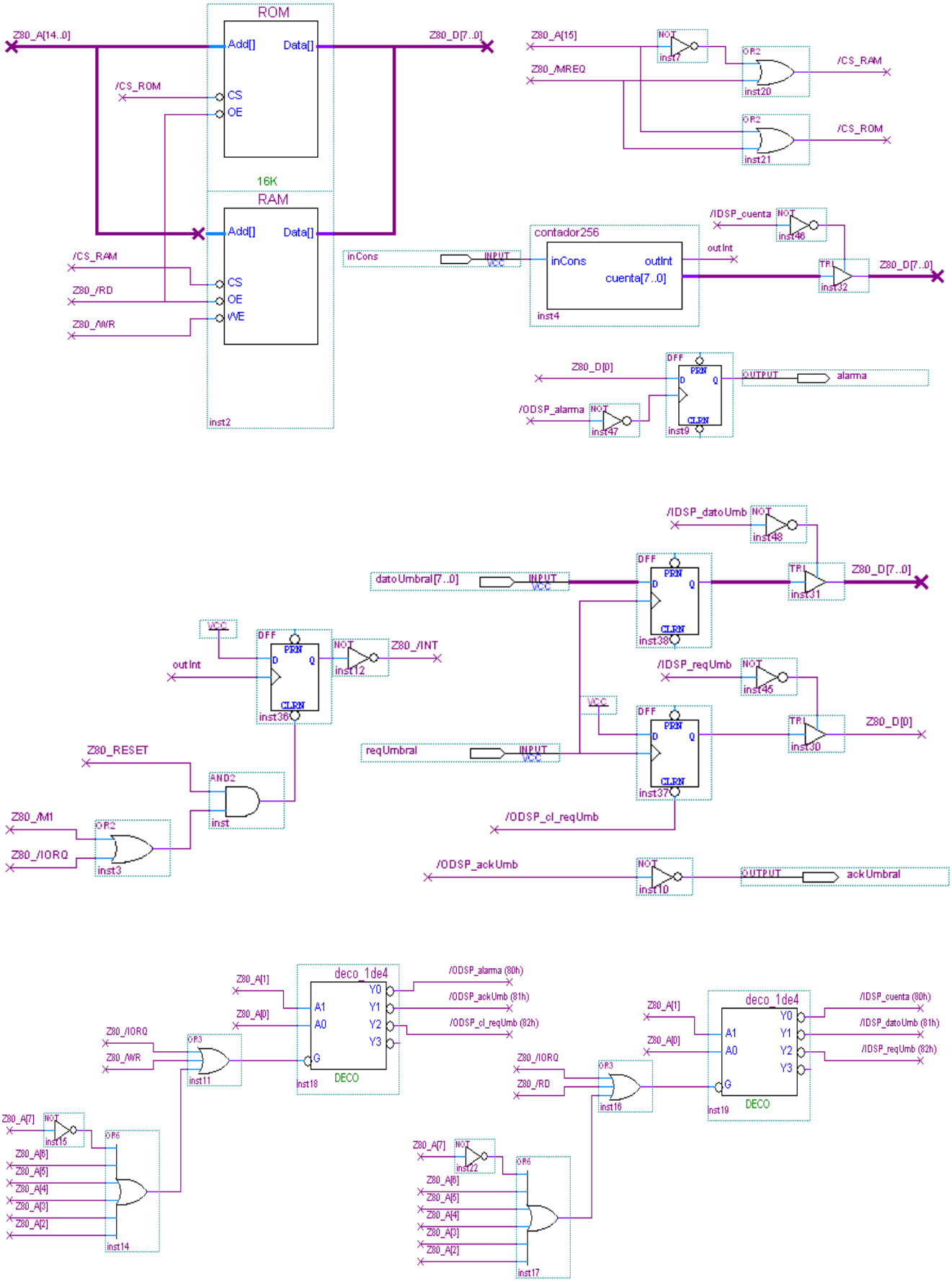
PROBLEMA 1 – Solución

Parte a)



Parte b)

```
ORG 0x0038
rutInt:
      PUSH AF
      PUSH HL
      PUSH BC
;; si estoy en overflow retorno
      LD A, (over_flow)
      CP 0xFF
      JP Z, fin
;; comparo antes de incrementar
;; si es igual al umbral
      LD A, (num_int)
      LD HL, umbral
      CP (HL)
      JP M, else
;; prende alarma
      LD B, A
      LD A, 1
      OUT (alarma), A
      jr fin_si:
else:
;; apaga alarma
      LD B, A
      LD A, 0
      OUT (alarma), A
fin_si:
;; si num_int = 255
;; overflow y no increm.
      LD A, B
      CP 255
      JP NZ, else2
;; desbordamiento
      LD A, 0xFF
      LD (over_flow), A
      JP fin
else2:
      INC A
      LD (num_int), A
fin:
      POP BC
      POP HL
      POP AF
      EI
      RET
```

