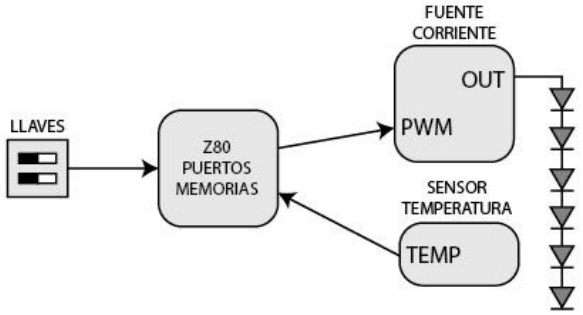
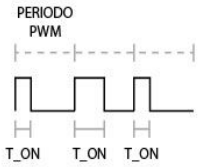
- Nombre y CI en cada hoja - Numere las hojas - Indique el total de hojas en la primera - Utilice solo un lado de las hojas	- Incluya un solo problema por hoja - **Sea prolijo** - **Aprobación:** mínimo UN problema

PROBLEMA 1

Se desea diseñar, basandose en un sistema con Z80, un circuito de control para leds de alta intensidad. El controlador cuenta con 2 llaves de entrada para seleccionar la intensidad de luz, un sensor de temperatura para proteger los leds y una salida PWM que comanda una fuente de corriente.



La señal PWM es una secuencia de pulsos de período constante cuyo tiempo en 0 y 1 puede variar para controlar la cantidad de corriente enviada a los leds.



A mayor tiempo en 1, mayor corriente. Dependiendo del valor de las llaves es la intensidad de luz que deben emitir los leds: 00: leds apagados (salida PWM siempre 0) , 01: leds emitiendo en nivel 1, 10: leds emitiendo en nivel 2, 11: leds emitiendo en nivel 3.

El período de la señal PWM y el tiempo en 1 (T_ON) para cada caso se encuentra almacenado en una ROM como se explica más adelante.

Para evitar que los leds se dañen, se debe utilizar el sensor para conocer la temperatura de los mismos y ajustar la corriente entregada. En caso de estar por encima de la temperatura máxima t_max, se debe enviar la mitad de la corriente seleccionada (disminuir T_ON a la mitad) hasta que la temperatura baje hasta el óptimo t_opt. La temperatura es un valor de 8bits que está siempre disponible en el sensor.

Los parámetros de funcionamiento se encuentran almacenados en una ROM de configuración de 1K. El contenido de las direcciones más relevantes es:

00h	período de ciclo en us
01h	temperatura máxima
02h	temperatura óptima
11h	T_on en us para nivel 1
12h	T_on en us para nivel 2
13h	T_on en us para nivel 3

Como base de tiempo para la generación del PWM se debe utilizar una seña TIC que tiene un flanco de bajada cada 1us y debe interrumpir al microprocesador.

Se pide:

a) Hardware del sistema que incluya además del chip de ROM de 1K, 16K de ROM y 16K de RAM.

b) Inicialización del sistema, declaración de variables en memoria y programa principal. Este último se encarga de monitorear el estado de los switch y el sensor de temperatura para reportar a la rutina de atención a interrupciones los valores necesarios para generar la señal PWM.

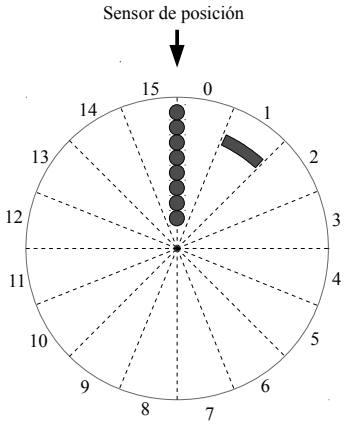
c) Rutina de atención a interrupciones encargada de generar la señal PWM. Los cambios de período y tiempos de 1 y 0 se actualizan luego de finalizado cada período PWM.

