esperando el reset
          end
;=================================================================
;         Fusibles de programación
;         Osc                 XT
;         Watchdog            ON
;         Code protect        OFF
;         Power-Up-Timer      ON
;         Micro.              PIC16F84
;=================================================================
```

Figura 2.53. Programa del ejercicio con el *watchdog timer*

el display de siete segmentos (1 ó 0), dado que se ha programado el registro OPTION para que se asigne la máxima preescala al temporizador del *watchdog*, el display cambia cada 2.3 segundos aproximadamente.

De esta manera comprobamos que el micro se está reseteando cada vez que termina el conteo del *watchdog* y no se utiliza la instrucción CLRWDT para reiniciarlo. Luego de esta prueba quitamos el punto y coma (;) que precede a la instrucción CLRWDT con el objeto de habilitarla. En este caso el display no cambia de estado, lo que nos demuestra que el microcontrolador no está sufriendo reset como en el caso anterior.

Cuando se va a utilizar el *watchdog* en un programa complejo debemos asegurarnos que el flujo del programa permita ubicar una o varias instrucciones CLRWDT en sitios específicos, por los cuales se pase antes de que se cumpla el período del temporizador, con el objeto de hacer que el programa se reinicie en caso de que el

programa se quede enclavado en un ciclo en el que no debía quedarse. Se debe tener claridad en que por tratarse de un oscilador del tipo RC, su período de tiempo puede variar un poco con las condiciones de voltaje y temperatura, por lo que se recomienda tener unos buenos niveles de tolerancia en los tiempos que se manejan.

Uso de la memoria de datos EEPROM en el PIC16F84

La memoria de datos EEPROM es una de las herramientas más valiosas que tiene el PIC16F84, se puede emplear para almacenar los datos de calibración en un instrumento, un código o clave de seguridad en una alarma, la hora en un sistema de tiempo real, etc. Este microcontrolador posee 64 bytes de memoria EEPROM, el acceso a estas posiciones se consigue a través de dos registros de memoria RAM:

- El registro EEADR (posición 09h), que debe contener la dirección o posición de la memoria EEPROM de datos que se va a leer o escribir.
- El registro EEDATA (posición 08h), que contiene el dato de 8 bits que se va a escribir o el que se leyó.

Adicionalmente existen dos registros de control, el EECON1 (posición 88h) que posee cinco bits que controlan las operaciones de lectura y escritura y el EECON2 (posición 89h), que aunque no está implementado físicamente, es necesario para las operaciones de escritura. En la figura 1.16 se muestra el registro EECON1, debe ponerse especial atención a la función de sus bits para comparalos con los valores que toman en el programa que se utiliza como ejemplo.

La lectura toma un ciclo del reloj de instrucciones (en este caso un microsegundo), mientras que la escritura, por ser controlada por un temporizador incorporado y requerir una operación previa de borrado de la posición de interés, tiene un tiempo nominal de 10 milisegundos; este tiempo puede variar con la temperatura y el voltaje. Según el fabricante, el número típico de ciclos de borrado/escritura de la EEPROM de datos es de 1’000.000 (un millón).

El programa que se muestra en la figura 2.54 es un simple ejemplo de un contador decimal que se incrementa con un pulsador conectado al pin RB0 y muestra el dato en un display de siete segmentos, lo interesante es que el número del conteo se guarda en la posición 00 de la memoria EEPROM de datos, por lo tanto, si se retira la alimentación del sistema, esta puede recuperar el valor que tenía una vez que se restablezca el voltaje. Las rutinas LEER y ESCRIB se encargan de recuperar el dato guardado en la memoria y de almacenarlo nuevamente cuando se ha incrementado, respectivamente.

Como función especial se ha dispuesto al inicio del programa una verificación que permite establecer si el número que se lee de la memoria de datos está entre 0 y 9, esto se hace con el fin de garantizar que la primera vez que se conecte la alimentación al sistema, el dato que se muestre en el display sea menor o igual a 9. Se supone que para las ocasiones siguientes en que se conecte la fuente, el dato ya estará entre los valores adecuados, por lo tanto en esas ocasiones esa verificación será transparente.