Parte c)

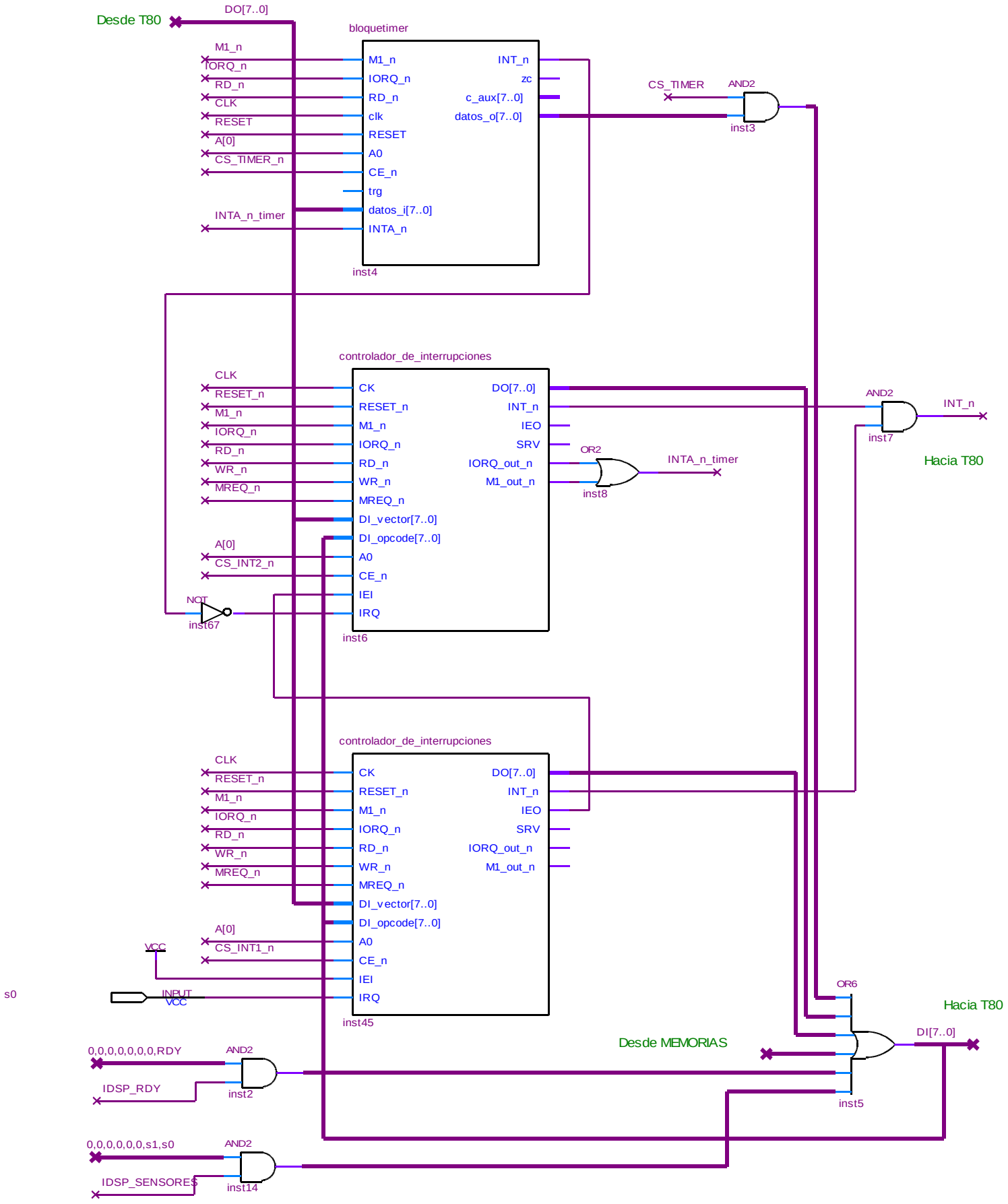
```
ORG ALGUN_LUGAR_DE_ROM
consAcum:
      PUSH AF
;; cheque desbordamiento
      LD A, (over_flow)
      CP 0xFF
      JP NZ, no_ovf
      LD BC, 0xFFFF
      JP finn
no_ovf:
      IN A, (cuenta)
      LD C, A
      LD A, (num_int)
      LD B, A
finn:
      POP AF
      RET
```