FACULTAD DE INGENIERIA INTRODUCCION A LOS MICROPROCESADORES
UNIVERSIDAD DE LA REPUBLICA FEBRERO 2012

PROBLEMA 2

Se desea construir un dispositivo que permita mostrar imágenes dentro de un círculo, basado en el principio POV (persistence of vision). Para esto se hace girar un disco con una tira de 8 leds dispuestos sobre uno de sus radios, y se van encendiendo/apagando los leds a medida que estos pasan por cada uno de los 16 sectores equiespaciados.

En la figura se muestran: los 16 sectores (separados por lineas punteadas); y como se vería si se encendiera el led Nº 6 (numerándolos de 0 a 7) en el sector 1.



Si la velocidad de giro de la tira de leds es suficientemente alta (aprox. 25 vueltas por segundo), el ojo humano no detecta el giro de la tira de leds, con lo que se consigue una imagen “fija”.

Para la correcta visualización de las imágenes (en la vuelta n) se debe estimar la velocidad actual de giro midiendo el tiempo de una vuelta completa (en la vuelta n-1), para ir encendiendo/apagando los leds al comienzo de cada uno de los sectores.

Para la medición del tiempo de una vuelta, se utiliza un sensor de posición sens_pos, que indica cuándo se alcanzó la posición 0 mediante un pulso de bajada.

Los datos de la imagen a mostrar en la “pantalla” se encuentran en una tabla tbl_led de 16 bytes en RAM. El contenido de la tabla indica el estados de los 8 leds ('0'-apagado, '1'-encendido) para cada uno de los sectores, de 0 a 15. En la dirección tbl_led se encuentra el dato de la posición 0, y en cada byte el bit mas significativo corresponde al estado del led exterior de la tira.

Dado que el programa principal debe realizar otras tareas, las funciones pedidas deben implementarse mediante interrupciones. Para la temporización y generación de interrupciones se deberá utilizar un CTC. Se utilizará el canal 0 programado en modo counter para generar una interrupción con cada pulso del sensor sens_pos, el canal 1 para medir el tiempo de vuelta y el canal 2 para generar una interrupción al comienzo de cada sector.

La velocidad de giro nominal es de 25 vueltas/seg, (tiempo de vuelta nominal = 40ms). Si la velocidad de giro es muy lenta el timer utilizado para medir el tiempo de vuelta desbordará. En ese caso no deben encenderse los leds durante la vuelta siguiente. El reloj del sistema es de 1MHz (T = 1us). Notar que:

1 us * 256 * 156,25 = 40 ms - por lo que un canal es suficiente utilizando prescaler = 256
1 us * 256 * 256 = 65,536 ms - el timer desbordará con tiempo de vuelta mayor o igual a 65,536 ms (velocidades de giro menores que 15 vueltas/seg aproximadamente).

Notar que el tiempo correspondiente a cada sector (1/16 del tiempo de vuelta anterior) se obtiene fácilmente cargando la duración medida con el canal 1 pero utilizando el prescaler = 16 en el canal 2. (256/16= 16).

Se pide:
a) Hardware completo del sistema. El sistema debe contar con 16kB de ROM y 16kB de RAM, en las direcciones más bajas, debiendo quedar los restantes 32 kB del espacio de memoria reservados para futuras ampliaciones. Además de los puertos necesarios para la función descrita el sistema debe equiparse con dos puertos de entrada (p_in0 y p_in1) y dos puertos de salida (p_out0 y p_out1), todos de 8 bits.