```
;Este programa hace un contador decimal en
;un display de 7 segmentos. el número que lleva el conteo
;se guarda en la EEPROM de datos del micro
status    equ       03h       ;registro de estados
ptoa      equ       05h       ;el puerto A está en la dirección 05 de la RAM
ptob      equ       06h       ;el puerto B está en la dirección 06 de la RAM
eedata    equ       08h       ;registro de datos de la memoria EEPROM
eeadr     equ       09h       ;registro de direcciones de la memoria EEPROM
conta     equ       0ch       ;lleva el conteo de pulsaciones
loops     equ       0dh       ;utilizado en retardos (milisegundos)
loops2    equ       0eh       ;utilizado en retardos
conta2    equ       0fh
trisa     equ       85h       ;registro de configuración del puerto A
trisb     equ       86h       ;registro de configuración del puerto B
eecon1    equ       88h       ;registro de control de la memoria EEPROM
eecon2    equ       89h       ;registro de control de la memoria EEPROM
z         equ       02h       ;bandera de cero del registro de estados
w         equ       00h       ;indica que el resultado se guarda en W
c         equ       00h       ;bandera de carry
;bits especiales del registro eecon1
eeif      equ       04h
wrerr     equ       03h
wren      equ       02h
wr        equ       01h
rd        equ       00h

reset     org       0         ;el vector de reset es la dirección 00
          goto      inicio    ;se salta al inicio del programa

          org       5         ;el programa empieza en la dirección 5

retardo   ;subrutina de retardo de 100 milisegundos
          movlw     D'100'    ;el registro loops contiene el número
          movwf     loops     ;de milisegundos del retardo
top2      movlw     D'110'    ;
          movwf     loops2    ;
top       nop
          nop
          nop
          nop
          nop
          nop
          decfsz    loops2    ;pregunta si terminó 1 ms
          goto      top
          decfsz    loops     ;pregunta si termina el retardo
          goto      top2
          retlw     0

leer      bsf       status,5  ;se ubica en segundo banco de RAM
          bsf       eecon1,rd ;pone el bit que inicia la lectura
          bcf       status,5  ;vuelve al primer banco de memoria
          movf      eedata,w  ;el dato leído se pasa al registro W
          movwf     conta2    ;se guarda el dato en conta2
          movwf     conta
          return

escrib    bsf       status,5  ;se ubica en el segundo banco de RAM
          bsf       eecon1,wren  ;habilita escritura en memoria EEPROM
          bcf       eecon1,eeif  ;se asegura que la bandera esté en cero

          movlw     055h      ;esta secuencia es obligatoria
          movwf     eecon2    ;para escribir en la memoria de datos EEPROM
          movlw     0aah
          movwf     eecon2

          bsf       eecon1,wr ;orden de escribir el dato que se cargo
                              ;previamente en el registro eedata en la
                              ;posición de memoria direccionada por eeadr
```

```
espera    btfss     eecon1,eeif ;pregunta si terminó la escritura
          goto      espera    ;si no, espera a que termine
          bcf       eecon1,eeif  ;borra la bandera de fin de escritura
          bcf       eecon1,wren  ;deshabilita la escritura en memoria EEPROM
          bcf       status,5  ;se ubica en el primer banco de RAM
          retlw     0

inicio    bsf       status,5  ;se ubica en el segundo banco de RAM
          movlw     0f0h      ;se carga el registro W con 0f0
          movwf     trisa     ;se programan los pines del puerto A como salidas
          movlw     0ffh      ;se carga el registro W con ff
          movwf     trisb     ;se programan los pines del puerto B como entradas
          bcf       status,5  ;se ubica en el primer banco de memoria RAM
          clrf      eeadr     ;cuando se enciende el sistema se verifica
          call      leer      ;que el dato guardado en memoria esté entre 0 y 9
          movlw     0ah       ;la prueba se hace porque la primera vez que
          subwf     conta2,w  ;se encienda el sistema se puede tener un número
          btfss     status,c  ;fuera del rango, para las ocasiones
          goto      ciclo     ; posteriores el proceso es invisible

ini2      clrf      conta     ;inicia contador en cero
          clrf      eedata
          call      escrib    ;inicia dato de memoria en 0

ciclo     call      leer      ;leer memoria, devuelve dato en W
          movwf     ptoa      ;pasa el valor de W al puerto A (display)
          call      retardo   ;retardo esperando que suelten la tecla

