NT1007

3.7 INTERFACE A PANTALLAS MULTIPLEXADAS CON LOS PUERTOS DE ENTRADA Y SALIDA

MULTIPLEXIÓN DE PANTALLAS Y CONTROL DE CADA SEGMENTO DEL "DISPLAY"

Preparado por: Rangel Alvarado
Estudiante Graduando de Lic. en Ing. Electromecánica
Universidad Tecnológica de Panamá
Panamá, Panamá
"e-mail": issaiass@cwpanama.net
"web site": http://www.geocities.com/issaiass/

ÍNDICE

3.7.1 Introducción

Muchas veces no solo necesitamos visualizar información de manera numérica, sino también escrita, y de alguna forma, se puede transmitir esta información por "displays" de siete segmentos. Utilizando los puertos de E/S se puede manejar directamente los LEDs o segmentos del "Display".

Aclarando que se debe tener mucha precaución de la poca capacidad de manejo de corriente de salida de los puertos, se debe tener especial cuidado con el resistor que se elija, pues si excede la capacidad de manejo, puede arruinar el puerto.

Aplicaciones comunes de este tipo incluyen quizá el manejo de carteles de anuncios electrónicos de venta, o presentar información de la escala y los vuelos en un aeropuerto. A pesar que se pueden presentar caracteres, no todos son visibles en la pantalla, limitante de la cantidad de segmentos que posee.

La nota se basa en documentos anteriores, NT1005 y NT1006, por consiguiente, la explicación del funcionamiento se obvia y se pasa directamente al esquema de aplicación. La aplicación de esta nota, es similar a las de las notas NT1005 y NT1006 que utiliza el despliegue de la conversión de ADC por medio de 3 siete segmentos.

3.7.2 Materiales

1. Microcontrolador: JL3 / JK3 / GP32
2. Tarjeta de Desarrollo TD68HC908JL3 o similar y su microcontrolador
3. Plantilla de proyectos: “Breadboard” JE27, Jameco Part No: 20811
4. Fuente de Poder
5. Alambres AWG 20
6. Pelador de Alambres
7. “Kit” de Aficionado: Pinzas del “Kit”, Jameco Part No: 99629
8. Integrado: Decodificador de BCD a 7 Segmentos, Jameco Part No.: 47790
   Multiplexor o Selector de Canales, Jameco Part No.: 46607
9. “Display”: Visualizador de 7 Segmentos, Jameco Part No.: 24731, MAN71
10. Resistores: Resistencias de 7 × 150Ω ¼ Watt, Chocolate, Verde, Negro.
    3 × 2.2kΩ ¼ Watt, Rojo, Rojo, Naranja
11. Transistores PNP: 3 x 2N2907, Jameco Part No.: 38279

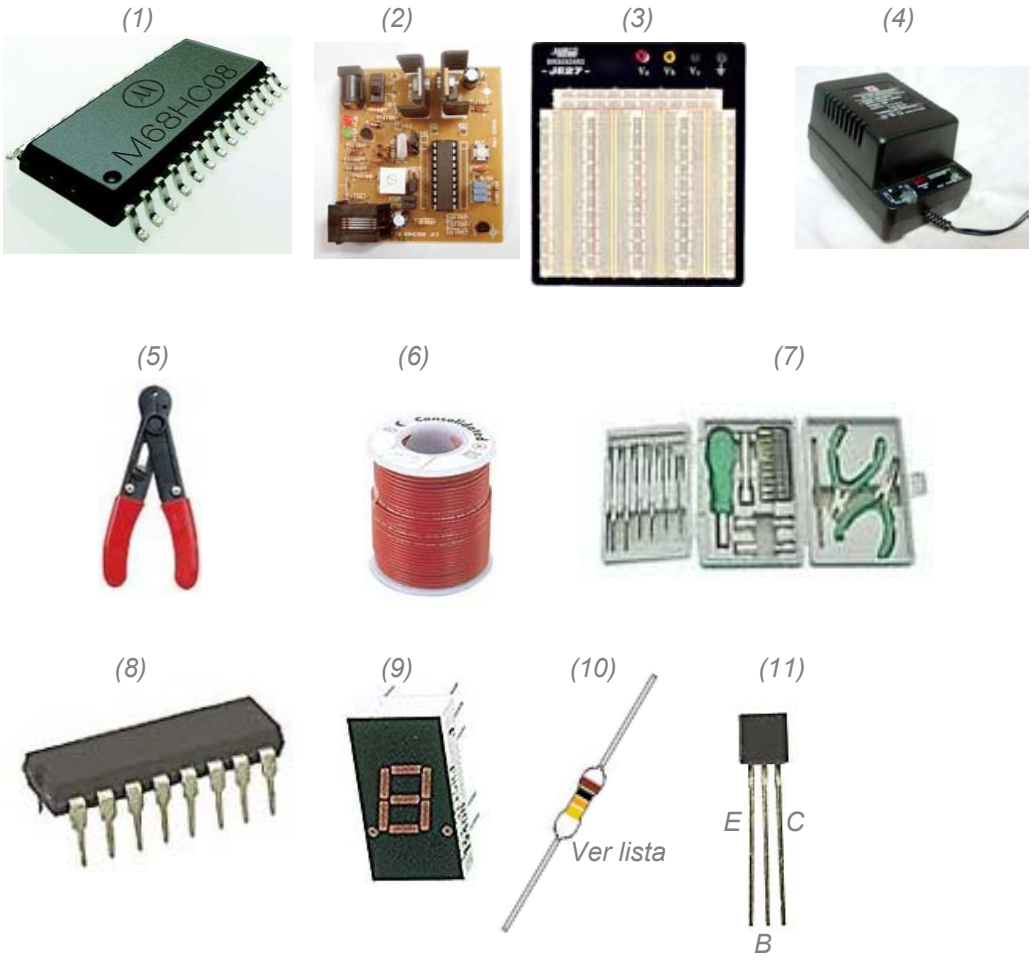

Figura 197. Listado de Materiales a Utilizar para Pantallas Multiplexadas y Puertos Generales de Entrada y Salida

