Notas Técnicas de Uso y Aplicación

NT1005

3.5 INTERFASE A PANTALLA DE SIETE SEGMENTOS
CONTADOR DE CERO (0) A NUEVE (9) UTILIZANDO UN DECODIFICADOR 74LS47

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

3.5.1 Introducción

Los LEDs (Diodo Emisor de Luz) son dispositivos semiconductores que emiten luz si se aplica corriente en sentido directo. Normalmente son utilizados como luz piloto, la cual puede significar: el estado de una fuente encendida, la activación de una alarma o un evento indeseado, el despliegue de información.

Generalmente, cuando se trata de desplegar información con LEDs, se utiliza un conjunto de ellos en paquetes que traen una cantidad significativa, así, con los denominados siete segmentos, podemos desplegar: El valor de una entrada analógica codificada en números naturales, el contenido de la memoria del microcontrolador, etc. Claro que los caracteres desplegados son limitados, pero nos dan cierto sentido a la información que el usuario busca.

La nota tratará de explicar los conceptos básicos de:

- Cálculo de la resistencia limitadora: El puerto del microcontrolador tiene la capacidad de manejar directamente LEDs, pero con pocos miliamperios de entrega. Se explicará como calcular dicho resistor para no dañar el puerto.
- ¿Qué es un LED, Decodificador, Siete segmentos?: Breve explicación del "hardware" que se utiliza normalmente para desplegar información
- "Hardware" y "Software": La nota termina con el esquemático y un "software" de aplicación que genera un conteo ascendente empezando desde cero (0) hasta nueve (9) utilizando una tabla.

Advertencia: En todas las notas técnicas solo se hace referencia en cuanto a pines con el "hardware" específico, para otro tipo de "hardware", ver data de fabricante.

3.5.2 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
9. Display: Visualizador de 7 Segmentos, Jameco Part No.: 24731, MAN71
10. Resistores: Resistencias de 7 × 470Ω ¼ Watt, Amarillo, Violeta, Negro.