pulsa     btfsc     ptob,0    ;pregunta si el pulsador está oprimido
          goto      pulsa     ;si no lo está continúa revisándolo
          call      retardo   ;si está oprimido retarda 100 milisegundos
          btfsc     ptob,0    ;para comprobar
          goto      pulsa     ;si no lo está vuelve a revisar
          incf      conta     ;si lo confirma incrementa el contador
          movf      conta,w   ;carga el registro W con el valor del conteo
          movwf     eedata    ;el dato del conteo lo guarda en memoria
          call      escrib    ;para recuperarlo en caso de un apagón
          movf      conta,w
          xorlw     0ah       ;hace operación xor para ver si es igual a 0ah
          btfss     status,z  ;prueba si el contador llegó a 0ah (diez)
          goto      ciclo     ;si no es igual se incrementa normalmente

          goto      ini2      ;
          end                 ;
;=====================================================================
;         Fusibles de programación
;         Osc                 XT
;         Watchdog            OFF
;         Code protect        OFF
;         Power-Up-Timer      ON
;         Micro.              PIC16F84
;=====================================================================
```

Figura 2.54. Programa que utiliza la memoria EEPROM de datos

Proyecto N°8: Control de un motor paso a paso

Por sus características y precisión de movimientos, los motores de paso se constituyen en un elemento muy valioso a la hora de diseñar un sistema de control o una máquina especializada. Además, dado que las señales necesarias para controlar esta clase de motores son de naturaleza digital, éstos se pueden conectar fácilmente a sistemas de ese tipo.

En este proyecto emplearemos un microcontrolador PIC como elemento principal para guiar la posición de un motor de esta clase. El proyecto se ha dividido en varias partes. Primero, se hace una breve descripción del tipo de motor que se utilizará, luego se muestra la forma de conectarlo al microcontrolador para conseguir movimientos de paso completo y por último se hace un ejercicio con movimientos de medio paso.

Motores paso a paso