3.7.3 Esquemático de Aplicación

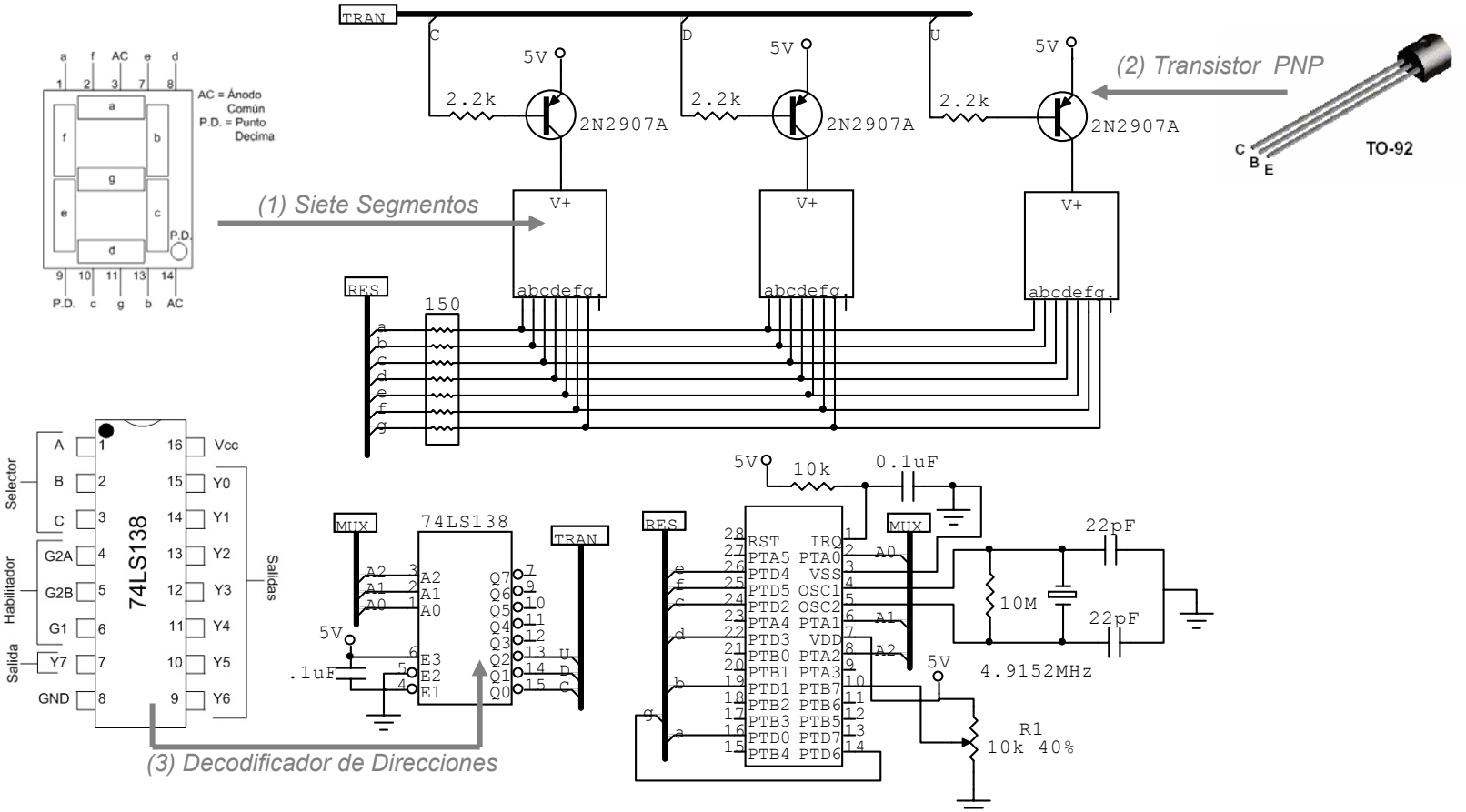

Figura 198. Esquema de Aplicación del Manejo de Siete Segmentos Multiplexado desde el puerto de Entrada y Salida. Para la multiplexión de pantallas se utiliza un decodificador de direcciones 74LS138, de manera tal este encendieda una pantalla a su vez. Las pantallas deben refrescarse a una rata de "n" × 60 Hz, donde "n" es el número de pantallas a multiplexar. Adicionalmente se controla este esquema para visualizar en tres siete segmentos la información arrojada por el convertidor analógico digital y se escribe la información directamente desde el puerto del microcontrolador hacia los segmentos.

3.7.4 Diagrama de Flujo

El siguiente programa utiliza un Decodificador de Direcciones (74LS138) y maneja directamente con los puertos de entrada/salida y decodifica la información recopilada del ADC para luego enviarla a tres visualizadores de siete segmentos. Si desea conocer en detalles como se hizo esta nota ud. necesita de los documentos básicos, NT0010, NT0011, NT0012; intermedios NT0101; avanzados NT1005 y NT1006; y Apéndices B, C, I, M y N.

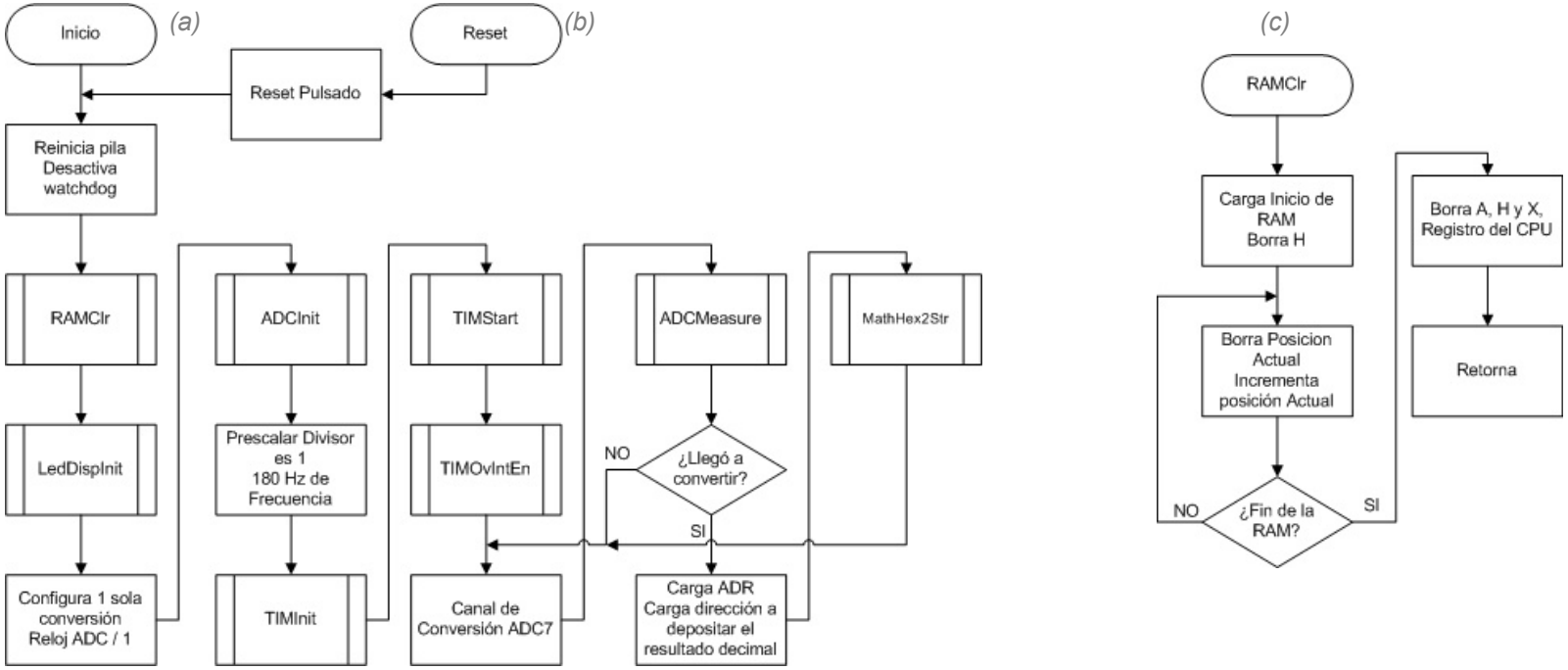

Figura 199. NT1007 – LED. (a) Programa Principal. Inicia el convertidor en el PTB7 y un temporizador a 180 Hz, el cual refrescará cada una de las tres pantallas a 60 Hz. Adicionalmente, se convierte el resultado del ADC a enteros de 0 a 9 que representan el valor escalado en decimal de la variación del potenciómetro, de 0 a 255 desplegado por los puertos del microcontrolador directamente. (b) Reinicio del Sistema. Al presionar el botón "Reset", sin importar lo que suceda de momento, el sistema saltará nuevamente al inicio. (c) RAMClr. Rutina que borra la memoria RAM y los registros del CPU.

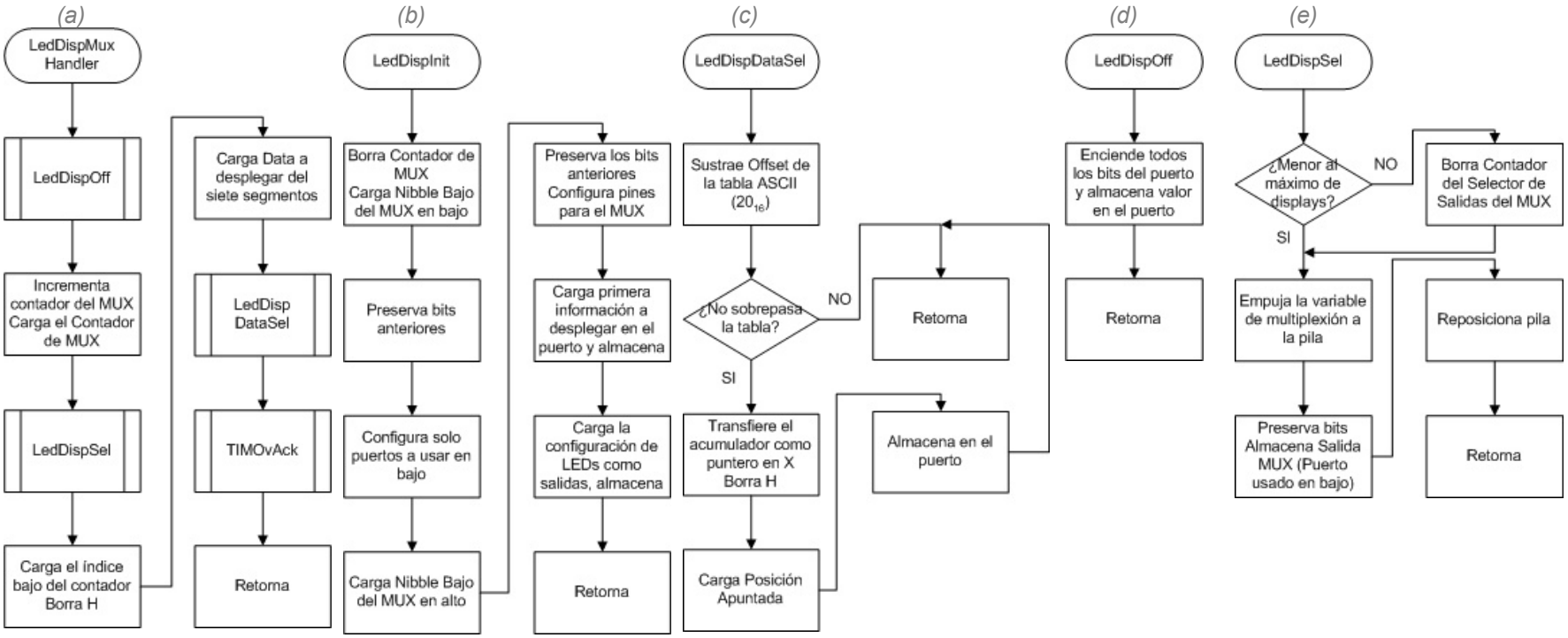

Figura 200. NT1007 – LED – Continuación. (a) LedDispMuxHandler. Es la rutina que refresca las pantallas, esta rutina debe estar administrada por el temporizador del sistema que realice un barrido a "n" × 60 Hz, en donde "n" es el número de pantallas. (b) LedDispInit. Inicializa el visualizador de siete segmentos configurando el microcontrolador a manejar el primer dígito (centenas) y los puertos para los segmentos en bajo. (c) LedDispDataSel. Selecciona la pantalla a encender y despliega la información en la siguiente pantalla a multiplexar. (d) LedDispOff. Apaga las pantallas. (e) LedDispSel. Selecciona la pantalla a encenderse.

