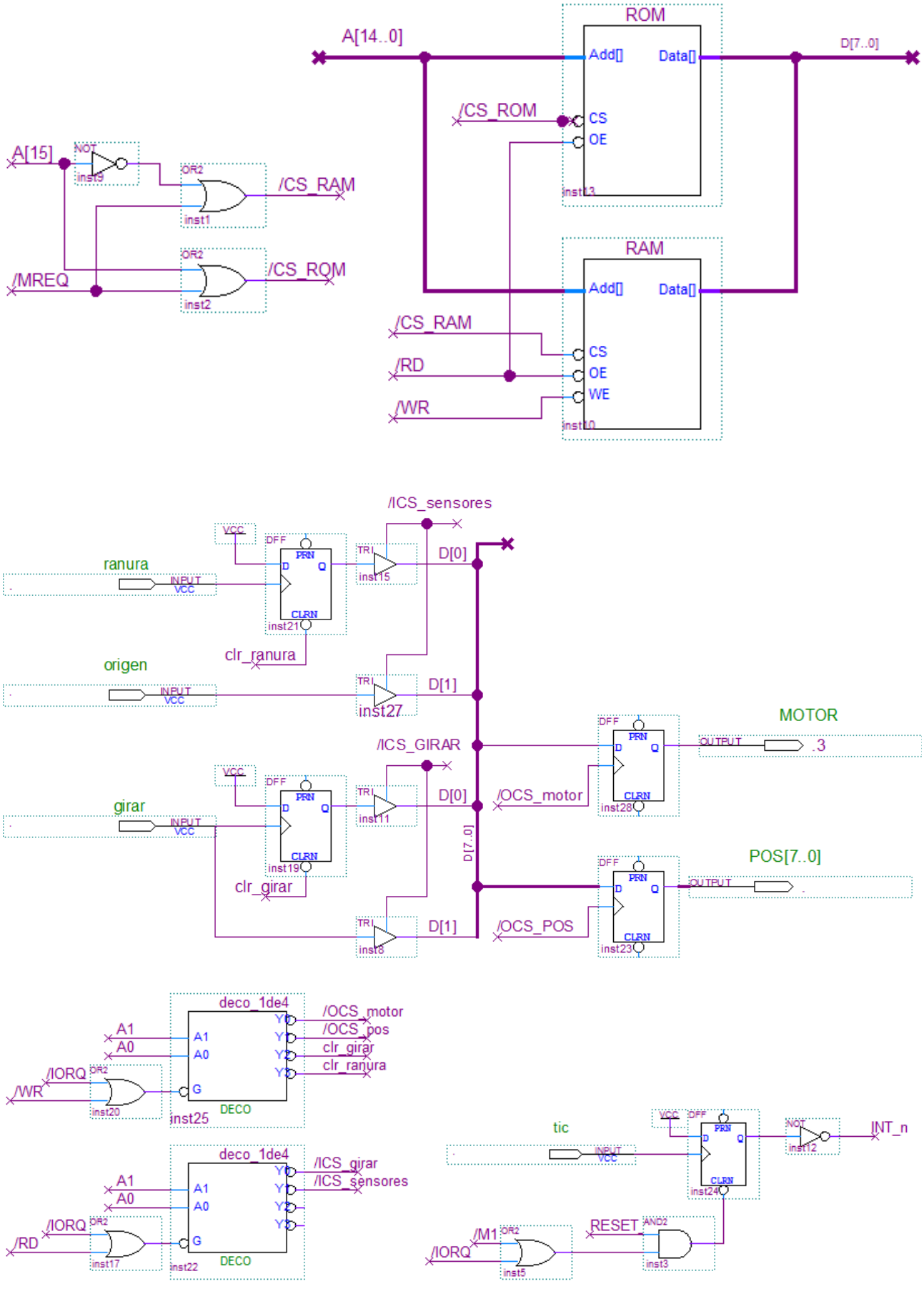
PROBLEMA 1 - a) Hardware



b)Rutina de servicio de interrupcion

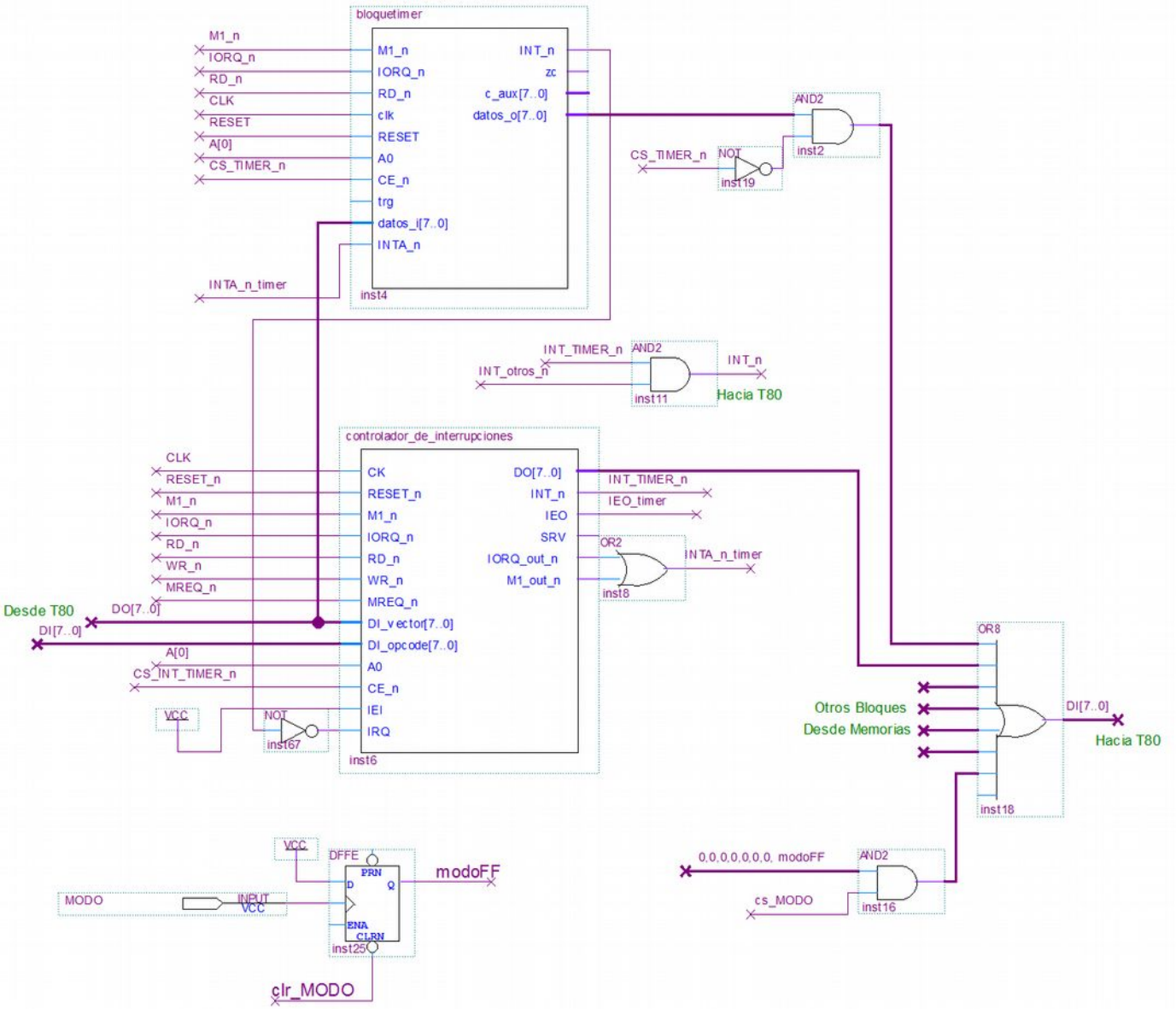
```
;;; Puertos entrada
GIRAR    equ  0 ; b1: directo, b0: FF
SENSORES equ  1 ; b1: orig. b0: ranura

;;; Puertos salida
MOTOR      equ  0 ; 1 bits
POS        equ  1 ; 8 bit

;;; pulsos salida
clr_girar  equ  2 ;
clr_ranura equ  3 ;

;;; Constantes
E_IDLE     equ  0
E_CORTO    equ  1
E_LARGO    equ  2

; --- rutina de atencion a interrupcion
; preservar registros
; switch estado {
; idle:
;    cnt=5
;    if girar then
;       enciendo motor
;       estado=corto
;    endif
; corto:
;    if (girar == 0) then
;       apago motor
;       estado = idle
;    else
;       cnt--
;       if cnt == 0 then
;          estado = largo
;          borro flanco
;    endif
; largo:
;    if (girarFF) then
;       apago motor
;       estado = idle
;       borro girarFF
;    endif
; }
; restauro registros
; habilito interrupciones
; retorno

      ORG 38H
rutint:
      push af
      ld a, (estado)
      cp a, E_IDLE