Para el ejercicio se utiliza un motor de 4 bobinas. Este tipo de motores se identifica porque tienen 5, 6 u 8 cables. En el primer caso existe un cable que es común a todos los demás, para el de 6 cables se tiene un común para cada pareja de bobinas y en el de 8 cables cada bobina es independiente. Para el proyecto empleamos uno de 8 cables. En la figura 2.55 se muestra su estructura, incluyendo los colores de los cables (que son estándares).

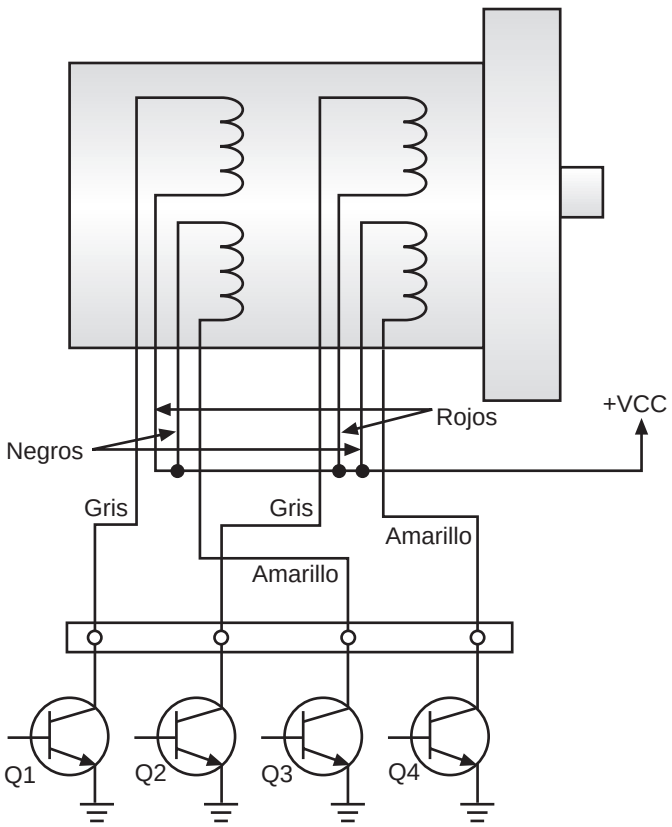


Figura 2.55. Conexión de los transistores que manejan el motor

Este tipo de motores tiene la ventaja de operar con una sola fuente, mientras que los motores de dos bobinas requieren polaridad positiva y negativa, haciéndose necesario utilizar circuitos en puente. En la figura 2.55 también se muestra la forma de conectar el motor de 8 cables. Debe notarse que los cables rojos y negros se unen para conectarlos a la fuente positiva. De aquí se puede deducir como sería la conexión para 6 y 5 cables. El transistor que se utiliza para activar las bobinas es el típico TIP122, que es un Darlington con protección para el manejo de cargas inductivas (diodo interno). Nótese que se ha invertido el orden de las bobinas de los transistores Q2 y Q3.

El movimiento del motor se consigue energizando las bobinas en un orden determinado. Esto se puede hacer de dos formas: La primera consiste en activar una sola bobina a la vez (manejo por ola), la segunda es el manejo de dos fases o bobinas al mismo tiempo. Con este último método se puede conseguir mayor torque, por eso es el método que empleamos en estos ejemplos. La figura 2.56 muestra los diagramas de tiempo de los dos métodos anteriores. De ahora en adelante se supondrá que el motor se ha conectado de la forma que se muestra en la figura 2.55; esto se hace con el fin de no repetir la misma gráfica en cada ejercicio.

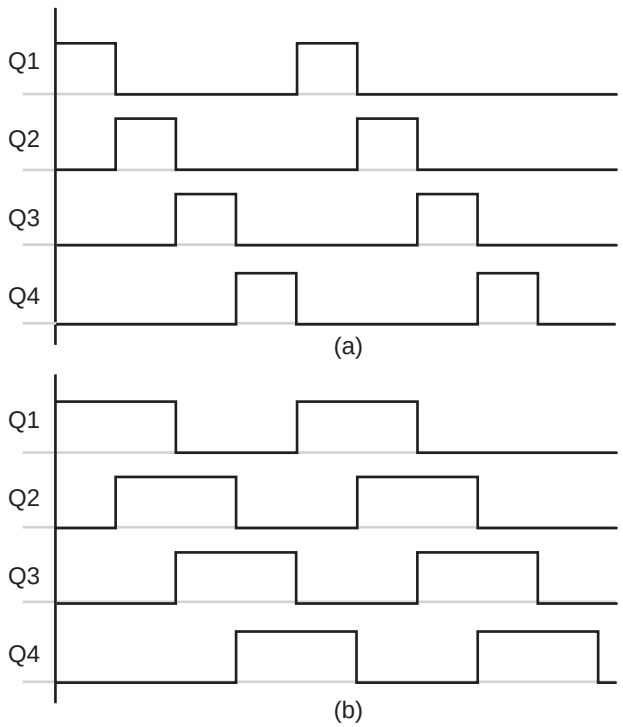
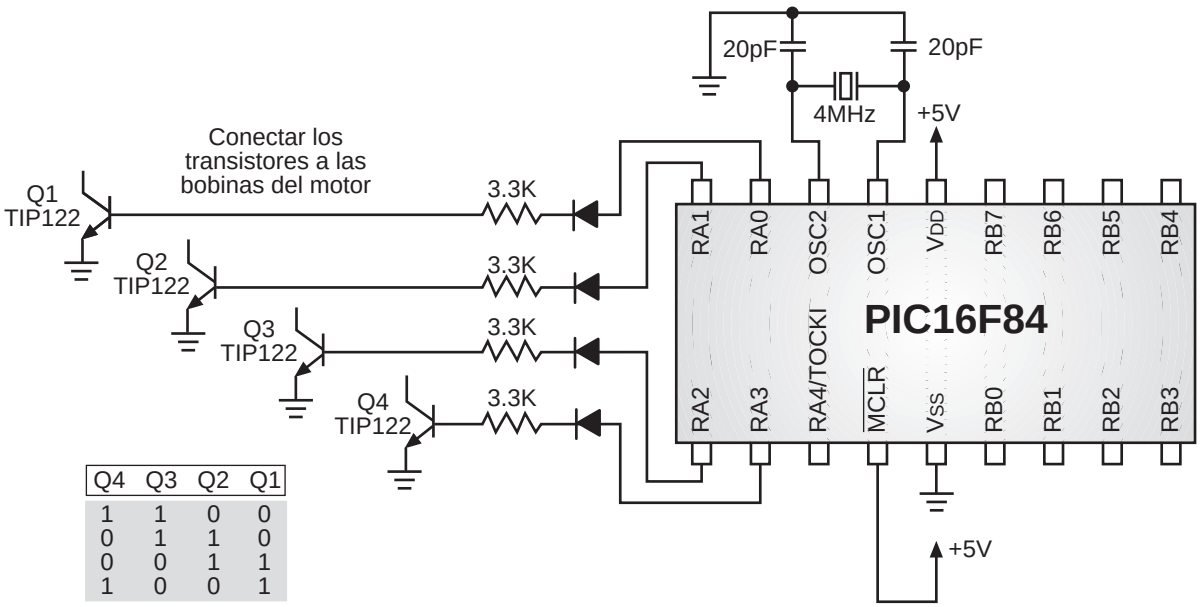


Figura 2.56. Secuencias para manejar las bobinas del motor de paso. **a)** una fase, **b)** dos fases

Control del motor a pasos completos