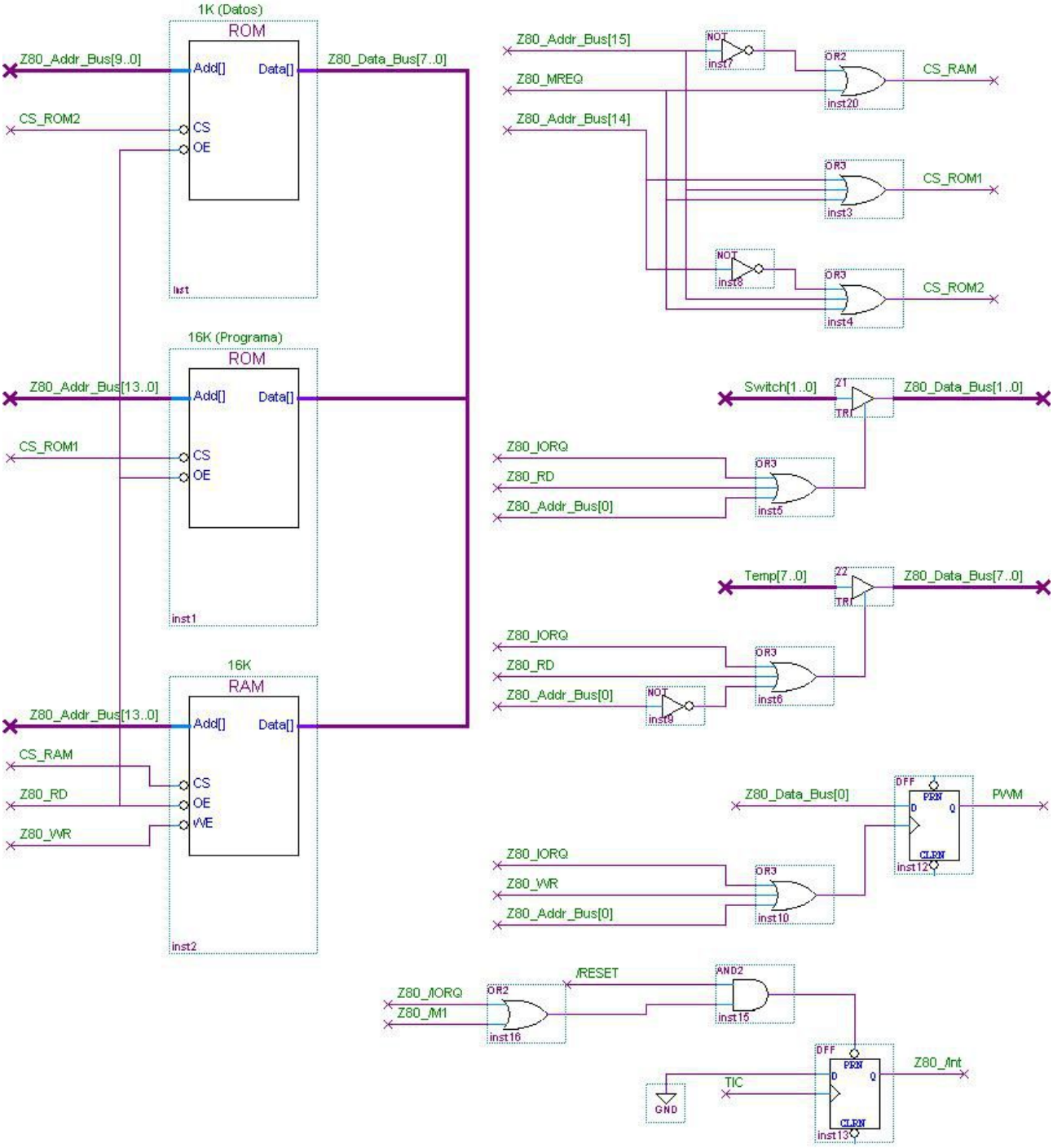
b) Rutinas de atención a las interrupciones:
b.i) Desborde de timer usado para medir tiempo de vuelta. Si se produce el disco está girando a una velocidad menor a la mínima requerida. Deberá indicarlo seteando una bandera que pueda ser consultada por las otras interrupciones.
b.ii) Pulso en sens_pos, debe medir el tiempo de vuelta anterior y configurar el timer de tiempo de sector.
b.iii) Timer de tiempo de sector que realiza el encendido/apagado de los leds.

c) Software de inicialización y reservas de memoria.

Observaciones:
La posicion 0 no es necesario “temporizarla”.