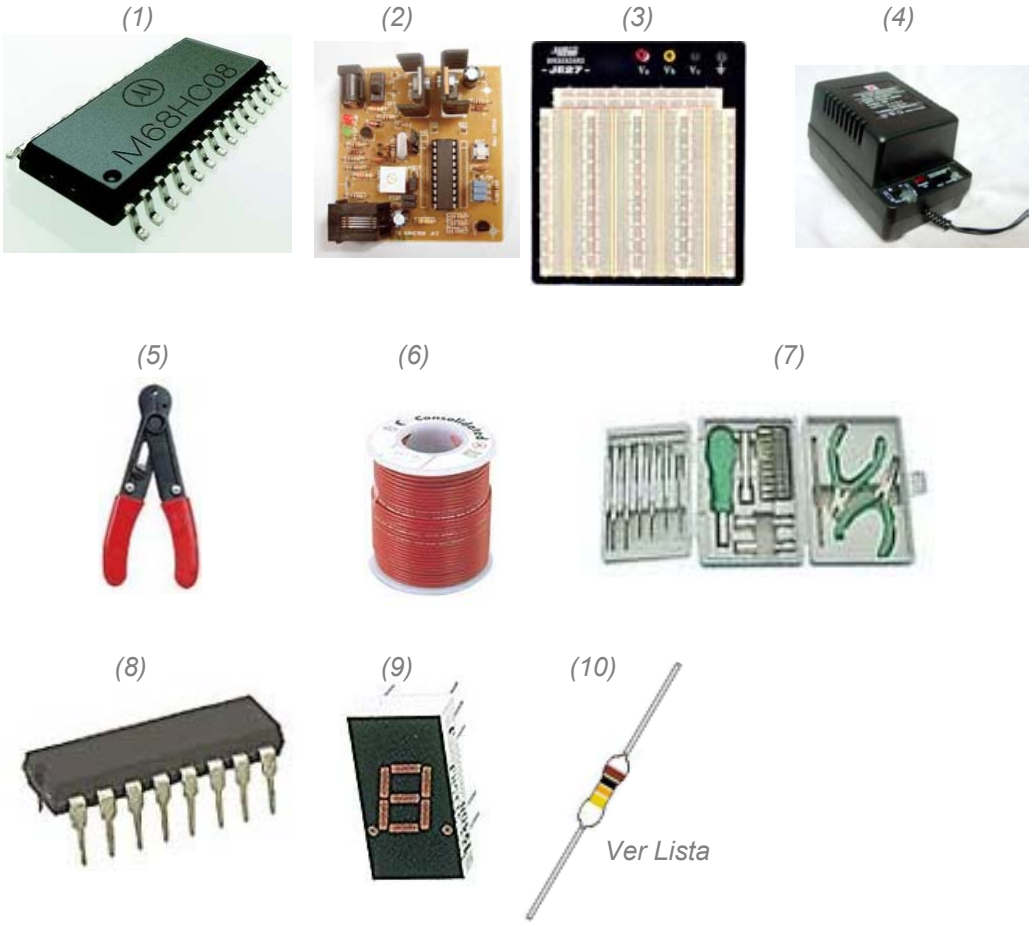


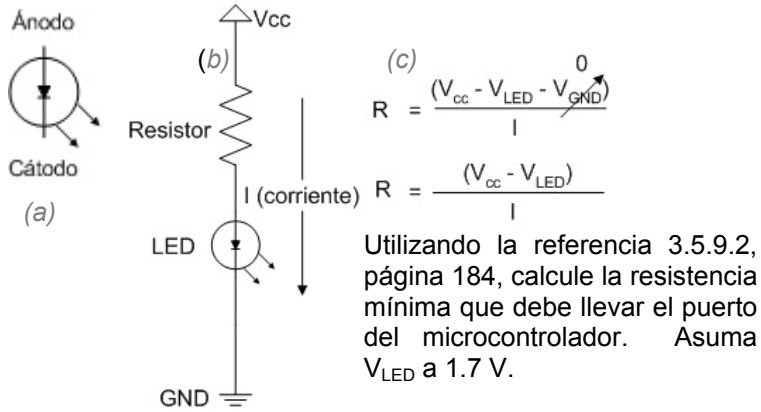
Figura 183. Listado de Materiales para Pantalla de Siete Segmentos

3.5.3 Descripción Física del "Hardware"

Para el siguiente párrafo, ayudarse por medio de la figuras de esta sección.

3.5.3.1 ¿Qué es un LED?

Un LED o diodo emisor de luz es un dispositivo semiconductor que tiene como peculiaridad irradiar luz, pero si solo se polariza directamente (ver figura 184(b)), es decir, su ánodo (terminal en forma de A) debe estar potencialmente a mayor voltaje que su cátodo (terminal con raya). Los mismos se pueden conectar directamente al puerto del microcontrolador si se les conecta un resistor de carga (ver cálculo en figura 184(c)).



Utilizando la referencia 3.5.9.2, página 184, calcule la resistencia mínima que debe llevar el puerto del microcontrolador. Asuma V_{LED} a 1.7 V.

$$R = (5 - 1.7)V / 1.6\,mA = 2k\Omega$$

3.5.3.2 ¿Qué es un Siete Segmentos?

Son arreglos de LEDs en encapsulados, de los cuales se estudian en la nota los visualizadores numéricos. Dentro de estos existen de tipo Ánodo Común y Cátodo Común (figura 184(d)), la palabra común, simboliza que sus terminales se entrelazan (ver figura 184(e)) y se conecta a V_{CC}. Así, para que cada LED sea activado, se debe enviar un pulso en bajo para polarizar el LED. El dispositivo a utilizar para controlar los segmentos es estudiado en la siguiente sección 3.5.3.3.

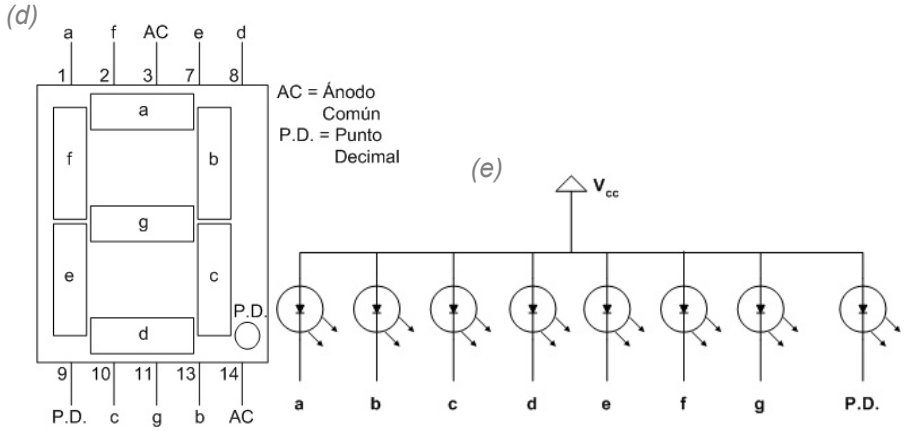


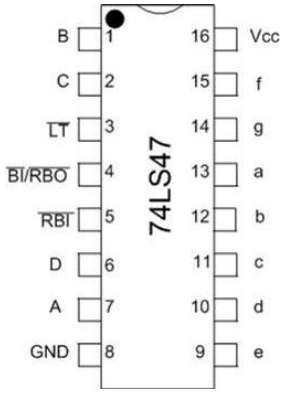
Figura 184. Descripción Física de LEDs y Siete Segmentos. (a) Esquemático de un LED. Diodo que emite luz, entre alguna de sus funciones está ser una luz piloto o aviso de alarma de un evento, existen en diferentes colores como azul, rojo, verde y blanco dependiendo de su fabricación. (b) Polarización de un LED. Para encender el LED, su ánodo debe estar a mayor potencial que su cátodo y debe llevar obligatoriamente un resistor para limitar la corriente eléctrica, su caída entre terminales está alrededor de los 1.7 a 2 V. (c) Cálculo del Resistor Mínimo en los Pines de E/S de un HC08. La resistencia más pequeña que puede soportar un puerto del microcontrolador HC08 depende de la corriente de suministro que por lo general es menor a los 1.6 mA, ver data del fabricante. (d) Hardware, Siete Segmentos tipo Ánodo Común. Denominado siete segmentos, debido a los caracteres numéricos imprimibles dentro de él, solo son dados por 7 diodos LED de la "a" a la "g" . (e) Esquemático, Siete Segmentos Ánodo Común. Su terminal común o Ánodo, debe conectarse al voltaje V_{CC} para que cada segmento se encienda dado un voltaje bajo en su cátodo.