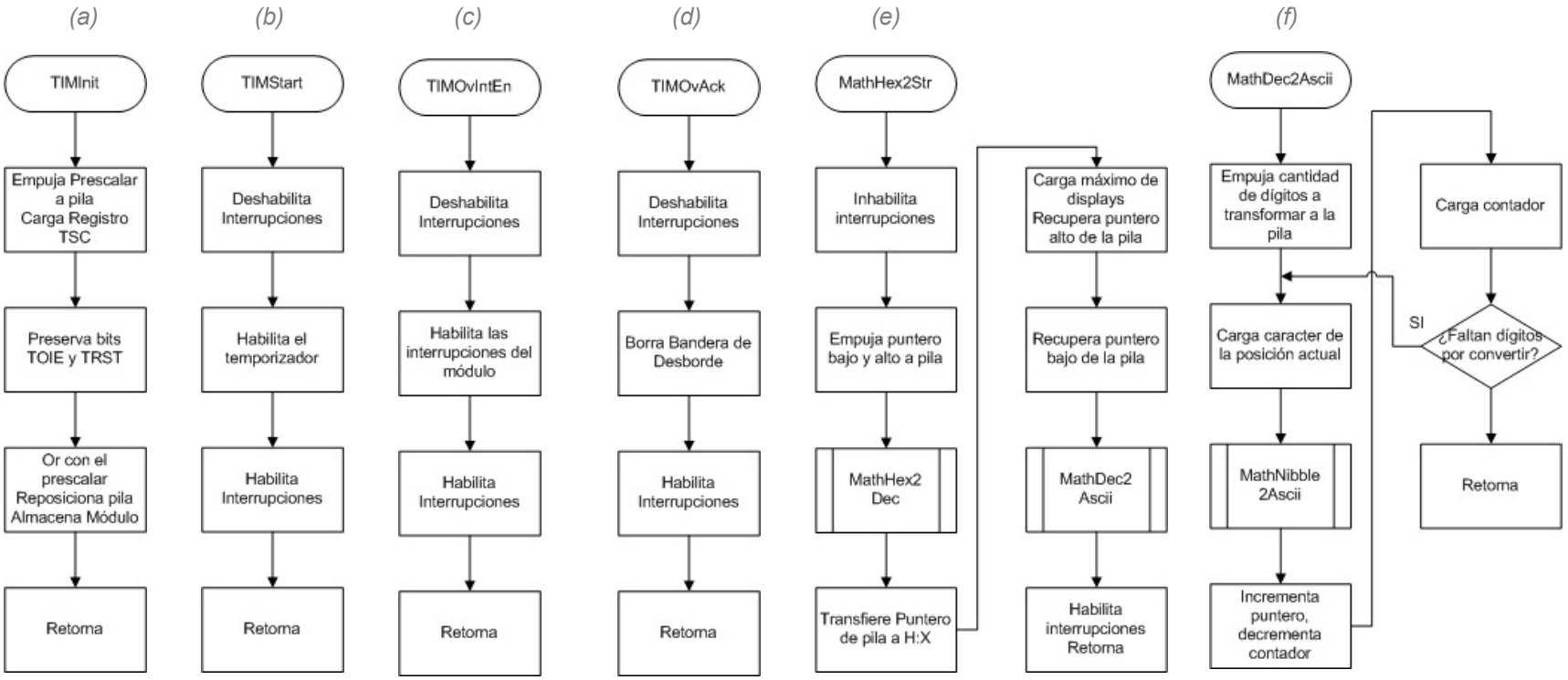

Figura 201. NT1007 – LED – Continuación. (a) TIMInit. Inicializa el temporizador dado el preescalar y el módulo del contador. (b) TIMStart. Inicia el conteo del temporizador. (c) TIMOvIntEn. Habilita las interrupciones de sobreflujo del temporizador. (d) TIMOvAck. Reconoce las interrupciones de sobreflujo del temporizador. (e) MathHex2Str. Transforma un número hexadecimal a una cadena de caracteres ASCII. (f) MathDec2Ascii. Transforma un número decimal a Ascii.

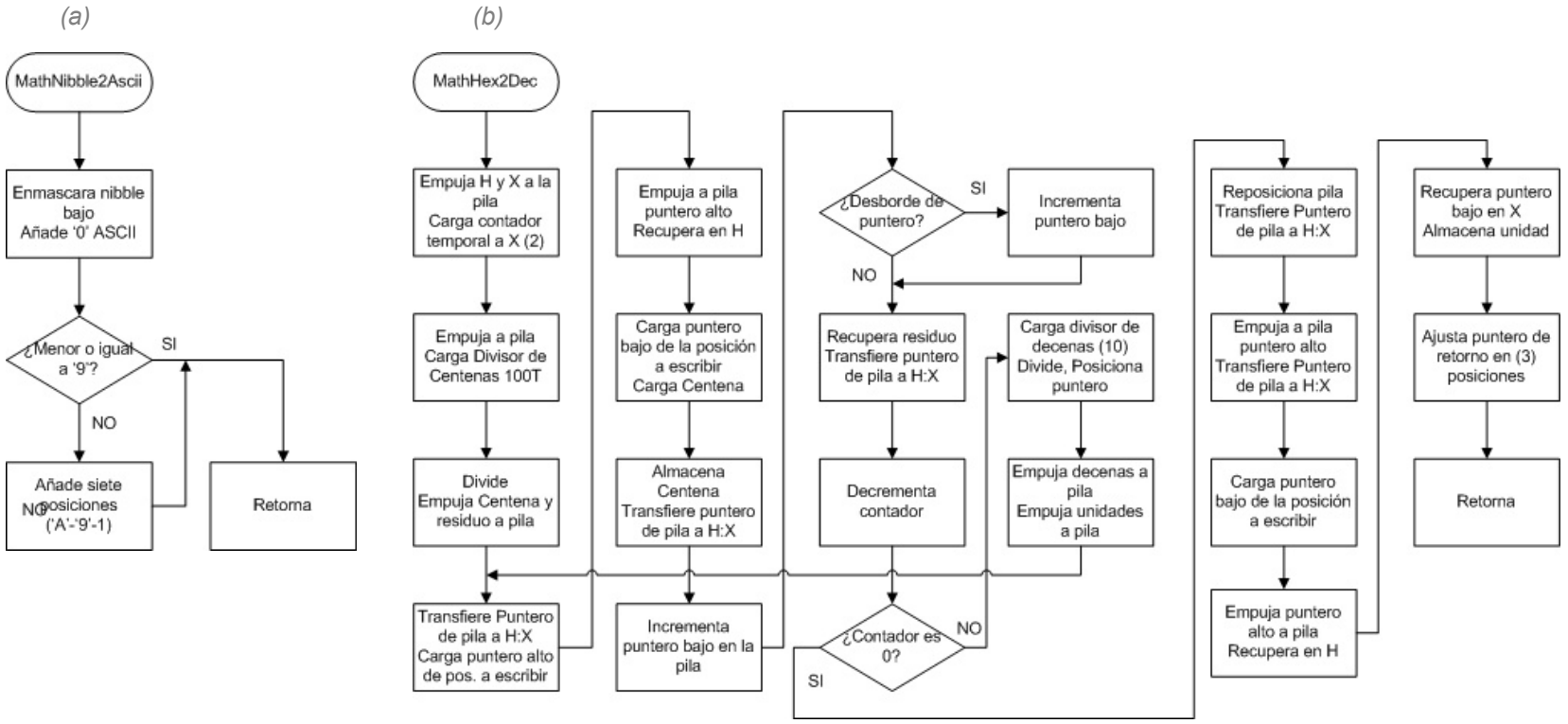

Figura 202. NT1007 – LED – Continuación. (a) MathNibble2Ascii. Transforma la parte baja de un caracter hexadecimal a ASCII. (b) MathHex2Dec. Transforma un caracter hexadecimal a su equivalente en tres dígitos decimales.