      jr nz, sigo1
; idle:
;    cnt=5
;    if girar then
;       enciendo motor
;       estado=corto
;    endif
      ld a, 5
      ld (cnt), a
      in a, (GIRAR)
      bit 1, a
      jr z, fin_switch
      ld a, 0xff
      out (MOTOR), a
      ld a, E_CORTO
      ld (estado), a
sigo1:
      cp a, E_CORTO
      jr nz, sigo2
; corto:
;    if (girar == 0) then
;       apago motor
;       estado = idle
;    else
;       cnt--
;       if cnt == 0 then
;          estado = largo
;          borro girarFF
;    endif
      in a, (GIRAR)
      bit 1, a
      jr nz, else
      ld a, 0
      out (MOTOR), a
      ld a, E_IDLE
      ld (estado), a
else:
      dec (cnt)
      jr nz, fin_switch
      ld a, E_LARGO
      ld (estado), a
      out (clr_girar), a
sigo2:
; largo:
;    if (girarFF) then
;       apago motor
;       estado = idle
;       borro girarFF
;    endif
      in a, (GIRAR)
      bit 0, a
      jr z, fin_switch
      ld a, 0
      out (MOTOR), a
      ld a, E_IDLE
      ld (estado), a
      out (clr_girar), a
fin_switch:
      pop af
      ei
      ret
```