El motor que se escogió para el ejercicio tiene un paso de 7.5º, es decir que en un círculo completo (360º) tiene 48 posiciones de reposo. Cada uno de los pasos se logra cuando se encienden las bobinas correspondientes según la secuencia. También se debe aclarar que el punto común de las bobinas del motor se debe conectar a +5V, ya que este es el dato suministrado por el fabricante en la placa de datos del mismo.

Para controlar el motor se utiliza el microcontrolador PIC16F84, el cual está empleando un cristal de 4 MHz y utiliza los cuatro pines bajos del puerto A (RA0-RA3) como salidas, para activar o desactivar los transistores de potencia TIP122 que manejan cada una de las bobinas. En la figura 2.57 se muestra el diagrama esquemático correspondiente, el cual incluye una tabla con la secuencia que se debe seguir para conseguir que el motor gire.



Q4	Q3	Q2	Q1
1	1	0	0
0	1	1	0
0	0	1	1
1	0	0	1

Figura 2.57. Diagrama del circuito para mover el motor a paso completo

Como se había mencionado anteriormente, el control del motor se hará con dos bobinas energizadas al mismo tiempo. Viendo la tabla de la secuencia se puede deducir que con simples rotaciones de un registro se puede conseguir el objetivo (nótese que la tabla tiene cuatro valores diferentes). Cada vez que se encienden las dos bobinas correspondientes a un valor de la tabla, este se mueve un paso. Por lo tanto, dado que el motor es de 48 pasos por vuelta, se debe repetir los valores de la tabla 12 veces para conseguir que el eje del motor de un giro completo.

Si se desea que el motor gire en sentido contrario al que se logró con la secuencia anterior, sólo se debe invertir la secuencia, es decir, leer la tabla en sentido inverso (de abajo hacia arriba). Cuando el motor no está en movimiento se debe garantizar que todas las bobinas estén desenergizadas. Esto se hace con el fin de evitar posibles daños del motor.

El primer ejercicio que vamos a implementar consiste en hacer que el motor de un giro completo en un sentido, permanezca quieto un momento y luego gire en sentido contrario, para repetir el mismo ciclo. Dado que la secuencia es simple, el programa del microcontrolador también lo es. La figura 2.58 muestra el listado completo.