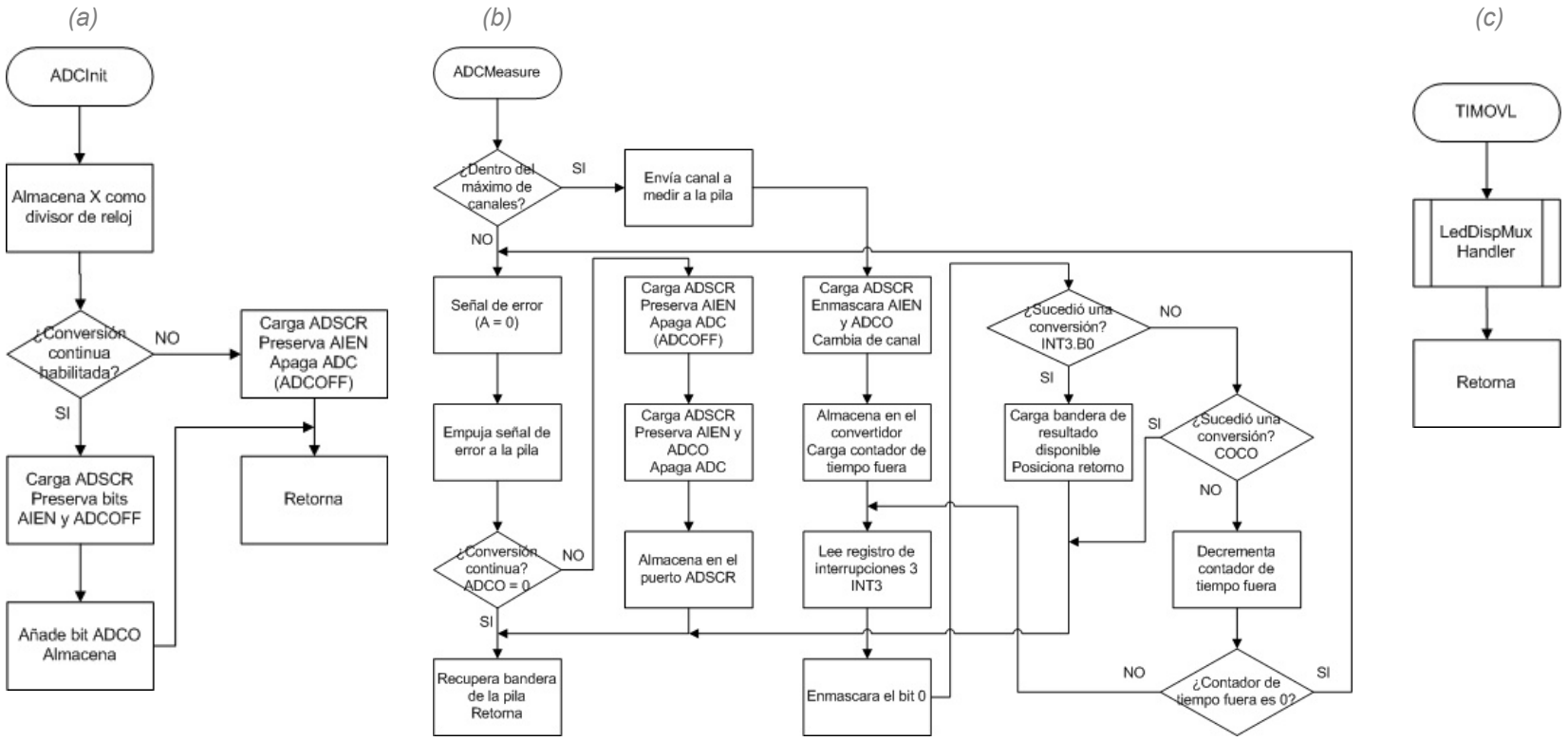

Figura 203 NT1007 – LED – Continuación. (a) ADCInit. Inicializa el convertidor analógico a digital dado el divisor del reloj y el tipo de conversión. (b) ADCMeasure. Mide un canal respectivo y retorna una bandera la cual simboliza una medición correcta o no. (c) TIMOVL. Subrutina de interrupción encargada de la multiplexión de los siete segmentos.

3.7.5 Código

Advertencia, del siguiente programa solo se listará las partes necesarias para que la rutina opere de manera apropiada, si desea ver el resto del contenido de un archivo cabecera incompleto, dirigirse al archivo comprimido. Para mejor entendimiento de la nota técnica, referirse a las notas técnicas básicas NT0010, NT0011, NT0012, NT0101, NT1005 y NT1006; además de los Apéndices B, C, I, M y N.

```
;=====================================================================
; ARCHIVO      : NT1007 - LED - 26 12 04.asm
; PROPÓSITO    : Adquiere una señal del ADC7 y la despliega en 3 siete seg-
;                mentos multiplexando la información, adicionalmente, existe
;                un temporizador que multiplexa los displays a una rata de
;                N*60 Hz, en donde N es el número de pantallas.
; NOTAS        : Ninguna
;
;
;
; REFERENCIA   : NT1007 - LED - 04 01 05.doc
;
; LENGUAJE     : IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   26  12  04   Creado.
;   04  01  05   Modificado.
;=====================================================================

;=========================================
;            Cabecera de Macros, Const. y Memoria
;=========================================
$include '\MAP\includes.equ'              ; Definiciones de usuario y mapa de memoria
```

```
;===========================================
; OBJETIVO     :  Inicio de Codif. del Ensam-
;                 blador en Memoria FLASH.
;===========================================
        org FLASH_START             ; Inicio Mem. FLASH

;===========================================
; OBJETIVO     :  Convierte el valor Hexadeci
;                 mal resultado de la conver-
;                 sión en Decimal y despliega
;                 en 3 siete segmentos.
;===========================================
START
        rsp                         ; Inic.Stack = $00ff
        bset BIT0,CONFIG1           ; Desactiva Watchdog
        jsr RAMClr                  ; Borra RAM y registros
        jsr LedDispInit             ; Inicializa pantalla
        clra                        ; Una sola conversión
        clrx                        ; ADICLK = XTAL/(2^(0+2))
        jsr ADCInit                 ; Inicializa ADC
        clra                        ; DIV1
        ldhx #$1AAA                 ; Desborde a 3*60 = 180 Hz
        jsr TIMInit                 ; Inicializa TIM
        jsr TIMStart                ; Inicia temporizado
        jsr TIMOvIntEn              ; Habilita interrupciones
LEDAGAIN0101.AA lda #7T             ; ADC7 = PTB7
        jsr ADCMeasure              ; Mide el ADC
        beq LEDAGAIN0101.AA         ; ¿No hubo medición?, salir
        lda ADR                     ; Carga Resultado de conversión
        ldhx #LedDispData           ; Carga lugar a transformar
        jsr MathHex2Str             ; Convierte a Cadena
        bra LEDAGAIN0101.AA         ; Captura una nueva conversión

;===========================================
;              Declaración y definición de funciones
;===========================================
$include '\FUNCTIONS\INCLUDES.inc' ; Incluye Funciones

;===========================================
;            Declaración y Definición de interrupciones
;===========================================
$include '\INTERRUPTS\interrupt.inc'        ; Incluye interrupciones del
microcontrolador
```