3.5.3.3. ¿Qué es un Decodificador?

Circuito lógico que acepta un código binario BCD (Decimal Codificado en Binario) y lo transforma a su correspondiente salida numérica. La manera más rápida a saber si es un decodificador, más sin embargo no muy apropiada, es que el decodificador posee pocas entradas y muchas salidas. Dichas salidas dependen del valor de las entradas, p.e., un decodificador de "n" entradas, posee 2^n posibles combinaciones booleanas (1s y 0s) entre las salidas, de las cuales, el decodificador puede no utilizarlas todas.

Para decodificar un conjunto binario (BCD 8421, ver referencia 3.5.8.6), utilizaremos un 74LS47 o decodificador de BCD a Siete Segmentos, el cual toma una entrada BCD y transforma al correspondiente dígito decimal encendiendo los segmentos correspondientes. Cabe destacar que el 74LS47 es del tipo colector abierto y solo enviará en sus salidas un cero lógico, es decir, es un decodificador especial para encender pantallas de tipo Ánodo Común.

(a)



A, B, C, D = entradas (micro -> 74LS74)
a @ g = salidas (74LS47 -> 7 Segmentos)
VCC y GND = Alimentación (+5 V)

Tabla 74. Tabla de Verdad del 74LS74

	Entradas						
Decimal	LT	RBI	D	C	B	A	RBO
0	H	H	L	L	L	L	H
1	H	X	L	L	L	H	H
2	H	X	L	L	H	L	H
3	H	X	L	L	H	H	H
4	H	X	L	H	L	L	H
5	H	X	L	H	L	H	H
6	H	X	L	H	H	L	H
7	H	X	L	H	H	H	H
8	H	X	H	L	L	L	H
9	H	X	H	L	L	H	H
10	H	X	H	L	H	L	H
11	H	X	H	L	H	H	H
12	H	X	H	H	L	L	H
13	H	X	H	H	L	H	H
14	H	X	H	H	H	L	H
15	H	X	H	H	H	H	H

X = No importa.

(b) Salidas Activadas por 74LS47 a 7Segmentos AC

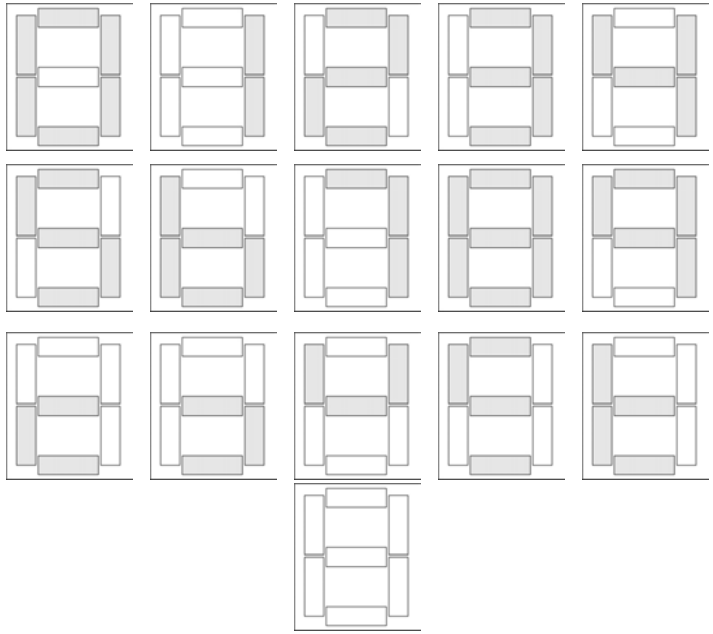


Figura 185. Función del Decodificador de BCD a 7 Segmentos. (a) "Hardware", Decodificador de BCD a 7 Segmentos. Estructura física de un decodificador de BCD a 7 Segmentos; los pines A, B, C y D son las entradas en BCD, "a" @ "g" salidas a activar los segmentos del 7 segmentos tipo ánodo común. (b) Salidas activadas por 74LS74. Dependiendo del estado de las entradas marcadas en gris en la tabla 74, genera una salida que enciende igualmente cada segmento, formando así una figura que corresponde a su equivalente número. Nota: Luego del 9, los caracteres no pertenecen al grupo decimal, son "caracteres extraños".