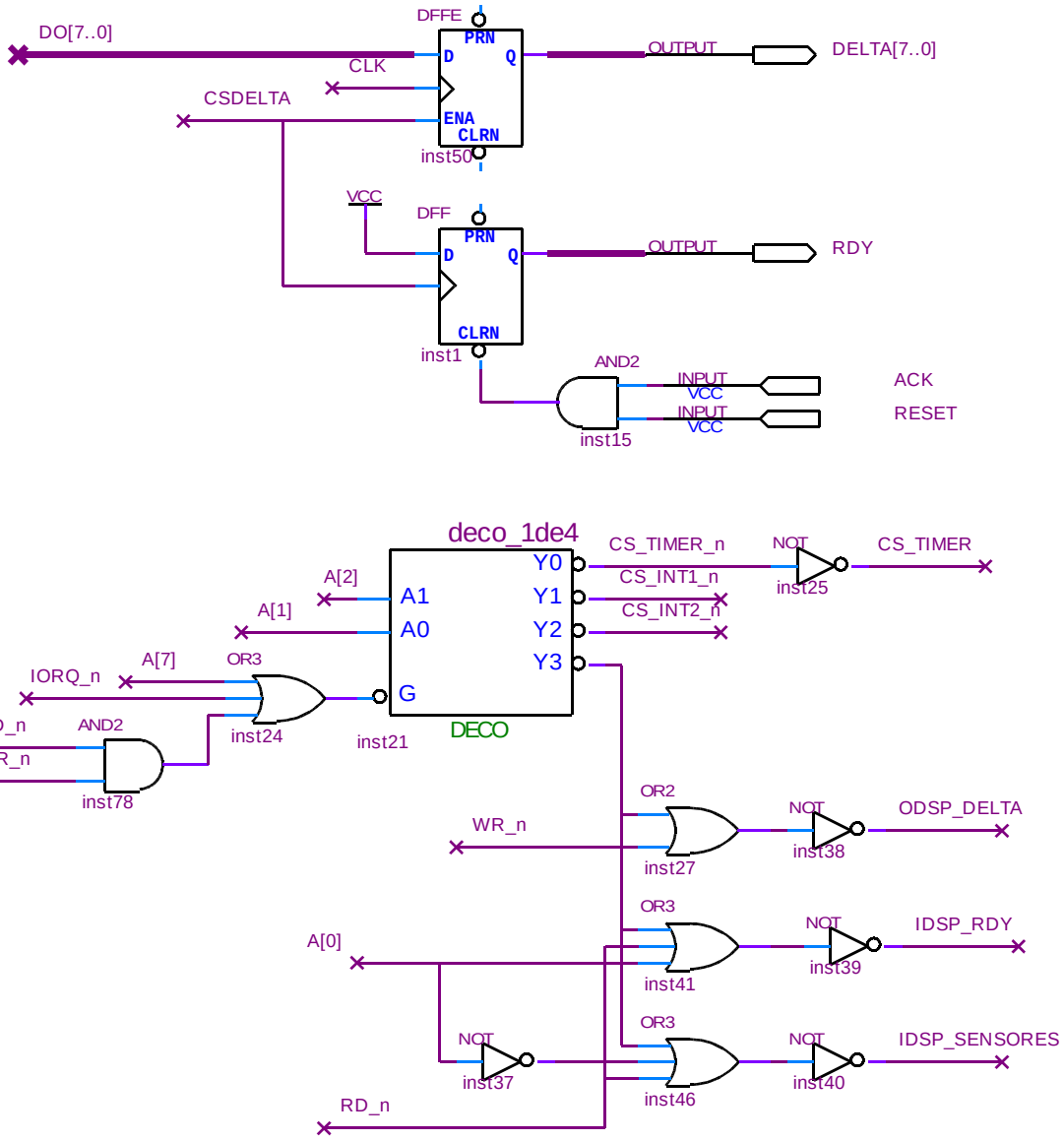
Parte d)

```
reqUmb      equ 0x82
datoUmb     equ 0x81
cuenta      equ 0x80
cl_reqUmb   equ 0x82
ackUmb      equ 0x81
alarma      equ 0x80
UMBRAL_INI  equ ....

ORG 0x8000
num_int:    db
umbral:     db
over_flow:  db

ORG 0x0000
      LD SP, 0
      LD A, 0
      LD (num_int), A
      LD (over_flow), A
      OUT (alarma), A
      OUT (cl_reqUmb), A
      LD A, UMBRAL_INI
      LD (umbral), A
      IM 1
      EI
      JP Loop

ORG OTRO_LUGAR_DE_ROM
verifUmbral:
      PUSH AF
      IN A, (reqUmb)
      AND 0x01
      JP Z, finnn
      IN A, (datoUmb)
      LD (umbral), A
      OUT (cl_reqUmb), A
      OUT (ackUmb), A
finnn:
      POP AF
      RET
```