Listado 47. NT1007 – LED. El programa inicia el temporizador a 180 Hz y captura las conversiones del ADC, PTB7, para desplegarlo en tres (3) siete segmentos por medio de los puertos de entrada y salida encendiendo cada segmento.

```
;=================================================================
; ARCHIVO      : LED.inc
; PROPÓSITO    : Librería para utilidad de Siete segmentos.
; LENGUAJE     : IN-LINE ASSEMBLER
; NOTAS        : UTILIZA UNA ESTRUCTURA EN RAM QUE UBICA LA
;                INFORMACIÓN A DESPLEGAR EN LOS SIETE SEGMENTOS.
;                UTILIZAR LA RUTINA "LEDMUXHANDLER" CON UN
;                TEMPORIZADOR A N_DISPLAYS * 60 Hz.
; ---------------------------------------------------------------------
; HISTORIAL
;  DD MM AA
;  26  05  03   Creado.
;  25  12  04   Modificado.
;=================================================================
; INICIALIZACIÓN
;        : Configurar...
;                1 - El nibble para el multiplexor:
;                        $SET, Selecciona la configuración y uso del nibble alto
;                        $SETNOT, Selecciona el nibble bajo
;                2 - Las características de operación del Multiplexor
;                        2.1 - Puerto de Multiplexión = LED_MUX_PORT
;                        2.2 - Registro del Puerto del Multiplexor = LED_MUX_DDR
;                        2.3 - Máximo de caracteres a multiplexar = LED_MAX_DISP
;                3 - La estructura de despliegue de información Ubicando la
;                        Dirección en la memoria RAM de:
;                        3.1 - Variable de control del Multiplexor = LedCtr
;
; ADVERTENCIA
;                : La estructura en RAM, depende de la ubicación de LedMuxCtr
;                  y usa "el número de caracteres a multiplexar" + 2, definida
;                  por la ubicación (es decir, empezando) por la variable
;                  LedMuxCtr.
;                  LA RUTINA "LEDMUXHANDLER", DEBE IR ADMINISTRADA POR
;                  UN TEMPORIZADOR INICIADO A "N * 60 Hz", EN DONDE "N" ES
;                  LA CANTIDAD DE 7 SEGMENTOS A UTILIZAR.
;
;                 a
;               -----        +-----------------------------------------+
;             f /    /       | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0|
;              / g  / b      -------------------------------------------
;             /-----/        |P.D.| g |  f | e |  d | c |  b |  a |
;           e /    / c       +-----------------------------------------+
;            /    /
;           ------  o <-  P.D.
;             d
;
;
;=================================================================
```

```
;$SET    MUX_NIBBLE_HI                  ; Configura para nibble alto del puerto
                                        ; de MUX
$SETNOT MUX_NIBBLE_HI                   ; Configura para nibble bajo del puerto
                                        ; de MUX

;=============================================
;                   Constantes & Macros
;=============================================
;=============================================
;           DEFINA PUERTO DE MULTIPLEXIÓN
;=============================================
LED_MUX_PORT    equ PORTA               ; Puerto del Multiplexor
LED_MUX_DDR     equ DDRA                ; Registro del Puerto del Multiplexor
LED_MAX_DISP    equ 3T                  ; Máximo de caracteres a multiplexar

;=============================================
;       DEFINA PUERTO DE SIETE SEGMENTOS
;=============================================
LED_SSEG_PORT   equ PORTD               ; Puerto del Decodificador
LED_SSEG_DDR    equ DDRD                ; Registro del Puerto del Decodificador

;=============================================
; DEFINA LUGAR DE LA ESTRUCTURA DE DESPLIEGUE
;=============================================
LedMuxCtr       equ $80                 ; Variable de control del Multiplexor
LedDispData     equ LedMuxCtr+1         ; Inicio de data a desplegar en 7
segmentos
LedDispDataTop  equ {LedDispData + LED_MAX_DISP}
                                        ; Fin de la data a desplegar en 7
                                        ; segmentos

;=============================================
;           CONSTANTES DE MULTIPLEXOR
;=============================================
LED_MUXNIBHIOFF equ $8F                 ; Nibble Alto en estado bajo
LED_MUXNIBLOOFF equ $F8                 ; Nibble Bajo en estado alto
LED_MUXNIBHIOUT equ $70                 ; Nibble Alto son salidas
LED_MUXNIBLOOUT equ $07                 ; Nibble Bajo son salidas

LED_ALL_OUTS    equ $FF                 ; Todos son salidas (para los
                                        ; segmentos)
LED_SSEG_OFF    equ $FF                 ; Todos los siete segmentos apagados
```

```
;===========================================
; LEDDISPINIT :  Inicializa el módulo de 7
;                Segmentos, Leds.
; OBJETIVO    :  Inicializa el módulo de
;                multiplexión de 7 segmentos,
;                leds
; ENTRADA     :  Ninguna
; SALIDA      :  Ninguna
; REGISTROS
; AFECTADOS   :  ACCA, LED_MUX_PORT,
;                LED_MUX_DDR, LED_SSEG_PORT,
;                LED_SSEG_DDR, LedMuxCtr(STRU)
; NOTAS       :  Asegúrese de inicializar
;                correctamente el número de
;                displays a multiplexar.
;===========================================
LedDispInit
        clr LedMuxCtr                   ; Borra contador del multiplexor
$IF  MUX_NIBBLE_HI
        lda #LED_MUXNIBHIOFF            ; Estado de nibble alto en bajo
$ELSEIF
        lda #LED_MUXNIBLOOFF            ; Estado de nibble bajo en bajo
$ENDIF
        and LED_MUX_PORT                ; Preservar bits
        sta LED_MUX_PORT                ; Puertos a usar en bajo
$IF  MUX_NIBBLE_HI
        lda #LED_MUXNIBHIOUT            ; Nibble alto como salidas
$ELSEIF
        lda #LED_MUXNIBLOOUT            ; Nibble bajo como salidas
$ENDIF
        ora LED_MUX_DDR                 ; Preserva bits
        sta LED_MUX_DDR                 ; Puertos a usar son salidas
        lda LedDispData                 ; carga la información a desplegar
        sta LED_SSEG_PORT               ; Configura los bits del puerto a usar
                                        ; en bajo
        lda #LED_ALL_OUTS               ; Configura todos como salida
        sta LED_SSEG_DDR                ; Puerto de Segmentos es salida
        rts                             ; SI, Salir
```

```
;=========================================
; LEDDISPOFF :  Apaga el display de siete
;               Segmentos
; OBJETIVO   :  Envía la data $0F al display
;               para apagar todos los seg-
;               mentos.
; ENTRADA    :  Ninguna
; SALIDA     :  Ninguna
; REGISTROS
; AFECTADOS  :  ACCA, LED_SSEG_PORT
;=========================================
LedDispOff
        lda #LED_SSEG_OFF           ; A apaga los segmentos
        sta LED_SSEG_PORT           ; Almacena en el puerto
        rts                         ; Apaga los displays

;=========================================
; LEDDISPSEL :  Selecciona el Display a ser
;               encendido
; OBJETIVO   :  Enciende el display actual
;               dado el número del display
; ENTRADA    :  ACCA    contiene el número
;               del canal actual
;               a multiplexarse
; SALIDA     :  Ninguna
; REGISTROS
; AFECTADOS  :  ACCA, LedMuxCtr (RAM) , SP,
;               LED_MUX_PORT
;=========================================
LedDispSel
        cmp #LED_MAX_DISP           ; Compara con el máximo de displays
                                    ; a multiplexar
        blo LEDCTRCLR0102.B         ; ¿Faltan por multiplexar?, SI, Saltar
        clr LedMuxCtr               ; No, reiniciar multiplexión
        clra                        ; Limpia contador
LEDCTRCLR0102.B
$IF  MUX_NIBBLE_HI
        nsa                         ; Cambia nibble bajo por alto
$ENDIF
        psha                        ; Empuja variable de multiplexión
$IF  MUX_NIBBLE_HI
        lda #LED_MUXNIBHIOFF        ; Nibble alto apagado
$ELSEIF
        lda #LED_MUXNIBLOOFF        ; Nibble bajo apagado
$ENDIF
        and LED_MUX_PORT            ; Preserva bits
        ora 1,SP                    ; Suma contenido del puerto
        sta LED_MUX_PORT            ; Puertos a usar en bajo
        pula                        ; Reposiciona pila
        rts                         ; Retorna
```

```
;===========================================
; LEDDISPDATASEL
;                : Selecciona data a desplegar
; OBJETIVO       : Despliega la información exis
;                  tente en la estructura de
;                  RAM.
; ENTRADA        : ACCA    es el índice de la
;                  posición en la tabla
;                  a desplegar
; SALIDA         : Ninguna
; REGISTROS
; AFECTADOS      : ACCA, LED_SSEG_PORT
;===========================================
LEDDispDataSel
        sub #$20                        ; Sustrae el offset
        cmp #{LedSsegAsciiTblTop - LedSsegAsciiTbl}
                                        ; Compara con el máximo de pantallas
        bhi LEDOUT0102.C                ; ¿Mayor a los segmentos?, SI, Salir
        tax                             ; Transfiere como índice
        lda LedSsegAsciiTbl,x           ; Carga el caracter
        sta LED_SSEG_PORT               ; NO, Desplegar en la pantalla
LEDOUT0102.C
        rts                             ; Retorna

;===========================================
; LEDDISPMUXHANDLER
;                : Multiplexa el display de
;                  siete segmentos.
; OBJETIVO       : Decide el display a encender
;                  y reconoce la interrupción
;                  del temporizador.
; ENTRADA        : Ninguna
; SALIDA         : Ninguna
; REGISTROS
; AFECTADOS      : ACCA, H, X, LedMuxCtr (RAM)
;===========================================
LedDispMuxHandler
        jsr LedDispOff                  ; Apaga el display
        inc LedMuxCtr                   ; Incrementa contador de mux
        lda LedMuxCtr                   ; Carga contador
        jsr LedDispSel                  ; Selecciona el display a encender
        ldx LedMuxCtr                   ; Carga índice bajo con contador
        clrh                            ; Borra H
        lda LedDispData,x               ; Carga data a desplegar en siete
                                        ; segmentos
        jsr LedDispDataSel              ; Selecciona el número a desplegar
        jsr TIMOvAck                    ; Reconoce la interrupción del
                                        ; temporizador
        rts                             ; Retorna
```

```
;===========================================
; LEDSSEGASCIITBL
;                  :  Tabla de decodificación
; OBJETIVO         :  Tabla ASCII de los carac-
;                     teres imprimibles
;                  a
;                ------
;              f /      /
;                / g  / b
;               /-----/
;            e /     / c
;              /     /
;             ------   o <-  P.D.
;               d
;===========================================
LedSsegAsciiTbl
            ;P.D.,GFEDCBA                   ; Nota:  El punto decimal permanecerá
                                            ;           apagado
                                            ; BIT[0:6] = SEG[A:G], BIT7 = P.D.
                                            ; La Tabla Imprimible ASCII Comienza
                                            ; en #$20
          db %11111111                      ; ' '
          db %01111001                      ; '!', Desplegado como 1 y el P.D.
          db %11011101                      ; '"'
          db %11111111                      ; '#', No Desplegado
          db %11111111                      ; '$', No Desplegado
          db %11111111                      ; '%', No Desplegado
          db %11111111                      ; '&', No Desplegado
          db %11111101                      ; '''
          db %11000110                      ; '(', Desplegado como '['
          db %11110000                      ; ')', Desplegado como ']'
          db %11111111                      ; '*', No Desplegado
          db %11111111                      ; '+', No Desplegado
          db %11111111                      ; ',', No Desplegado
          db %10111111                      ; '-'
          db %01111111                      ; '.', Desplegado con el P.D.
          db %11111111                      ; '/', No Desplegado
          db %11000000                      ; '0'
          db %11111001                      ; '1'
          db %10100100                      ; '2'
          db %10110000                      ; '3'
          db %10011001                      ; '4'
          db %10010010                      ; '5'
          db %10000011                      ; '6'
          db %11111000                      ; '7'
          db %10000000                      ; '8'
          db %10011000                      ; '9'
          db %11111111                      ; ':', No Desplegado
          db %11111111                      ; ';', No Desplegado
          db %11111111                      ; '<', No Desplegado
          db %10110111                      ; '='
          db %11111111                      ; '>'
          db %00101100                      ; '?', Desplegado con el P.D.
```

```
db %10100000        ; '@', Desplegado como 'a'
db %10001000        ; 'A'
db %10000011        ; 'B', Desplegado como 'b'
db %10100111        ; 'C', Desplegado como 'c'
db %10100001        ; 'D', Desplegado como 'd'
db %10000110        ; 'E'
db %10001110        ; 'F'
db %10011000        ; 'G', Desplegado como '9'
db %10001001        ; 'H'
db %11111001        ; 'I', Desplegado como '1'
db %11110001        ; 'J'
db %11111111        ; 'K', No Desplegado
db %11000111        ; 'L'
db %11111111        ; 'M', No Desplegado
db %11001000        ; 'N'
db %11000000        ; 'O', Desplegado como '0'
db %10001100        ; 'P'
db %11111111        ; 'Q', No Desplegado
db %10101111        ; 'R', Desplegado como 'r'
db %10010010        ; 'S', Desplegado como '5'
db %11111000        ; 'T', Desplegado como '7'
db %11000001        ; 'U'
db %11000001        ; 'V', Desplegado como 'U'
db %11111111        ; 'W', No Desplegado
db %10001001        ; 'X', Desplegado como 'H'
db %10010001        ; 'Y'
db %10100100        ; 'Z', Desplegado como '2'
db %11000110        ; '['
db %11111111        ; '\', No Desplegado
db %11110000        ; ']'
db %11111111        ; '^', No Desplegado
db %11110111        ; '_'
db %11111111        ; '`', No Desplegado
db %10100000        ; 'a'
db %10000011        ; 'b', Desplegado como '6'
db %10100111        ; 'c'
db %10100001        ; 'd'
db %10000110        ; 'e'
db %10001110        ; 'f', Desplegado como 'F'
db %10010000        ; 'g'
db %10001011        ; 'h'
db %11111011        ; 'i'
db %11110011        ; 'j'
db %11111111        ; 'k', No Desplegado
db %11111001        ; 'l', Desplegado como '1'
db %11111111        ; 'm', No Desplegado
db %10101011        ; 'n '
db %10100011        ; 'o'
db %10001100        ; 'p', Desplegado como 'P'
db %10011000        ; 'q', Desplegado como 9
db %10101111        ; 'r'
db %10010010        ; 's', Desplegado como '5'
```

```
        db %10111011                    ; 't'
        db %11100011                    ; 'u'
        db %11100011                    ; 'v', Desplegado como 'u'
        db %11111111                    ; 'w', No Desplegado
        db %11111111                    ; 'x', No Desplegado
        db %10010001                    ; 'y', Desplegado como 'Y'
        db %11111111                    ; 'z'
        db %11000110                    ; '{', Desplegado como '['
        db %11001111                    ; '|'
        db %11110000                    ; '}', Desplegado como ']'
        db %11111111                    ; '~', No Desplegado
LedSsegAsciiTblTop
        db %11111111                    ; ' '
```