```
;ESTE PROGRAMA PERMITE CONTROLAR EL GIRO DE UN MOTOR DE
;PASO, A PASOS COMPLETOS
;          ******  PIC16F84   ********
; ******  WDT = OFF, OSC = XT  ******
TMR0       EQU          1H              ;CONTADOR DE TIEMPO REAL
PC         EQU          2H              ;CONTADOR DE PROGRAMA
STATUS     EQU          3H              ;REGISTRO DE ESTADOS
PTOA       EQU          5H              ;PUERTOS
PTOB       EQU          6H
R0D        EQU          0DH             ;
R10        EQU          10H
R11        EQU          11H
Z          EQU          2H              ;BANDERA DE CERO
C          EQU          0H              ;BANDERA DE CARRY
W          EQU          0H              ;PARA ALMACENAR EN W
R          EQU          1H              ;PARA ALMACENAR EN EL REGISTRO

           ORG          00              ;Vector de reset
           GOTO         INICIO          ;Inicia programa principal
           ORG           05H
RETARDO    CLRF         TMR0            ;Retardo de 8.192 ms
           NOP                          ;empleando el TMR0
RETAR1     BTFSS        TMR0,7
           GOTO         RETAR1
           RETLW        0

RETARD2    MOVLW        38H             ;Retardo auxiliar
           MOVWF        R0D             ;del giro del motor
DECR2      CALL         RETARDO
           CALL         RETARDO
           DECFSZ       R0D,R
           GOTO         DECR2
           RETLW        0

PASOAD     BCF          STATUS,C        ;Esta rutina hace mover el
           RRF          R10,R           ;motor solo un paso en un
           BTFSC        STATUS,C        ;sentido, lo  que hace es
           BSF          R10,3           ;rotar los dos unos lógicos que
           MOVF         R10,W           ;encienden dos transistores
           MOVWF        PTOA            ;al tiempo
           CALL         RETARDO
           CLRF         PTOA
           RETLW        0

PASOAT     BCF          STATUS,C        ;Esta  rutina  hace mover el
           RLF          R10,R           ;motor un paso, en sentido
           BTFSC        R10,4           ;contrario a la anterior
           BSF          R10,0
           MOVF         R10,W
           MOVWF        PTOA
           CALL         RETARDO
           CLRF         PTOA
           RETLW        0

; ***********  PROGRAMA PRINCIPAL  *******

INICIO     MOVLW        00H             ;Los puertos se programan de
           TRIS         PTOA            ;acuerdo al circuito
           MOVLW        0FFH
           TRIS         PTOB
           MOVLW        0C5H            ;se programa el TMR0 con
           OPTION                       ;preescala de 64
           CLRF         PTOA
           CLRF         R10
           BSF          R10,2           ;Estos son los unos que rotan
           BSF          R10,1           ;en el registro R10