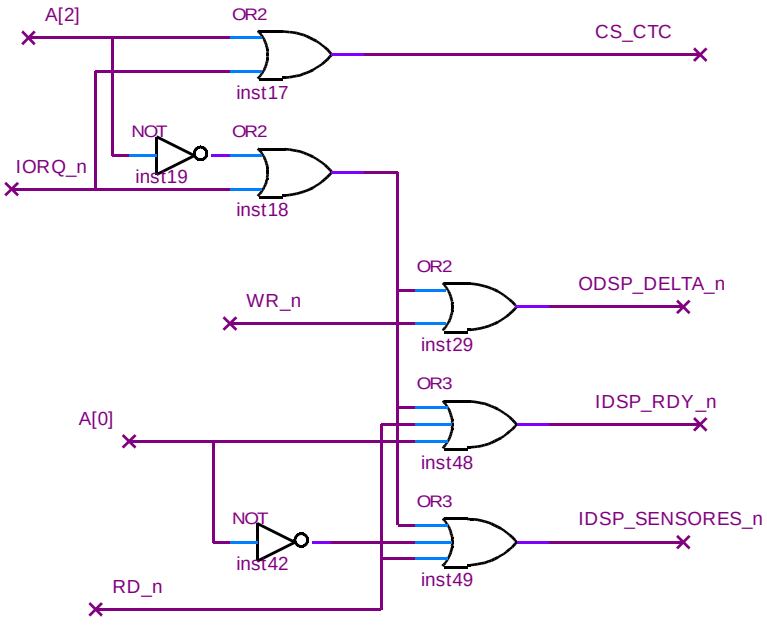
PROBLEMA 2 – Solución

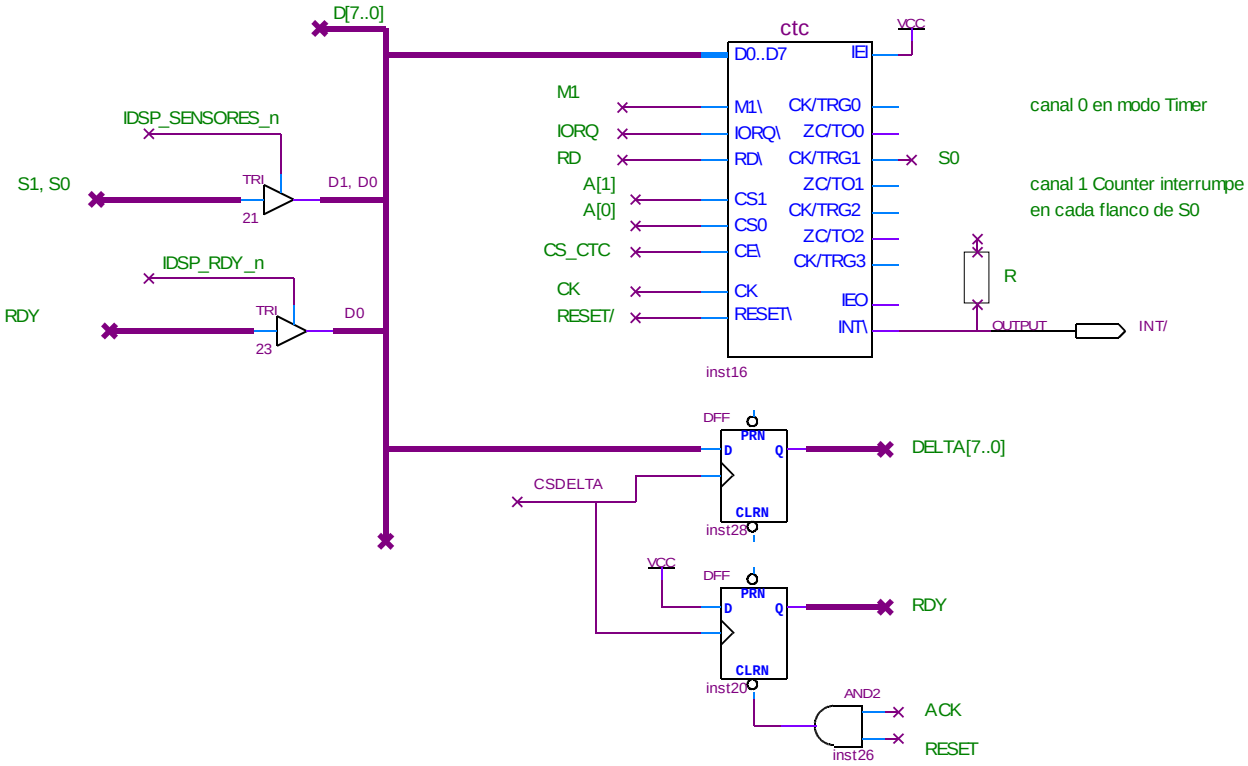
Parte a) Conexionado puertos (caso T80 y bloques lab)