3.5.4 Esquemático de Aplicación

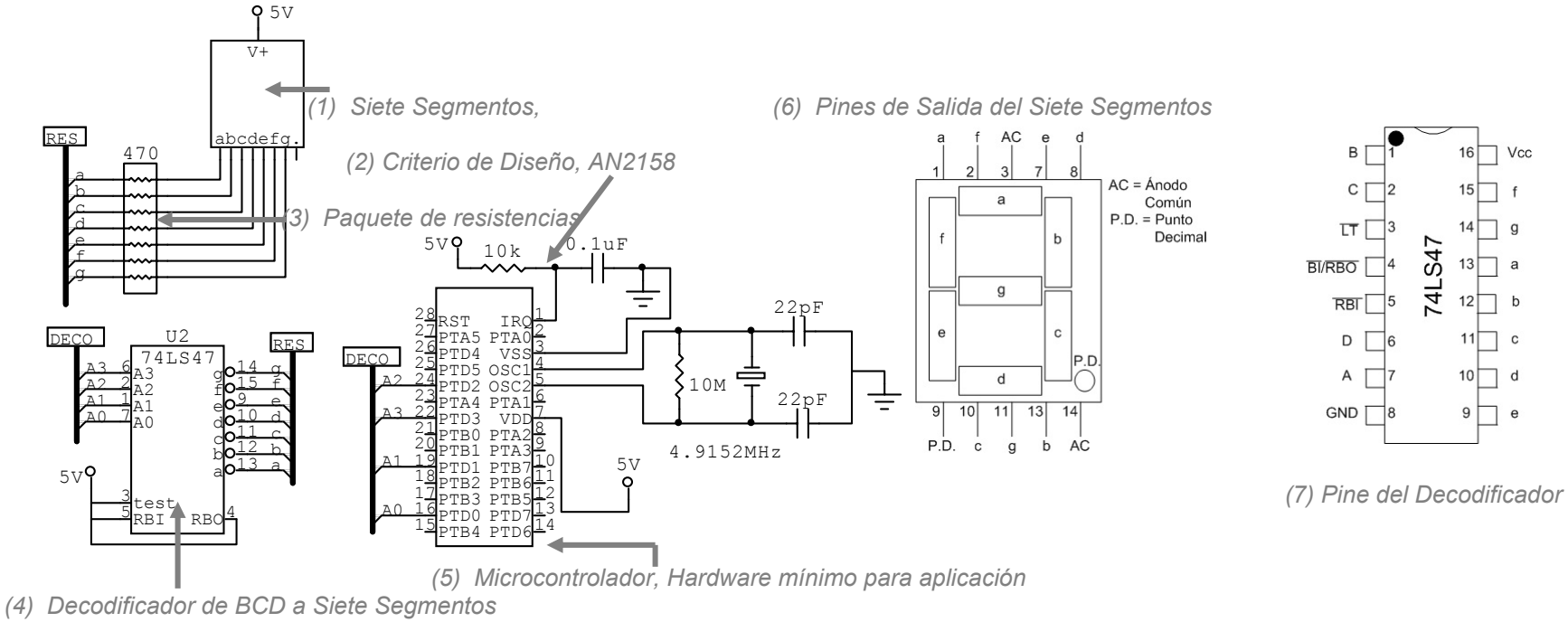


Figura 186. Esquema de Aplicación del Manejo de un Siete Segmentos. El esquema de aplicación consta de un siete segmentos a utilizarse como contador. La señal de salida es procesada paralelamente hacia un decodificador de BCD a 7 segmentos que se encargará de encender los respectivos LEDs generando los números naturales desde cero (0) hasta nueve (9).

3.5.5 Diagrama de Flujo

El siguiente programa utiliza un siete segmentos y un decodificador de BCD a Siete Segmentos (74LS47), el mismo genera una cuenta ascendente de cero (0) a nueve (9) y utiliza el temporizador del sistema para realizar el refrescado de la pantalla a 1 Hz. Cabe destacar que se utiliza una tabla de búsqueda o LUT para acceder al BCD, pero la misma no es necesaria por la ubicación de los pines, aún así se muestra su mecánica de búsqueda. Si desea conocer en detalles como se hizo esta nota ud. necesita de los documentos básicos, NT0010, NT1005 y Apéndices C, I y M.