Listado 48. NT1007 – LED – LED.inc. Archivo de funciones de Manejo de Siete Segmentos con un decodificador de Direcciones 74LS138.

```
;*******************************************************************
; ARCHIVO      : ADC.inc
; PROPÓSITO    : Inclusión de la función de inicialización y medición de ADC
; LENGUAJE     : IN-LINE ASSEMBLER
; -------------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   30  08  04   Creado.
;   30  08  04   Modificado.
;*******************************************************************

;===========================================
;                       Constantes & Macros
;===========================================
ADC_MAX_CH      equ 12T             ; Máximo número de canales del MUX
ADC_TO          equ 15T             ; Tiempo fuera de Conversión

;===========================================
; ADCINIT      : Inicializa el convertidor
; OBJETIVO     : Inicializa el ADC con su di-
;                visor y tipo de conversión
; ENTRADA      : A  es el tipo de conversión
;                X  contiene el valor del
;                divisor del reloj de en-
;                trada del ADC
; SALIDA       : Ninguna
; REGISTROS
; AFECTADOS    : ADSCR, ADICLK, ACCA, X
;===========================================
ADCInit
        stx ADICLK                  ; Almacena divisor del ADC
        cmp #1                      ; ¿A habilitar conv. continua?
        beq ADC0011.E               ; Si no es igual, apagar ADC
        lda ADSCR                   ; Carga el ADSCR
        and #AIEN                   ; Habilita interrupciones
        ora #ADCOFF                 ; Apaga el ADC
        bra ADC0011.F               ; Salir
ADC0011.E
        lda ADSCR                   ; A = ADSCR
        and #{AIEN|ADCOFF}          ; Preserva bits de int. y canal
        ora #ADCO                   ; añade bit ADCO = 1
ADC0011.F
        sta ADSCR                   ; Almacena en el registro del ADC
        rts                         ; Retorna
```

```
;=========================================
; ADCMEASURE
;               : Decide si el canal es hábil
; OBJETIVO      : Canal disponible o no
; ENTRADA       : A   contiene el canal a medir
; SALIDA        : A   decide si el canal está
;                 disponible o no
; REGISTROS
; AFECTADOS     : ADSCR, ADICLK, ACCA, X
; NOTA          : SALIDA = 0, Señal de error
;                 SALIDA = 1, Lectura Dispo-
;                 nible
;=========================================
ADCMeasure
        cmp #ADC_MAX_CH                 ; ¿Dentro del rango de multiplexión?
        bge ADCOUT0011.D                ; NO, salir
        psha                            ; Envía a la pila
        lda ADSCR                       ; Carga Registro ADC
        and #{AIEN|ADCO}                ; Enmascara los bits altos
        ora 1,SP                        ; Cambia de canal
        sta ADSCR                       ; Almacena en el convertidor
        ldx #ADC_TO                     ; Contador de Tiempo Fuera
ADCLOOP0011.B
        lda INT3                        ; leer INT3
        bit #BIT0M                      ; Enmascara BIT0
        bne AREAD0011.A                 ; Hay interrupción, leer
        brset COCO,ADSCR,AREAD0011.A    ; Si termina la conversión, leer
        decx                            ; ADC_TO = ADC_TO - 1 (TO =
                                        ; TimeOut)
        cpx #0                          ; ¿contador = 0?
        bne ADCLOOP0011.B               ; Salta y pregunta nuevamente
        ais #1                          ; Posiciona la dirección de retorno
ADCOUT0011.D
        clra                            ; Señal de error
        bra ADCOUT0011.C                ; salir del ADC
AREAD0011.A
        lda #1                          ; Resultado Disponible
        ais #1                          ; Posiciona la dirección de retorno
ADCOUT0011.C
        psha                            ; empuja a la pila
        brset BIT5,ADSCR,ADC0011.I      ; Si ADCO = 0, salir
        lda ADSCR                       ; Carga ADSCR
        and #{AIEN|ADCO}                ; Preserva bits
        ora #ADCOFF                     ; apaga el ADC
        sta ADSCR                       ; Almacena en el registro
ADC0011.I
        pula                            ; recupera de la pila
        rts                             ; Retorna
```