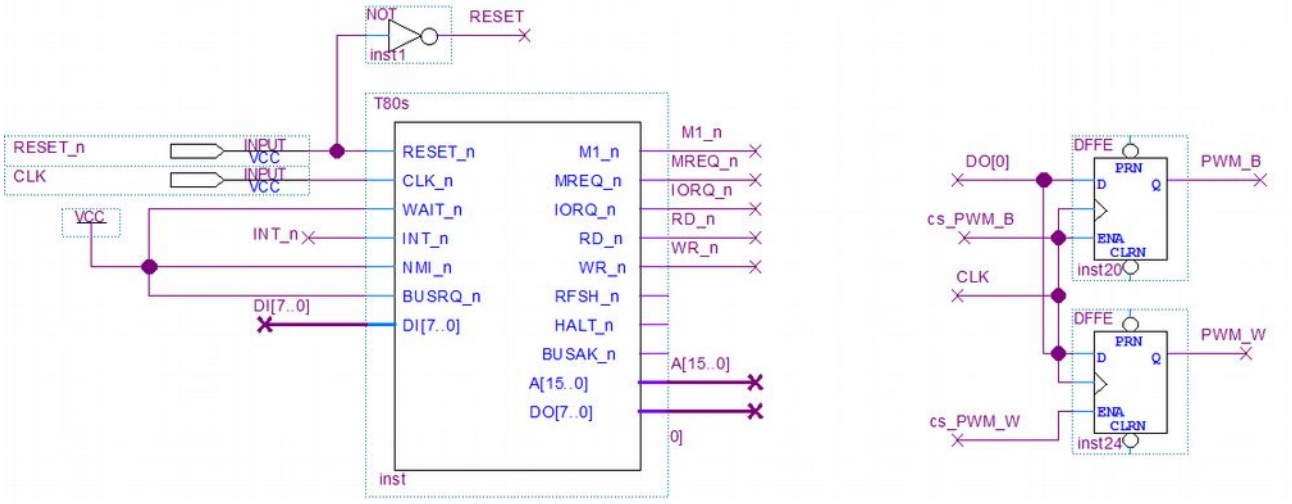
c) inicializ. y prog. Principal

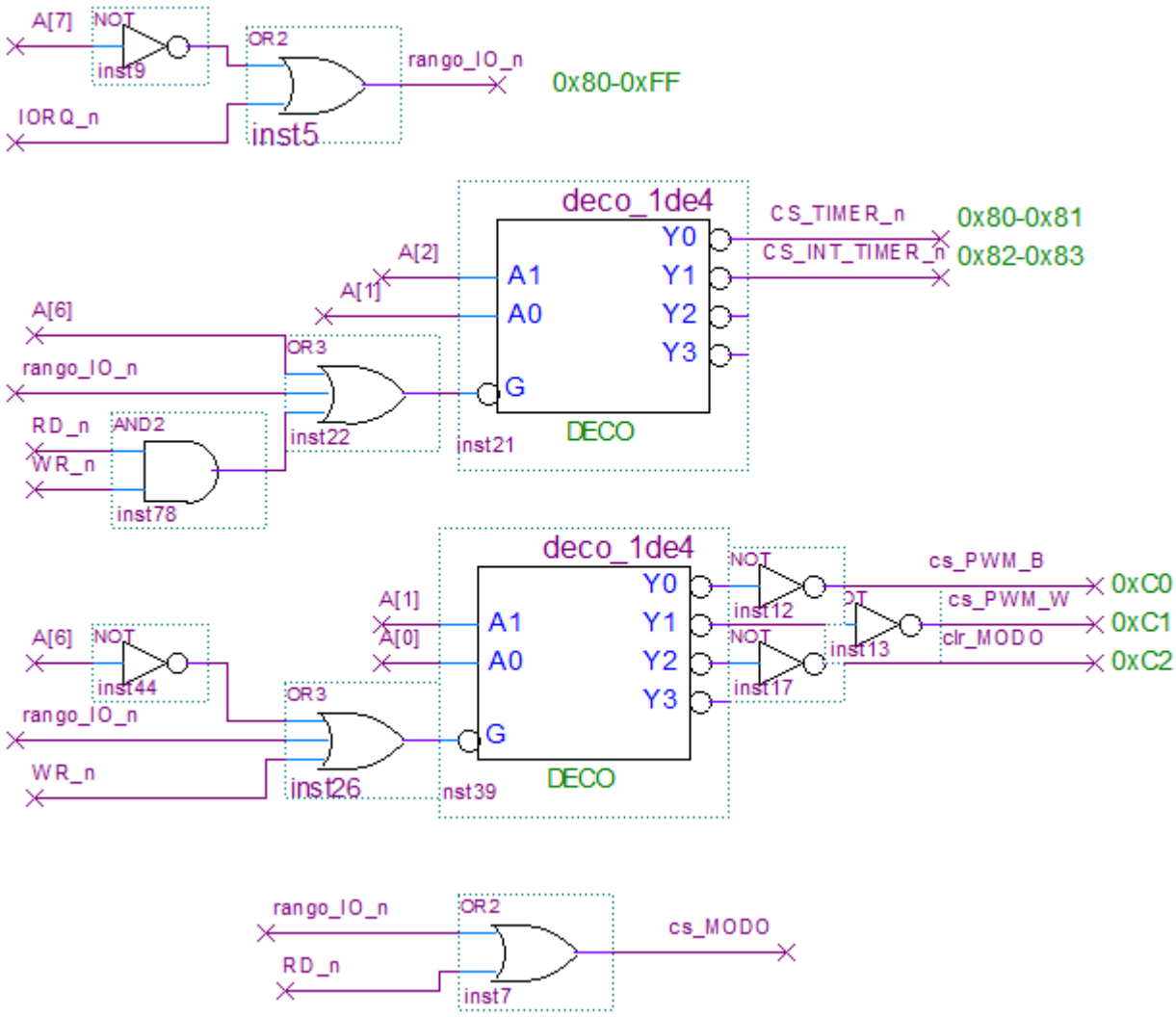
```
;;;   Inicializacion
;   modo 1
;   sp
;   estado = idle
;   cnt = 5
;   pos = 0xFF
;   enciendo motor
;   espero origen
;
;   pos = 0
;   borrar irq
;   borrar ffs flancos
;   habilito interrupciones
;
      im 1
      ld sp, 0
      ld a, E_IDLE
      ld (estado), a
      ld a, 5
      ld (cnt), a
      ld a, 0xFF
      out (POS), a
      ld (varpos),a
      call buscar_origen
      ld a, 0
      out (POS), a
      ld (varpos), a
      out (clr_girar), a
      out (clr_ranura), a
      ei
      jr ppal
```

```
; --- Programa principal
; forever{
;    si origen then pos=0
;    si flanco then
;       pos++;
;       borro flanco
; }

      ORG 0x100
ppal:
      in a, (SENSORES)
      bit 1, a
      jr z, ver_ranura
      ld a, 0
      ld (varpos), a
      out (POS), a
ver_ranura:
      bit 0, a
      jr nz, fin_forever
      ld a, (varpos)
      inc a
      ld (varpos), a
      out (POS), a
fin_forever:
      jr ppal

;;;;;;;; --- subrutina buscar_origen
buscar_origen:
      ld a, 1
      out (MOTOR), a
espero:
      in a, (SENSORES)
      bit 1, a
      jr z, espero
      ld a, 0
      out (MOTOR), a
      ret

;;;;;;;; --- reserva memoria

      ORG 0x8000
estado:  db
varpos:  db
cnt:     db
```