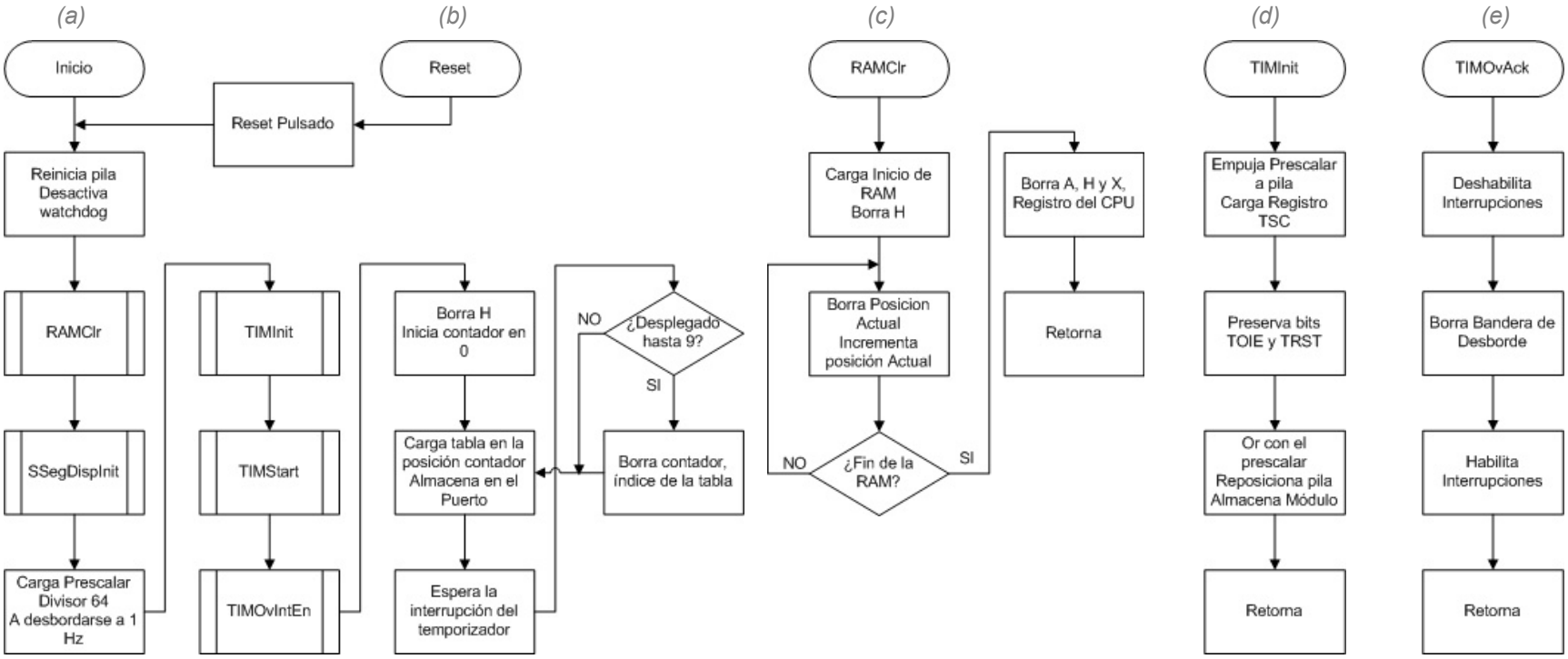


Figura 187. NT1005 – SSEG. (a) Programa Principal. Inicia un temporizador y espera la interrupción, luego del retorno empieza a mostrar los el carácter de momento, la operación se repite cíclicamente realizando un contador de cero (0) hasta nueve (9). (b) Señal de Reinicio del sistema. Al presionar, reinicia el sistema sin importar lo que se esté haciendo. (c) RAMClr. Rutina que incia en 0 toda la memoria del microcontrolador. (d) TIMInit. Rutina programable que inicia un temporizado, en este caso a 1 segundo (1 Hz). (e) TIMOvAck. Rutina que Reconoce la interrupción del módulo.