Listado 49. NT1007 – LED – ADC.inc. Archivo de funciones de ADC, solo se listan las funciones utilizadas ADCInit que inicia el ADC y ADCMeasure que mide un canal.

```
;*****************************************************************
; ARCHIVO      : RAM.inc
; PROPÓSITO    : Funciones de uso de la RAM
; NOTAS        : ADVERTENCIA - Se debe especificar el inicio y el origen de
;                RAM de su microcontrolador por medio de los
;                macros RAM_ORG y RAM_END
;
; LENGUAJE     : IN-LINE ASSEMBLER
; -------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   05   10   04   Creado.
;   06   10   04   Modificado.
;*****************************************************************

;*****************************************
;                CONSTANTES & MACROS
;*****************************************
RAM_ORG      equ $0080                  ; Inicio de la memoria RAM
RAM_END      equ $00FF-1                ; Fin de limpieza de la RAM

;*****************************************
; RAMCLR       : Borra la Ram utilizada y re-
;                gistros inherentes
; OBJETIVO     : Borra registros
; ENTRADA      : Ninguna
; SALIDA       : A, H:X y RAM en 0
; REGISTROS
; AFECTADOS    : RAM, H:X, A
;*****************************************
RAMClr                                  ; Borra la RAM y registros
        clrh                            ; Borra H
        ldx #RAM_ORG                    ; Carga con el origen
RAM_EMPTY0000.A
        clr ,x                          ; rellena con "0" la posición actual
        aix #1                          ; incrementa puntero de RAM
        cphx #RAM_END                   ; Compara hasta el final deseado
        bne RAM_EMPTY0000.A             ; Si no concuerda entonces sigue
                                        ; limpiando
        clra                            ; Borra A
        clrh                            ; Borra H
        clrx                            ; Borra X
        rts                             ; retorna
```

Listado 50. NT1007 – LED – RAM.inc. Archivo de funciones de RAM, borra el contenido de la RAM.

```
;=================================================================
; ARCHIVO      : MATH.inc
; PROPÓSITO    : Librería para ruinas matemáticas.
; LENGUAJE     : IN-LINE ASSEMBLER
; NOTAS        : Ninguna
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   26   12   04   Creado.
;   26   12   04   Modificado.
;=================================================================

;=========================================
; MATHHEX2DEC
;                : Convierte un byte a Decimal
; OBJETIVO       : Convierte un caracter hexa-
;                  decimal a su correspondiente
;                  decimal.
; ENTRADA        : ACCA    es el caracter a
;                  convertir a decimal
;                  HX        puntero donde se ubi
;                            ca el resultado de
;                            la información
; SALIDA         : 3 Posiciones correspondientes
;                  al número equivalente son
;                  depositadas desde el ingreso
;                  del registro H:X.
;                  Una para las CENTENA
;                  La siguiente para las DECENA
;                  La siguiente para las UNIDAD
; REGISTROS
; AFECTADOS      : ACCA, H:X, SP, RAM
;=========================================
MathHex2Dec
        sei                             ; Inhabilita interrupciones
        pshx                            ; Empuja byte bajo
        pshh                            ; Empuja byte alto
        ldx #2T                         ; Carga contador temporal
        pshx                            ; Empuja a pila
        ldx #100T                       ; CCCA dividir Centenas
        div                             ; Divide
        psha                            ; Empuja Centena a la pila
        pshh                            ; Empuja Residuo a pila
MATHNEXT0126.B
        tsx                             ; Transfiere SP a H:X
        lda 3,x                         ; Carga Puntero Alto de la posición a
                                        ; escribir
        psha                            ; Empuja a pila puntero alto
        pulh                            ; Recupera en H para la posición a
                                        ; escribir
        ldx 4,x                         ; Carga Puntero Bajo de la posición a
                                        ; escribir
        lda 2,SP                        ; Carga Centena
```

```
        sta ,x                  ; Almacena Centena
        tsx                     ; Transfiere SP a H:X
        inc 4,x                 ; Incrementa puntero bajo en la pila
        tst 4,x                 ; Comprueba si es cero
        bne MATHNEXT0126.A      ; ¿0?, NO, Seguir
        inc 3,x                 ; SI, Incrementar puntero alto
MATHNEXT0126.A
        pula                    ; Recupera Residuo
        tsx                     ; Transfiere SP a H:X
        dec 1,x                 ; Decrementa contador
        tst 1,x                 ; Prueba si es 0
        beq MATHOUT0126.C       ; Salir si es cero
        ldx #10T                ; A dividir Decenas
        div                     ; Divide
        ais #1T                 ; Reposiciona puntero
        psha                    ; Empuja Decenas a la pila
        pshh                    ; Empuja Unidades a la pila
        bra MATHNEXT0126.B      ; Siguiente caracter
MATHOUT0126.C
        ais #1T                 ; Reposiciona pila
        tsx                     ; Transfiere SP a H:X
        ldx 2,x                 ; Carga Puntero Bajo de la posición a
                                ; escribir
        pshx                    ; Empuja a pila puntero alto
        tsx                     ; Transfiere SP a H:X
        ldx 2,x                 ; Carga Puntero Bajo de la posición a
                                ; escribir
        pshx                    ; Empuja puntero Alto de la posición a
                                ; escribir
        pulh                    ; Recupera en H
        pulx                    ; Recupera puntero Bajo en X
        sta ,x                  ; Almacena Unidad
        cli                     ; Habilita interrupciones
        ais #3T                 ; Ajusta Puntero de Retorno
        rts                     ; Retorna
```

```
;===========================================
; MATHNIBBLE2ASCII
;                 : Convierte de HEX a ASCII.
; OBJETIVO        : Convierte un caracter hexade
;                   cimal en su equivalente
;                   ASCII.
; ENTRADA         : ACCA    es el caracter a
;                   convertir en ASCII
;                   utilizando solo
;                   el nibble bajo.
; SALIDA          : ACCA    resultado de nibble
;                   bajo a ASCII.
; REGISTROS
; AFECTADOS       : ACCA
;===========================================
MathNibble2Ascii
        and #$0F                        ; Enmascara nibble bajo
        add #'0'                        ; Añade '0'
        cmp #'9'                        ; Compara con '9'
        ble MATHOUT0126.D               ; ¿Menor a o igual?, SI, Salir
        add #7T                         ; ACCA = ACCA + 'A' - '9' - 1
MATHOUT0126.D
        rts                             ; Retornar
```

```
;===========================================
; MATHDEC2ASCII
;                : Convierte una cadena de deci
;                  males a ASCII.
; OBJETIVO       : Convertir una cadena de
;                  números cada uno comprendido
;                  entre 0 y 9 a ASCII.  Y los
;                  almacena en el mismo lugar
;                  de donde fueron tomados.
; ENTRADA        : ACCA    número de decimales
;                            a convertir
;                  H:X       es la posición ini-
;                            cial de la cadena
;                            a convertir.
; SALIDA         : Ninguna.  Posiciones especi
;                  ficadas convertidas a ASCII.
; REGISTROS
; AFECTADOS      : ACCA, H:X, RAM
;===========================================
MathDec2Ascii
        psha                            ; Empuja el número de decimales a
                                        ; pila
MATHLOOP0126.F
        lda ,x                          ; Carga caracter
        jsr MathNibble2Ascii            ; Convierte a ASCII
        sta ,x                          ; Almacena en la misma posición
        aix #1                          ; Incrementa puntero
        dec 1,SP                        ; Decrementa contador
        lda 1,SP                        ; Carga Contador
        bne MATHLOOP0126.F              ; Sigue si faltan por convertir
        pula                            ; Reposiciona pila
        rts                             ; SI, Retorna
```

```
;=========================================
; MATHHEX2STR
;                : Transforma un número hexadecimal
;                  de 8 bits a una cadena
;                  ASCII.
; OBJETIVO       : Transforma un HEX a una ca-
;                  dena ASCII con un fin de
;                  línea.
; ENTRADA        : ACCA    es el dígito a con-
;                  vertir en decimal.
;                  H:X     contiene el puntero
;                  de la posición ini
;                  cial a transformar
; SALIDA         : Ninguna
; REGISTROS
; AFECTADOS      : ACCA, H:X
;=========================================
MathHex2Str
        sei                             ; Sección Crítica, Inhabilita
                                        ; interrupciones
        pshx                            ; Empuja Puntero bajo a pila
        pshh                            ; Empuja Puntero alto a pila
        jsr MathHex2Dec                 ; Tranforma 1 byte a decimal
        tsx                             ; Transfiere SP a H:X
        lda #3T                         ; Máximo de displays
        pulh                            ; Recupera puntero alto de la pila
        pulx                            ; Recupera puntero bajo de la pila
        jsr MathDec2Ascii               ; Transforma cada de bytes a ASCII
        cli                             ; Habilita interrupciones
        rts                             ; Retorna
```