SOLUCIÓN PROBLEMA 1

a) Al no existir restricciones para las direcciones de los puertos los coloco en donde me quede mejor. Tampoco se especifica que no deban existir copias fantasma de ROM o RAM, uso la decodificación mas sencilla.



b) Las direcciones de memoria, direcciones de puerto y direcciones para reserva en RAM, dependen del hardware diseñado en la parte a).

```
;direcciones de memoria
PerDir          EQU             4000h
TmaxDir         EQU             4001h
ToptDir         EQU             4002h
TonDirIni       EQU             4011h

;direcciones de puertos
SwIO            EQU             00h             ;puerto de entrada: switches
TempIO          EQU             01h             ;puerto de entrada: temperatura
PwmIO           EQU             00h             ;puerto de salida: pwm

;reserva de memoria para variables en RAM
                ORG             C000h
PWM:            DB                              ;proximo valor PWM a generar
PWMRUT:         DB                              ;valor PWM a generar
CONT:           DB                              ;contador de ciclos
OVRTMP:         DB                              ;flag de exceso de temperatura

                ;inicializacion del micro
                ORG             0000h           ;codigo en ROM
                LD              SP, 0000h       ; inicializo el stack pointer
                IM              1               ; interrupciones en modo 1

                ;inicializo el sistema y variables
                LD              A,00h
                LD              (PWM), A
                LD              (PWMRUT), A
                LD              (CONT), A
                LD              (OVRTMP), A
                LD              A,01h
                OUT             (PWMIO), A      ;prendo los leds
                EI                              ;habilito las INT
; forever{
;       si (sw=00) entonces { pwm = 0 }
;       else {
;               pwmtmp = tabla[sw]
;               si ( t >= tmax ) entonces { ovrtmp = 1 }
;               si ( t =< topt ) entonces { ovrtmp = 0 }
;               si (ovrtmp == 1) entonces { pwmtmp = vartmp / 2 }
;               pwm = pwmtmp
;               }
; }
                ;loop principal
Loop:           IN              A, (SwIO)       ;leo los switches
                AND             A,03h
                JP              NZ, Else

                LD              (PWM), 00h
                JP              Loop            ;si (sw=00) entonces { pwm = 0 }

Else:           LD              HL, TonDirIni   ;inicializo el puntero a la tabla
                LD              L, A
                LD              A, (HL)         ;leo T_on de la tabla
                LD              C, A            ;guardo el valor en registro C

                ;control de la temperatura
ContrTemp:      IN              A, (TempIO)     ;leo la temepratura
                LD              B, A
                LD              A, (TmaxDir)
                CP              B               ;hago (Tmax - TempActual)
                JP              P, TmaxBien     ;si >= 0 (es decir TempActual =< Tmax) salto

                LD              A, 0ffh
                LD              (OVRTMP), A     ;si < 0 (es decir TempActual > Tmax) prendo la bandera

TmaxBien:       LD              A, (ToptDir)
                CP              B               ;hago (Toptima – TempActual), en B tengo  TempActual
                JP              M, mayortopt    ;si < 0 (es decir TempActual > Toptima) salto

                LD              A, 00h
                LD              (OVRTMP), A     ;reseteo la bandera


mayortop:       LD              A, (OVRTMP)
                OR              A
                JP              Z, Fin          ;si no hubo calentamiento previo salto
```