BEGIN      MOVLW        D'48'           ;En este ciclo el motor hace un
           MOVWF        R11             ;giro completo
```

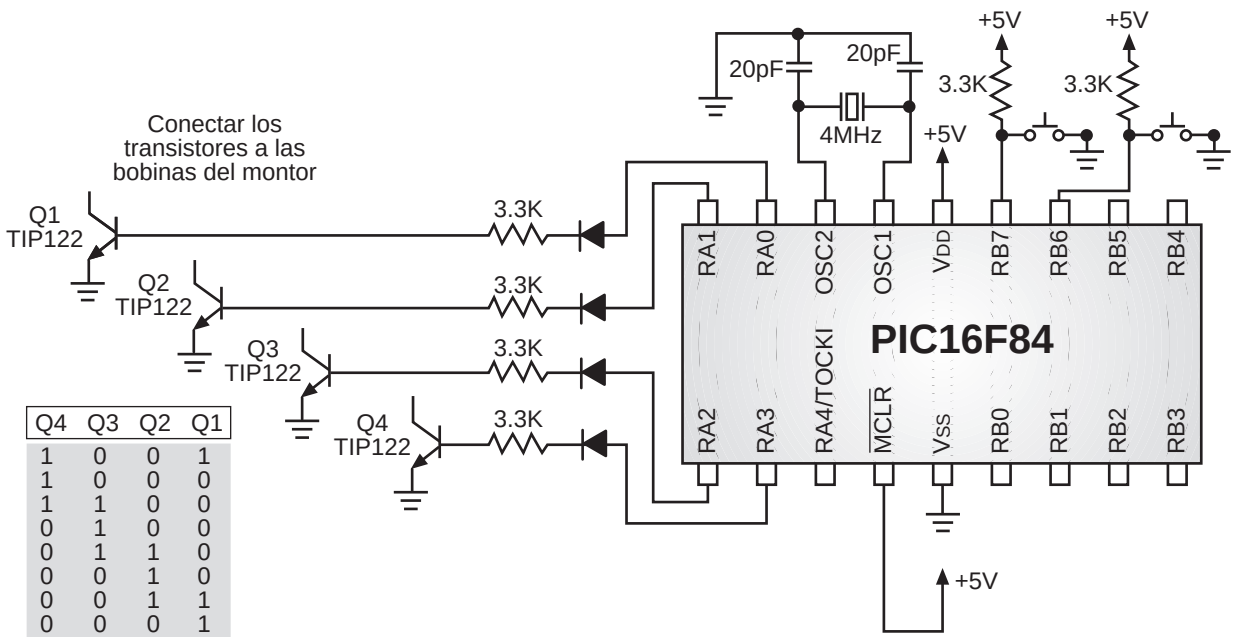
```
VUEL    CALL    PASOAD      ;se llama 48 veces la rutina
        DECFSZ  R11,R       ;que hace mover el motor 1 paso
        GOTO    VUEL
        CALL    RETARD2
        CALL    RETARD2
        MOVLW   D'48'
        MOVWF   R11
VUEL2   CALL    PASOAT      ;se llama 48 veces la rutina
        DECFSZ  R11,R       ;que lo mueve un paso en
        GOTO    VUEL2       ;sentido contrario
        CALL    RETARD2
        CALL    RETARD2
        GOTO    BEGIN
        END
```

Figura 2.58. Programa para controlar el motor a pasos completos

Las rutinas llamadas PASOAD y PASOAT se encargan de hacer la rotación de las señales que controlan la activación y desactivación de las bobinas del motor. El estado de las rotaciones se conserva en el registro de memoria RAM R10 y se pasa al puerto A cada vez que se desea mover el motor un paso. En el bloque principal del programa solamente se requiere cargar un contador con el número de pasos deseado (en este caso 48) para llamar la rutina correspondiente igual número de veces. La rutina de retardo está basada en el temporizador interno del microcontrolador para garantizar su precisión.

Control del motor a 1/2 paso

En el primer ejercicio se hizo girar el motor avanzando en pasos completos. Ahora la idea es hacerlo girar avanzando medio paso cada vez que se desee. Con esto podremos conseguir que en una vuelta completa del eje del motor, se tengan 96 intervalos diferentes, o sea que el eje se desplaza 3.75º en cada paso. Esta clase de movimientos de precisión hacen del motor de paso un elemento muy útil.



Q4	Q3	Q2	Q1
1	0	0	1
1	0	0	0
1	1	0	0
0	1	0	0
0	1	1	0
0	0	1	0
0	0	1	1
0	0	0	1

Figura 2.59. Circuito para controlar el motor a 1/2 paso

Para este ejercicio la idea es hacer que el motor gire en el sentido que se le indique oprimiendo uno de los dos pulsadores dispuestos para tal fin. El motor debe moverse tanto tiempo como esté oprimido el interruptor correspondiente. Para este ejercicio empleamos el mismo circuito del ejemplo anterior, pero con unas pequeñas variaciones. En la figura 2.59 se muestra el diagrama esquemático del circuito de control.

En los pines RB7 y RB6 del microcontrolador se han dispuesto los interruptores pulsadores que se encargan de controlar el sentido y la duración del movimiento. Además, en la gráfica se incluye la tabla con la secuencia que se requiere para que el motor pueda moverse a 1/2 paso. Como se ve, ahora la tabla tiene ocho valores diferentes. El microcontrolador se debe encargar de hacer la lectura de dichos valores en orden ascendente o descendente para obtener el valor que debe entregar por el puerto A para manejar las bobinas del motor.

El software que se graba en el microcontrolador aparece en la figura 2.60. Para este ejemplo se han modificado las rutinas debido a que la secuencia no es una simple rotación de las bobinas. Los valores que debe entregar el microcontrolador se han escrito en una tabla, para que la estructura del programa sea la misma en el momento que se desee trabajar a 1/4 de paso e inclusive a 1/8 de paso. En estos dos últimos casos sólo se debe cambiar los valores de la tabla y el límite máximo del contador que se encarga de leerla. Como ejercicio el lector puede deducir la secuencia que se emplearía en movimientos a 1/4 y 1/8 de paso.