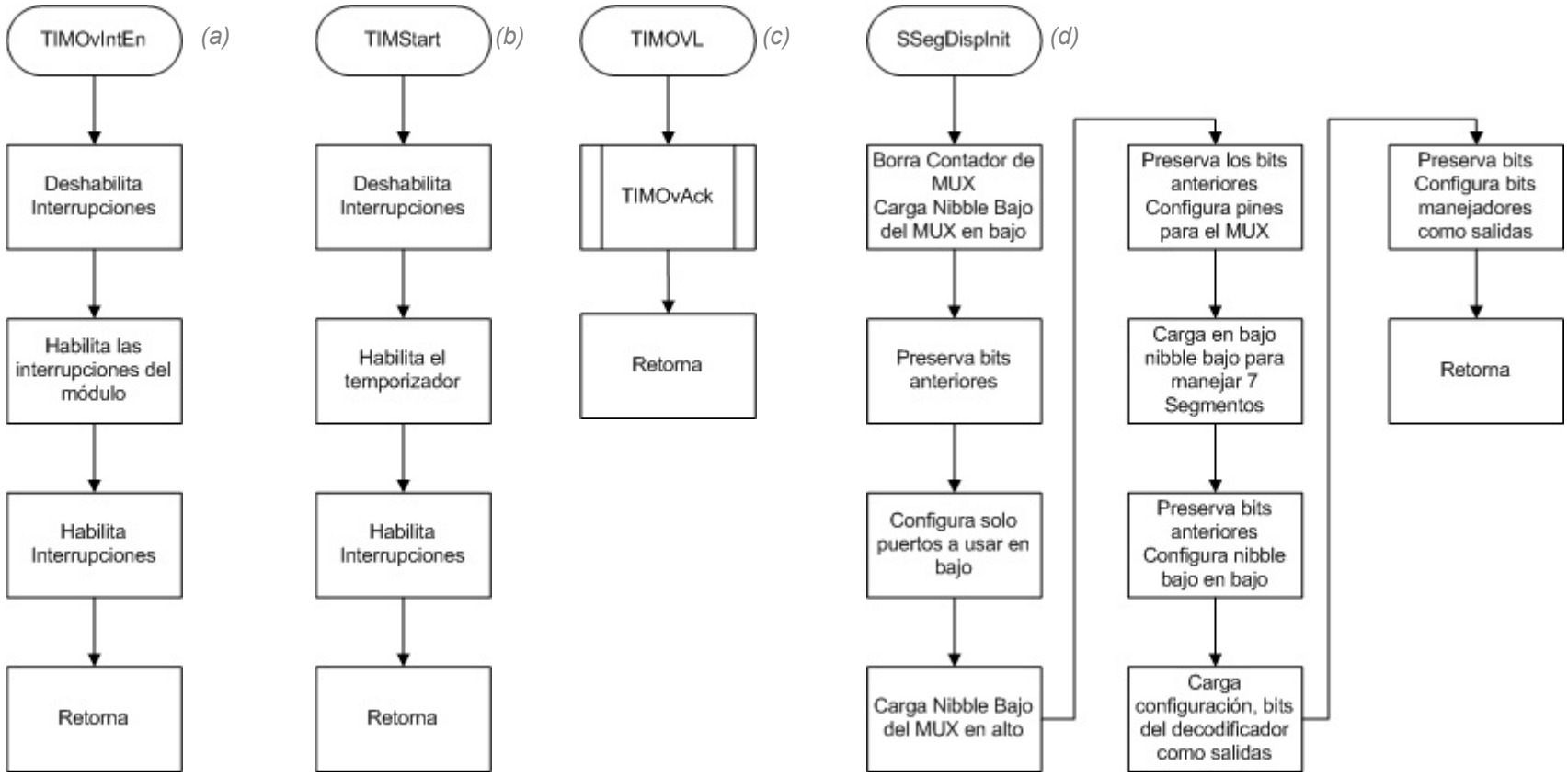


Figura 188. NT1005 – SSEG - Continuación. (a) TIMOvIntEn. Rutina que activa la interrupción de desborde del temporizador. (b) TIMStart. Rutina que arranca el temporizado. (c) TIMOVL. Subrutina de interrupción, en este caso, solo reconoce la interrupción. (d) SSegDispInit. Rutina de Inicialización de la Pantalla de Siete Segmentos. Configura el nibble (grupo de 4 bits) bajo del microcontrolador para utilizar el decodificador y futuramente un multiplexor.

3.5.6 Código

Antes de ingresar a leer el código, se recomienda las lecturas de las notas técnicas NT010, NT1005 y Apéndices C, I Y M.

```
;=================================================================
; ARCHIVO      : NT1005 - SSEG - 02 01 05.asm
; PROPÓSITO    : Despliega un conteo ascendente en el siete segmentos utili-
;                zando un decodificador de BCD a 7 segmentos (74LS47).
; NOTAS        : Ninguna
;
; REFERENCIA:  NT1005 - LED - 02 01 05.doc
;
; LENGUAJE     : IN-LINE ASSEMBLER
; ---------------------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   26   05   03   Creado.
;   02   01   05   Modificado.
;=================================================================

$SET    PGM                     ; PGM = 1, Vamos a programar la pastilla
                                ; la aplicación debe correr en tiempo real

;$SETNOT  PGM                   ; PGM = 0, Vamos a simular en la pastilla
                                ; la velocidad de simulación es menor en la
                                ; PC.

;=============================================
;            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     :  Despliega el contenido de
;                 una tabla en el siete seg-
;                 mentos.
;===============================================
START
        rsp                         ; Inic.Stack = $00ff
        bset BIT0,CONFIG1           ; Desactiva Watchdog
        jsr RAMClr                  ; Borra RAM y registros
        jsr SSegDispInit            ; Inicializa pantalla
$IF PGM
        lda #DIV64                  ; PS[0:2] = 6
        ldhx #$4B00                 ; Desborde a 1 Hz
        jsr TIMInit                 ; Inicializa TIM
        jsr TIMStart                ; Inicia temporizado
        jsr TIMOvIntEn              ; Habilita interrupciones
$ENDIF
        clrh                        ; Borra H
        clrx                        ; Inicializa contador
LEDAGAIN1005.AA
        lda LedDecTbl,x             ; Carga tabla
        sta PORTD                   ; Almacena valor
$IF PGM
        wait                        ; Espera desborde
$ENDIF
        incx                        ; Incrementa contador
        cmp #9T                     ; Compara con el máximo
        bne LEDAGAIN1005.AA         ; ¿Menor a 9?
        clrx                        ; Borra X
        bra LEDAGAIN1005.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 32. NT1005 – SSEG. Inicializa un temporizador, activa la interrupción y carga el primer carácter (0), luego espera la interrupción y al retornar carga el siguiente. La operación se repite indefinidamente realizando un contador ascendente desde (0) hasta nueve (9).

```
;=================================================================
; ARCHIVO      :  TABLES.inc
; PROPÓSITO    :  Tablas de búsqueda predefinidas por el usuario
; LENGUAJE     :  IN-LINE ASSEMBLER
; ----------------------------------------------------------------
; HISTORIAL
;   DD MM AA
;   06   09   04   Creado.
;   11   11   04   Modificado.
;=================================================================

;=========================================
;                  Tablas de Búsquedas
;=========================================
;=========================================
;                  TABLA DE MENSAJES
;=========================================
LedDecTbl
          db %00000000
          db %00000001
          db %00000010
          db %00000011
          db %00000100
          db %00001005
          db %00000110
          db %00000111
          db %00001000
          db %00001001
```

Listado 33. NT1005 – SSEG – TABLES.inc. Tabla de Decodificador, Valor actual de las entradas A, B, C, D dado por el índice X de momento. Nota: TABLES.inc está listado en USER.inc.