Listado 51. NT1007 – LED – MATH.inc. Archivo de funciones matemáticas en ensamblador, MathHex2Dec transforma un hexadecimal a Decimal; MathNibble2Ascii transforma el nibble bajo a ASCII; MathDec2Ascii transforma un decimal a ASCII; MathHex2Str transforma un número hexadecimal a cadena ASCII con un fin de línea (0).

```
;=====================================================================
; ARCHIVO      :  TIM.inc
; PROPÓSITO    :  Inclusión de la función de inicialización/utilitario de TIM
; LENGUAJE     :  IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   00   08   04   Creado.
;   04   09   04   Modificado.
;=====================================================================

;=============================================
;                       Constantes & Macros
;=============================================
TIM_MAX_CH          equ 2T              ; Máximo de canales por temporizador
TIM_OP_LV_HI        equ 0T              ; Pin inicialmente en Alto
TIM_OP_LV_LO        equ $10             ; Pin inicialmente en Bajo

TIM_IC_RISE         equ $04             ; Captura Flanco de Subida
TIM_IC_FALL         equ $08             ; Captura Flanco de Bajada
TIM_IC_RISE_FALL    equ $0C             ; Captura ambos flancos

TIM_OC_UN_TGL       equ $14             ; Conmuta al comparar, Unb.
TIM_OC_UN_CLR       equ $18             ; Bajo al comparar, Unb.
TIM_OC_UN_SET       equ $1C             ; Alto al comparar, Unb.
TIM_OC_BU_TGL       equ $24             ; Conmuta al comparar, Buf.
TIM_OC_BU_CLR       equ $28             ; Bajo al comparar, Buf.
TIM_OC_BU_SET       equ $2C             ; Alto al comparar, Buf.

TIM_PWM_UN_TGL      equ $16             ; PWM Conmuta, Unb.
TIM_PWM_UN_CLR      equ $1A             ; PWM Bajo al comp., Unb.
TIM_PWM_UN_SET      equ $1E             ; PWM Alto al comp., Unb.
TIM_PWM_BU_TGL      equ $26             ; PWM Conmuta, Buf.
TIM_PWM_BU_CLR      equ $2A             ; PWM Bajo al comp, Buf.
TIM_PWM_BU_SET      equ $2E             ; PWM Alto al comp. Buf.
```

```
;=========================================
; TIMINIT       : Inicializa el Módulo TIM
; OBJETIVO      : Inicializa el TIM y configura
;                 divisor y Módulo
; ENTRADA       : A  valor del prescalar
;                 HX contiene el valor del
;                 registro TMOD[H:L]
; SALIDA        : Ninguna
; REGISTROS
; AFECTADOS     : TSC, TMOD[H:L], ACCA, SP
;=========================================
TIMInit
        psha                            ; Empuja A a la pila
        lda TSC                         ; Carga TSC
        and #{TOIE|TSTOP}               ; Preserva Bits
        ora 1,SP                        ; or con el prescalar
        sta TSC                         ; Almacena en TSC
        ais #1                          ; reubica puntero de retorno
        sthx TMODH                      ; Almacena Módulo de contador
        rts                             ; Retorna

;=========================================
; TIMStart      : Inicia el módulo TIM
; OBJETIVO      : Apaga el bit de parada
; ENTRADA       : Ninguna
; SALIDA        : Ninguna
; REGISTROS
; AFECTADOS     : CCR, TSC
;=========================================
TIMStart
        sei                             ; Inhabilita Interrupciones
        bclr BIT5,TSC                   ; TSTOP = 0, TIM Hábil
        cli                             ; Habilita Interrpciones
        rts                             ; Retorna
```

```
;===============================================
; TIMOVACK    :  Borra bandera de sobreflujo
;                del TIM
; OBJETIVO    :  TSC_TOF = 0
; ENTRADA     :  Ninguna
; SALIDA      :  Ninguna
; REGISTROS
; AFECTADOS   :  TSC, CCR
;===============================================
TIMOvAck
        sei                             ; Deshabilita interrupciones
        bclr BIT7,TSC                   ; borra desborde
        cli                             ; Habilita interrupciones
        rts                             ; retorna
```

Listado 52. NT1007 – LED – TIM.inc. Archivo de funciones del temporizador. TIMInit inicializa el temporizador con un preescalar y tiempo determinado por el usuario, TIMOvIntEn habilita las interrupciones del módulo TIM, TIMStart inicia el temporizado mientras que TIMOvAck reconoce la interrupción. Solo se listan las funciones utilizadas.

```
;=================================================================
; ARCHIVO      : INTERRUPTSJL3.inc
; PROPÓSITO    : Plantilla de Funciones de Interrupciones para
;                JL3/JK1/JK3/Serie QT/QY
; LENGUAJE     : IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;  DD MM AA
;  22  08  04   Creado.
;  06  12  04   Modificado.
;=================================================================

;=========================================
;          Interrupción de Temporizador del Sistema
;=========================================
;=========================================
; TIMOFL       : Servicio de interrupción del
;                temporizador del sistema
; OBJETIVO     : Reconoce la interrupción y
;                retorna al programa principal
; ENTRADA      : Ninguna
; SALIDA       : Ninguna
; REGISTROS
; AFECTADOS    : Ninguno
;=========================================
TIMOFL                                  ; TIM Sobreflujo (Bajo)
      jsr LedDispMuxHandler             ; Reconoce la interrupción y
                                        ; multiplexa las pantallas
      rti                               ; NO, retorna de la interrupción
```

Listado 53. NT1007 – LED – INTERRUPTSJL3.inc. Solamente es listada la interrupción del temporizador, la cual es utilizada para refrescar las pantallas.

```
;=====================================================================
; ARCHIVO      :  VECTORSJL3.inc
; PROPÓSITO    :  Definir el vector de búsqueda de cada interrupción
; LENGUAJE     :  IN-LINE ASSEMBLER
; ---------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   22   08   04   Creado.
;   06   12   04   Modificado.
;=====================================================================

;=============================================
;                Vector de Temporizador del Sistema
;=============================================
        org TIMOFH                          ; Sobreflujo del TIM (Alto)
        dw TIMOFL                           ; Sobreflujo del TIM (Bajo)

;=============================================
;      Vector de Reinicio del Sistema
;=============================================
        org RESET_VEC                       ; Puntero Vec - RESET
        dw START                            ; al darse reset salta a Start
```

Listado 54. NT1007 – LED – VECTORSJL3.inc. Listado solamente del vector utilizado, temporizador.

3.7.6 Conclusión

Nuevamente, se utiliza el multiplexado de pantallas de siete segmentos, mostrando que independientemente del “hardware”, se puede conseguir un resultado similar. El propósito de haber mostrado tres notas con hardware diferente y resultados iguales, destaca que se puede conseguir cualquier fin previsto de muchas formas. La manera en que el programador decida realizar una aplicación dependerá de la genialidad del mismo.

La única diferencia entre estas tres notas, es “el costo”, a medida que integremos la solución más hacia el “chip”, es decir, tendencia a la programación y obviemos el “hardware”, el programa se hace más complejo, pero los costos disminuyen. Otro punto a favor es que el PCB o circuito impreso es reducido pues el hardware utilizado es mínimo.

Este tipo de rutinas multiplexadas, se utilizan en todo tipo de aplicaciones, no solo es aplicable a siete segmentos, también se pueden multiplexar la cantidad de canales del ADC y tomar datos de una red de sensores inteligentes, de los cuales luego pueden desplegarse en una pantalla o tomar decisiones con respecto a los mismos estados actuales.

3.7.7 Referencias

3.7.8.1 "Embedded Systems Building Blocks, Second Edition" - "Complete and Ready-to-Use Modules in C"
Autor: Jean J. Labrosse
Recurso: Capítulo 04 – Multiplexed LED Displays

3.7.8.2 Sistemas Digitales, Principios y Aplicaciones
Autor: Ronald J. Tocci
Recurso: Capítulo 9 – Sección 9.7, Multiplexores

3.7.8.3 Información Avanzada sobre el Microcontrolador
(a) http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC68HC08JL3.pdf

3.7.8.4 Manual de Referencia del CPU
(a) http://www.freescale.com/files/microcontrollers/doc/ref_manual/CPU08RM.pdf

3.7.8.5 Página web sobre esta Nota Técnica
(a) http://www.geocities.com/issaiass/

NT0101 – SSEG – Manejo de Siete Segmentos. – NT1005 Tesis
NT0102 – SSEGMUXD – Manejo de Siete Segmentos Multiplexado. – NT1006 Tesis
NT0026 – Módulos – Rutinas reusables para programación – Apéndices en Tesis

3.7.8.6 Multiplexión de Siete Segmentos
(a) http://www.mcmanis.com/chuck/robotics/fpga/projects/5/multi-led.html
(b) http://claymore.engineer.gvsu.edu/~jackh/eod/mechtron/mechtron-77.html

3.7.8.7 Nota de Aplicación Sobre el Cálculo de Resistores para LEDs
(a) http://www.freescale.com/files/microcontrollers/doc/app_note/AN1238.pdf

AN1238 – "HC05 MCU LED Drive Techniques Using the MC68HC705J1A"

3.7.8 Problemas Propuestos

3.7.9.1 Despliegue la palabra "CAN" en tres siete segmentos.

3.7.9.2 Realice con su rutina de despliegue caracteres secuencias de letras moviéndose hacia la izquierda.

Nota: Para los programas anteriores utilice la rutina LedDispStr.