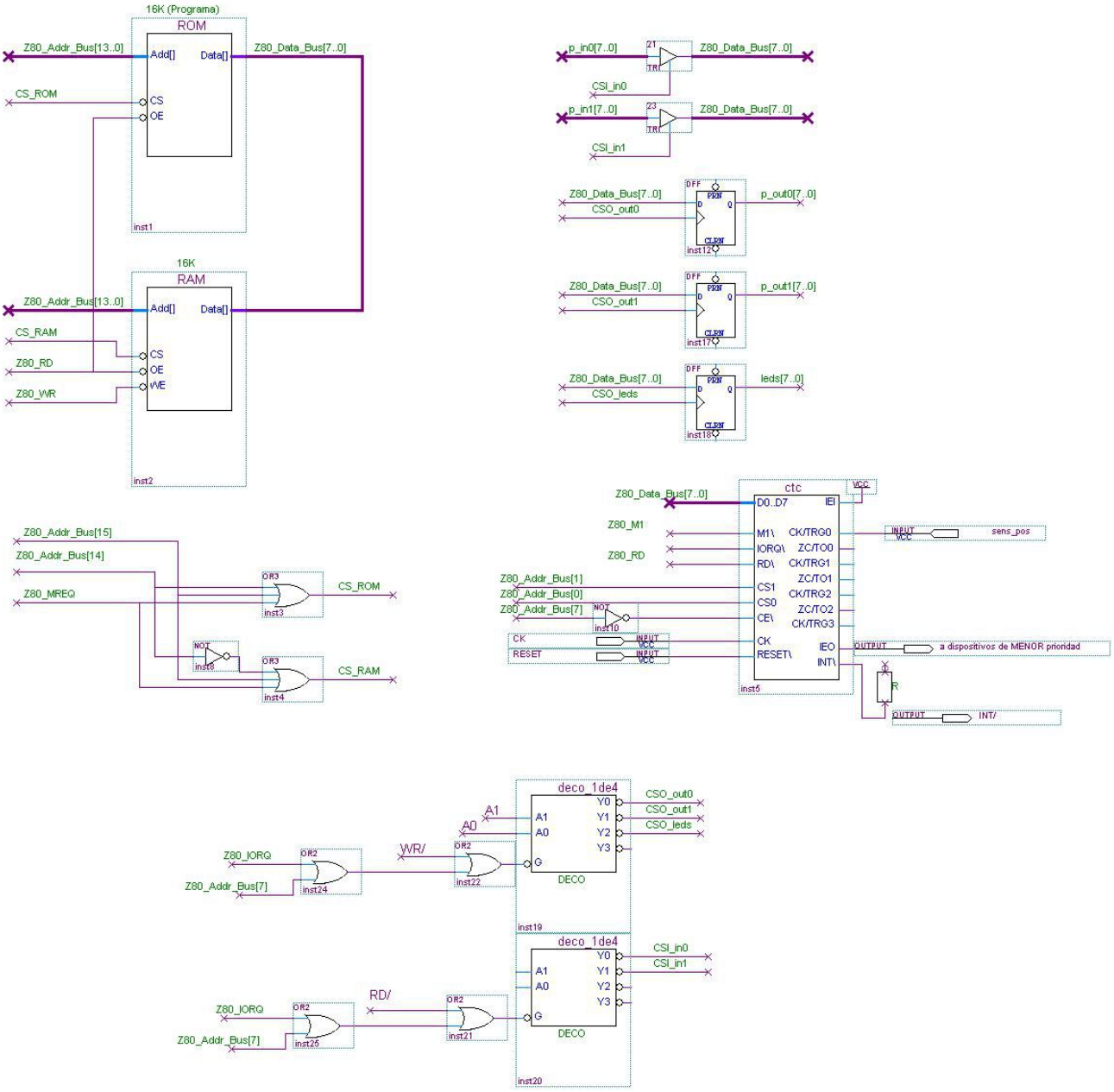
```
                SRL     C               ; divido por dos el valor leído de la tabla
                JP      Fin             ;voy a fin

Fin:            LD      A, C
                LD      (PWM), A
                JP      Loop
```

c) La rutina recibe los parámetros a través de las variables PWM, PWMRUT. Lleva el contador de cantidad de interrupciones CONT, para saber cuando finalizo un ciclo PWM.

- La variable CONT lleva la cuenta de la cantidad de interrupciones que han ocurrido y se usa para saber si se completo la cantidad de T_on. Se incrementa en cada INT.
- La variable PWMRUT es el valor PWM que debe generar la rutina, se actualiza con el valor PWM una vez completado un ciclo PWM.
- La variable PWM almacena el próximo valor PWM a generar, una vez que se completa un ciclo PWM. Es actualizada por el programa principal.

```
;rutina de atención a las interrupciones
                ORG     0038h

                PUSH    AF              ;preservo AF
                PUSH    BC              ;preservo BC

                LD      A, (CONT)
                LD      B, A
                LB      A, (PWMRUT)
                CP      B               ;hago (PWMRUT - CONT)
                JP      P, FinRut       ;si > 0 (o sea CONT < PWMRUT) no transcurrio T_ON

TranscurrioTON:
                LD      A, 00h
                OUT     (PwmIO), A      ;si <= 0 (o sea CONT >= PWMRUT), apago los leds

ContrCiclo:     LD      A, (PerDir)
                LD      B, A
                LD      A, (CONT)
                CP      B               ;hago (CONT - Periodos_us)
                JP      M, FinRut       ;si < 0 (o sea CONT < Periodos_us) salto

TranscurrioCiclo:
                LD      A,. 01h
                OUT     (PwmIO), A      ;si >= 0 (o sea CONT >= PWMRUT), prendo los leds

                LD      A, (PWM)
                LD      (PWMRUT), A     ;leo el proximo PWM a generar y lo cargo en el usado por la rutina

FinRut:         LD      A, (CONT)
                INC     A
                LD      (CONT), A       ;incremento el contador de INT

                POP     BC              ;recupero BC
                POP     AF              ;recupero AF
                EI
                RET                     ;retorno
```