```
;*****************************************************************
; 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 34. NT1005 – SSEG – RAM.inc. Rutina programable que limpia registros y memoria RAM.

```
;*****************************************************************
; 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.
;*****************************************************************

;*****************************************
; TIMINIT      : Inicializa el Módulo TIM
; OBJETIVO     : Inicializa el TIM y configura
;                divisor y Módulo
; ENTRADA      : A  valor del preescalar
;                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

;*****************************************
; 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
```

```
;=============================================
; 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
```

Listado 35. NT1005 – SSEG – TIM.inc. Solo se listan las rutinas utilizadas TIMInit y TIMOvAck y TIMStart, que realizan el control de tiempo del programa principal.

```
;=================================================================
; ARCHIVO      : SSEG.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 "SSEGMUXHANDLER" 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 o el decodificador de
;               teclado:
;               $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 = SSEG_MUX_PORT
;               2.2 - Registro del Puerto del Multiplexor = SSEG_MUX_DDR
;               2.3 - Máximo de caracteres a multiplexar = SSEG_MAX_DISP
;            3 - Las características del Decodificador de BCD-7Segmentos
;               3.1 - Puerto del Decodificador = SSEG_DEC_PORT
;               3.2 - Registro del Puerto del Decodificador = SSEG_DEC_DDR
;            4 - La estructura de despliegue de información Ubicando la
;               Dirección en la memoria RAM de:
;               4.1 - Variable de control del Multiplexor = SSegMuxCtr
;
; ADVERTENCIA :
;              La estructura en RAM, depende de la ubicación de LedCtr
;              y usa "el número de caracteres a multiplexar" + 2, definida
;              por la ubicación (es decir, empezando) por la variable
;              LedCtr.
;              LA RUTINA "SSEGMUXHANDLER", DEBE IR ADMINISTRADA POR UN
;              TEMPORIZADOR INICIADO A "N * 60 Hz", EN DONDE "N" ES LA
;              CANTIDAD DE 7 SEGMENTOS A UTILIZAR.
;
;=================================================================

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

```
;=============================================
;                  Constantes & Macros
;=============================================
;=============================================
;          DEFINA PUERTO DE MULTIPLEXIÓN
;=============================================
SSEG_MUX_PORT           equ PORTA     ; Puerto del Multiplexor
SSEG_MUX_DDR            equ DDRA      ; Registro del Puerto del Multiplexor
SSEG_MAX_DISP           equ 3T        ; Máximo de caracteres a multiplexar

;=============================================
;              DEFINA PUERTO DE CONTEO
;=============================================
SSEG_DEC_PORT           equ PORTD     ; Puerto del Decodificador
SSEG_DEC_DDR            equ DDRD      ; Registro del Puerto del Decodificador

;=============================================
; DEFINA LUGAR DE LA ESTRUCTURA DE DESPLIEGUE
;=============================================
SsegMuxCtr              equ $80       ; Variable de control del
                                      ; Multiplexor
SsegDecDispData         equ SSegMuxCtr+1
                                      ; Inicio de data a desplegar en 7
                                      ; segmentos
SsegDecDispDataTop      equ {SSegDecDispData + SSEG_MAX_DISP}
                                      ; Fin de la data a desplegar en 7
                                      ; segmentos

;=============================================
;  CONSTANTES DE MULTIPLEXOR Y DECODIFICADOR
;=============================================
SSEG_MUXNIBHIOFF        equ $8F       ; Nibble Alto en estado bajo
SSEG_MUXNIBLOOFF        equ $F8       ; Nibble Bajo en estado alto
SSEG_MUXNIBHIOUT        equ $70       ; Nibble Alto son salidas
SSEG_MUXNIBLOOUT        equ $07       ; Nibble Bajo son salidas

SSEG_DECNIBHIOFF        equ $0F       ; Nibble Alto en estado bajo
SSEG_DECNIBLOOFF        equ $F0       ; Nibble Bajo en estado alto
SSEG_DECNIBHIOUT        equ $F0       ; Nibble Alto son salidas
SSEG_DECNIBLOOUT        equ $0F       ; Nibble Bajo son salidas
SSEG_DECNIBHIHI         equ $F0       ; Nibble Alto en alto
SSEG_DECNIBLOHI         equ $0F       ; Nibble Bajo en alto
```