PROBLEMA 2 – Parte a) Hardware






A[7]
NOT
inst9
OR2
rango_IO_n
0x80-0xFF
IORQ_n
inst5
deco_1de4
CS_TIMER_n
0x80-0x81
CS_INT_TIMER_n
0x82-0x83
A[2]
A[1]
A1
A0
G
Y0
Y1
Y2
Y3
DECO
inst21
A[6]
rango_IO_n
OR3
inst22
RD_n
WR_n
AND2
inst78
deco_1de4
A[1]
A[0]
A[6]
NOT
inst44
rango_IO_n
WR_n
OR3
inst26
nst39
DECO
NOT
inst12
NOT
inst17
OT
inst13
cs_PWM_B
cs_PWM_W
clr_MODO
0xC0
0xC1
0xC2
rango_IO_n
RD_n
OR2
inst7
cs_MODO

PROBLEMA 2 – Software

b) Inicialización:

```
; direcciones IO
CS_MODO  equ 0xC2
CLR_MODO equ 0xC2
CS_PWM_W equ 0xC1
CS_PWM_B equ 0xC0
CS_INT_TIMER equ 0x82
CS_TIMER  equ 0x80

;constantes
CTE_TIMER EQU 200
CW_TIMER EQU 1011 0101 ; I=1 |
T_flanco=x | SW_reset=1 | T_auto = 1 |
Prescaler= 5
VI_TIMER EQU 0x04

ORG 0x1000
Tabla_modos:
  db 0 ;- Apagado W
  db 0 ;- Apagado B
  db 6 ;- Modo Calido W
  db 0 ;- Modo Calido B
  db 6 ;- Modo Neutro W
  db 2 ;- Modo Neutro B
  db 5 ;- Modo Frio W
  db 4 ;- Modo Frio B

ORG 0x8000
 offset_modo db
 paso_pwm db

ORG 0x9000
tabla_int:
 DW
 DW
 DW

org 0x0000
; inicializo stack y modo de
interrupciones
 ld SP, 0x0000
 im2

 ld A,tabla_int/256
 ld I,A
 ld HL, rutint_timer
 ld (tabla_int + VI_TIMER), HL
 call INI_OTRAS_INT

 ld A,VI_TIMER
 out (CS_INT_TIMER), A ; cargo vector de
interruciones en timer
 out (CS_INT_TIMER+1), A ; boro
peticiones pendientes

; borro flags que capruran flancos de
modo
 out CLR_MODO, A

; inicializo timer
 call init_timer

; modo de trabajo inicial
 ld A, 2
 ld (offset_modo),A

; habilito interrupciones y salto a
principal
 ei
 jp ppal


init_timer:
 ; inicializo timer para que interrumpa
con Tpaso = 128 us
   ld A, CTE_TIMER
   out (CS_TIMER), A
   ld A, CW_TIMER
   out (CS_TIMER +1)
 ; inicializa variable que lleva cuenta
de
 ; en que paso de pwm esta la
interrupcion
   ld A,0
   ld (paso_pwm),A
 ret

org 0x100
ppal:
 ; escaneo puertos para detectar cambio
de modo
   in A, (IE_MODO)
   jp Z, ppal

nuevo_modo:
 ; borro flag
   out (CL_MODO), A
 ; deplazo offset de tabla de modo 2
lugares
   ld A,(offset_modo)
   add A,2
   and 0x07
   ld (offset_modo),A

   call init_timer
   jp ppal
```

c) Rutinas de atención a interrupciones

```
org 0x500h
 ei
 push AF
 ld HL, tabla_modos
 ld A, (offset_modo)
 ld L,A
 add A,2
 and A,7
 ld (offset_modo),A

 ; salida PWM_W
 ld B,(HL) ; B= paso a partir del cual
la señal vale 0

 ld A, (paso_pwm); A= paso actual en
período PWM
 cp B  ; paso_pwm - PWM_W. Si <0 =>
salida PWM = 1
 jp M, PWM_W_1
PWM_W_0:
 ld A,0
 jp salida_PWM_W
PWM_W_1:
 ld A,1
salida_PWM_W:

 out (CS_PWM_W), A

 ; salida PWM_B
 inc L
 ld B,(HL) ; B= paso a partir del cual
la señal vale 0

 ld A, (paso_pwm); A= paso actual en
período PWM
 cp B  ; paso_pwm - PWM_B. Si <0 =>
salida PWM = 1
 jp M, PWM_B_1
PWM_B_0:
 ld A,0
 jp salida_PWM_B
PWM_B_1:
 ld A,1
salida_PWM_B:
 out (CS_PWM_B), A

 pop AF
 reti
```