```
;ESTE PROGRAMA PERMITE CONTROLAR EL GIRO DE UN MOTOR DE PASO, A 1/2 PASO
;     ******  PIC16F84   ********
;  ******  WDT = OFF, OSC = XT, CP = OFF   ******


TMRO      EQU         1H              ;CONTADOR DE TIEMPO REAL
PC        EQU         2H              ;CONTADOR DE PROGRAMA
STATUS    EQU         3H              ;REGISTRO DE ESTADOS
PTOA      EQU         5H              ;PUERTOS
PTOB      EQU         6H
R0D       EQU         0DH
R10       EQU         10H
Z         EQU         2H              ;BANDERA DE CERO
C         EQU         0H              ;BANDERA DE CARRY
W         EQU         0H              ;PARA ALMACENAR EN W
R         EQU         1H              ;PARA ALMACENAR EN EL REGISTRO
ADE       EQU         7               ;BITS DEL PTOB
ATRA      EQU         6

;         *********  PROGRAMA PRINCIPAL  *********

          ORG         00              ;Vector de reset
          GOTO        INICIO          ;Va a iniciar programa principal
          ORG         05H

RETARDO   CLRF        TMRO            ;Retardo de 2.048 ms
          NOP                         ;usando el TMRO
RETAR1    BTFSS       TMRO,5
          GOTO        RETAR1
          RETLW       0

PASOAD    INCF        R10,R           ;Esta rutina mueve el motor
          MOVF        R10,W           ;medio paso en un sentido
          CALL        SECUEN          ;está basada en una tabla
          MOVWF       PTOA            ;que contiene el estado de
```

```
          CALL      RETARDO         ;activación de los transistores
          CLRF      PTOA
          MOVLW     07H             ;cuando se termine la ultima
          XORWF     R10,W           ;posición de la tabla se
          BTFSS     STATUS,Z        ;empieza nuevamente
          GOTO      SALIR
          MOVLW     0FFH
          MOVWF     R10
SALIR     RETLW     0

PASOAT    DECF      R10,R           ;esta rutina mueve el motor en
          MOVF      R10,W           ;sentido contrario al anterior
          CALL      SECUEN          ;está basada en la misma tabla
          MOVWF     PTOA            ;que controla las salidas
          CALL      RETARDO
          CLRF      PTOA
          MOVLW     0FFH            ;cuando se termina la tabla
          ANDWF     R10,W           ;vuelve a empezar nuevamente
          BTFSS     STATUS,Z
          GOTO      SALE
          MOVLW     08H
          MOVWF     R10
SALE      RETLW     0

SECUEN    ADDWF     PC,R             ;Esta tabla contiene el
          RETLW     B'00000100'     ;estado de las salidas
          RETLW     B'00000110'
          RETLW     B'00000010'     ;Si se desea trabajar
          RETLW     B'00000011'     ;a 1/4 de paso sólo se debe
          RETLW     B'00000001'     ;cambiar la tabla
          RETLW     B'00001001'
          RETLW     B'00001000'
          RETLW     B'00001100'
          RETLW     0

INICIO    MOVLW     00H             ;programación de puertos
          TRIS      PTOA            ;según el circuito
          MOVLW     0FFH
          TRIS      PTOB
          MOVLW     0C5H            ;se programa preescala en 64
          OPTION                    ;para el TMRO
          CLRF      PTOA
          MOVLW     03H             ;se inicializa la secuencia
          MOVWF     R10             ;en algún estado

PRUE1     BTFSC     PTOB,ADE        ;si se oprime uno de los
          GOTO      PRUE2           ;pulsadores se llama la rutina
          CALL      PASOAD          ;que mueve el motor medio paso
          CLRF      PTOA
          GOTO      PRUE1

PRUE2     BTFSC     PTOB,ATRA       ;si se oprime el otro pulsador
          GOTO      PRUE1           ;el motor gira en sentido
          CALL      PASOAT          ;opuesto
          CLRF      PTOA
          GOTO      PRUE2

          END
```

Figura 2.60. Programa de control del motor a 1/2 paso

Las rutinas PASOAD y PASOAT se encargan de incrementar o decrementar el contador que sirve como puntero de la tabla. Además, según el sentido en que esté girando el motor, las rutinas hacen la verificación correspondiente cuando llega al final de la tabla, para volver a iniciar la lectura de la misma. La rutina SECUEN

contiene los valores que se deben entregar por el puerto del microcontrolador para energizar las bobinas correspondientes. Esta tabla es la que se debe modificar para conseguir movimiento a 1/4 o 1/8 de paso.

Como parte del ejercicio, el estudiante podría insertar un retardo entre cada movimiento (llamando una subrutina con un retardo largo), para ver en detalle como se mueve el motor y comprobar que en un círculo completo se tienen 96 escalas o posiciones de reposo.