SOLUCIÓN PROBLEMA 2



b) rutinas atención int.

```
CTC0 equ 80h
CTC1 equ 81h
CTC2 equ 82h
LEDS equ 03h
CTC0_CW    equ 11X0X111 ; int_ena, counter, X, falling edge, X, constant follows, reset
CTC1_CW    equ 101X0111 ; int_ena, timer, pre_256, X, auto_trg,  constant follows, reset
CTC2_CW_EI equ 100X0111 ; int_ena, timer, pre_16, X, auto_trg,  constant follows, reset
CTC2_CW_DI  equ  000X0011  ;  int_dis,  timer,  pre_16,  X,  auto_trg,   NO constant follows,
reset

ORG 2000h    ; algún lugar en ROM
;----------------------------
iar_overrun_detection:
 ei
 push af
 ld a, 0ffh
 ld (GIRO_LENTO), a
 pop af
 reti

;----------------------------
iar_one_turn_detection:
 ei
 push af
 push bc
 ld a,(GIRO_LENTO)           ; hubo overrun?
 cp a, 0ffh
 jp z, ultimo_giro_lento

ultimo_giro_ok:
 in a,(CTC1)
 neg a
 ld b,a                      ; b= duración de vuelta
 ld a, CTC2_CW_EI            ; cargo duracion en CTC2 aue lo divide en 16
 out (CTC2), a
 ld a,b
 out (CTC2), A
 jp fin_one_turn_detection

ultimo_giro_lento:
 ld a, CTC2_CW_DI            ; deshabilito interrupciones de sectores
 out (CTC2), a
 ld a,0
 ld (GIRO_LENTO), a          ; borro flag de giro lento

fin_one_turn_detection:
 ld a, CTC1_CW               ; reprogramo timer de vuelta empezando de 0
 out (CTC1), a
 ld a, 0
 out (CTC1), a
 ld a, 0
 ld (CURRENT_SECTOR),a
 pop bc
 pop af
 reti

;----------------------------
iar_sector:
 ei
 push af
 push hl
 push bc
 ld a,(CURRENT_SECTOR)
 ld bc, 00
 ld c,a
 inc a
 and 0Fh
 ld (CURRENT_SECTOR),a
 ld hl,tbl_led
 add hl,bc
 ld a,(hl)
 out (LEDS),a
```

```
 pop bc
 pop hl
 pop af
 reti
```

c) sw inicialización

```
ORG 4000h ; comienzo ram
GIRO_LENTO DB
CURRENT_SECTOR DB

org 3000h ; medio de rom
tabla_int:
 DW iar_one_turn_detection    ; CTC0
 DW iar_overrun_detection     ; CTC1
 DW iar_sector                ; CTC2

org 0000h
;inicializo modo 2
ld sp, 8000h ;fin ram + 1
ld i, tabla_int / 256
im2

; inicializo variables
ld a,0
ld (CURRENT_SECTOR),a
ld a,0FFh
ld (GIRO_LENTO),a  ; inicializo para descartar primer giro ya que será lento

;vector de interrupciones
ld a,0
out (CTC0), a ; vector de int en 0
; inicializo CTC0 ya que el resto se inicializan luego de dar la primera vuelta
ld a, CTC0_CW
out (CTC0),a
ld a,1
out (CTC0), a ; cte arranca en 1, genera interrpcion al primer flanco

jp ppal      ; salta al programa principal que realiza otras tareas.
```