```
;===========================================
; SSEGDISPINIT
;                 : Inicializa el módulo de 7
;                   Segmentos, Leds.
; OBJETIVO        : Inicializa el módulo de
;                   multiplexión de 7 segmentos,
;                   leds
; ENTRADA         : Ninguna
; SALIDA          : Ninguna
; REGISTROS
; AFECTADOS       : ACCA, SSEG_MUX_PORT,
;                   SSEG_MUX_DDR, SSEG_DEC_PORT,
;                   SSEG_DEC_DDR, SsegMuxCtr
;                   (STRU)
; NOTAS           : Asegúrese de inicializar
;                   correctamente el número de
;                   displays a multiplexar.
;===========================================
SSegDispInit
        clr SSegMuxCtr              ; Borra contador del multiplexor
$IF  MUX_NIBBLE_HI
        lda #SSEG_MUXNIBHIOFF       ; Nibble Alto en estado bajo
$ELSEIF
        lda #SSEG_MUXNIBLOOFF       ; Nibble Bajo en estado bajo
$ENDIF
        and SSEG_MUX_PORT           ; Preservar bits
        sta SSEG_MUX_PORT           ; Puertos a usar en bajo
$IF  MUX_NIBBLE_HI
        lda #SSEG_MUXNIBHIOUT       ; Nibble alto son salidas
$ELSEIF
        lda #SSEG_MUXNIBLOOUT       ; Nibble bajo son salidas
$ENDIF
        ora SSEG_MUX_DDR            ; Preserva bits
        sta SSEG_MUX_DDR            ; Puertos a usar son salidas
$IF  DEC_NIBBLE_HI
        lda #SSEG_DECNIBHIOFF       ; Nibble alto en bajo
$ELSEIF
        lda #SSEG_DECNIBLOOFF       ; Nibble bajo en bajo
$ENDIF
        and SSEG_DEC_PORT           ; Preserva los bits anteriores
$IF  DEC_NIBBLE_HI
        ora {SSegDecDispData < 4}   ; Carga el primer dato a desplegar
$ELSEIF
        ora SSegDecDispData         ; Carga el primer dato a desplegar
$ENDIF
        sta SSEG_DEC_PORT           ; Configura los bits del puerto a usar en bajo
$IF  DEC_NIBBLE_HI
        lda #SSEG_DECNIBHIOUT       ; Configura salidas del nibble alto a usar
$ELSEIF
        lda #SSEG_DECNIBLOOUT       ; Configura salidas del nibble bajo a usar
                                    ; Carga nibble
$ENDIF
```

```
    ora SSEG_DEC_DDR        ; Preserva bits
    sta SSEG_DEC_DDR        ; Configura bits de Decodificador como salidas
    rts                     ; SI, Salir
```

Listado 36. NT1005 – SSEG – SSEG.inc. Solo se listan la rutina utilizada SSegDispInit que realizan la configuración del siete segmentos.

```
;=================================================================
; 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)
$IF PGM
        jsr TIMOvAck            ; Reconoce la interrupción
$ENDIF
        rti                     ; NO, retorna de la interrupción
```

Listado 37. NT1005 – SSEG – INTERRUPTSJL3.inc. Archivo que contiene la definición de la interrupción del microcontrolador. Nota: Solo listada la interrupción utilizada.

```
;=====================================================================
; 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 38. NT1005 – SSEG – VECTORSJL3.inc. Archivo que contiene la declaración de la interrupción del microcontrolador. Nota: Solo listada la interrupción utilizada.

3.5.7 Conclusión

Los LEDs se utilizan como indicadores de estado a eventos ocurridos, en cambio, los siete segmentos, en este caso, numéricos, tienen uso para despliegue de información como lo es un contador, p.e., que fue el caso estudiado en la nota de aplicación.

El hecho de utilizarlo como contador, con un decodificador, desecha toda la capacidad del microcontrolador, pero si se compara con una lógica combinacional, que es la que generalmente se utiliza en este tipo de circuitos contadores, se observaría que el hardware es reducido y se puede reconfigurar el tiempo y ubicación de pines, dependiendo de la aplicación.

Frecuentemente estos dispositivos (matrices de LEDs) se pueden ver en un reloj digital, anuncios electrónicos, antiguas fuentes de voltaje digitales, etc. La mayoría de los dispositivos de años atrás utilizaban este tipo de visualizadores.

A pesar de que se pudiese formar una cadena de estos dispositivos y visualizar más caracteres numéricos, el mismo método puede ser perjudicial a futuro debido a que este tipo de pantallas eleva el consumo y no es recomendable para aplicaciones en donde se requiere un bajo consumo.

3.5.8 Referencias

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

3.5.8.2 Sistemas Digitales, Principios y Aplicaciones
Autor: Ronald J. Tocci
Recurso: Capítulo 9 – Sección 9.1, Decodificadores

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

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

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

NT0026 – Módulos – Rutinas reusables para programación.

3.5.8.6 Código BCD 8421
(a) http://www.shef.ac.uk/physics/teaching/phy107/codes.html
(b) http://www.monografias.com/trabajos3/bcd/bcd.shtml

3.5.8.7 LEDs, Siete Segmentos y Matrices de LEDS
(a) http://www.monografias.com/trabajos11/leds/leds.shtml

3.5.8.8 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"

3.5.9 Problemas Propuestos

3.5.9.1 Realice un contador ascendente – descendente de cero a nueve.

3.5.9.2 Haga que del programa anterior, se cambie el estado de conteo dependiendo del estado de un interruptor.