Conexionado puertos - Caso CTC





---

b) Software

b.i) Bucle programa principal

```
; forever{
;  mientras (rdy == 1) espero
;  mientras (flag_TMIN == 0) espero
;  mientras (v_delta == 0) espero
;  p_delta = v_delta
;  v_delta = 0
;  arranco timer
;  flag_TMIN = false
; }
    org 0x0100
main:
   ;; mientras (rdy == 1) espero
  in a, (p_rdy)
  bit 0, a
  jr nz, main
   ;; mientras (flag_TMIN == 0) espero
loop_tmin:
  ld a, (flag_tmin)
  cp 0
  jr z, loop_tmin
    ;; mientras (v_delta == 0) espero
espero_cambio:
  ld a, (v_delta)
  cp 0
  jr z, espero_cambio
    ;; p_delta = v_delta
    ;; v_delta = 0
    ;; arranco timer
    ;; flag_TMIN = false
  out (p_delta), a
  ld a, 0
  ld (v_delta), a
  ld (flag_tmin), a
  call arranco_timer
  jr main
```

b.ii) Rutinas interrupción

```
; isr_timer{
;   flag_TMIN == true
;   deshabilito int timer
; }
isr_timer:
  ei
  push af
  ld a, 0xff
  ld (flag_tmin), a
  call freno_timer
  pop af
  reti

; isr_s0{
;   si s1==0 entonces inc = +1
;   else inc = -1
;   v_delta = v_delta + inc
; }
isr_s0:
  ei
  push af
  push hl
  in a, (p_sensores)
  bit 1, a
  jr nz, else_s1
    ;; si s1==0 entonces inc = +1
  ld a, 1
  jr fin_si_s1
else_s1:
    ;; else inc = -1
  ld a, -1
fin_si_s1:
    ;; v_delta = v_delta + inc
  ld hl, v_delta
  add a, (hl)
```

```
  ld (v_delta), a
  pop hl
  pop af
  reti

;; tabla interrupciones en ROM
;;  y alineada a página
     ORG 0x0200
tabla:
    dw isr_timer
    dw isr_s0

;----------
;; subrutinas
;;--- caso bloques Timer y Contr
init_bloques:
    ;; inic vectores
  ld a, VEC_S0
  out (base_int_next+off_vector), a
  ld a, VEC_TIM
  out (base_int_tim+off_vector), a
    ;; borro peticiones anteriores
  out (base_int_next+off_status), a
  out (base_int_tim+off_status), a
  ret

arranco_timer:
  ld a, TIM_CTE
  out (base_tim + off_cte), a
  ld a, TIM_CTRL
  out (base_tim + off_ctrl), a
  ret

freno_timer:
  ;; deshabilito generación de interr
  ;; por el timer
  ld a, TIM_CTRL_DI
  out (base_tim + off_ctrl), a
  ret
```

```
;----------
;; subrutinas
;; -- caso CTC
init_bloques:
  ;; inic vectores
  ld a, VEC_CTC
  out (CTC_0), a
  ;; inic counter s0
  ld a, COUNTER_CTRL
  out (CTC_1), a
  ld a, COUNTER_CTE
  out (CTC_1), a
  ret

arranco_timer:
  ld a, TIM_CTRL
  out (CTC_0), a
  ld a, TIM_CTE
  out (CTC_0), a
  ret

freno_timer:
  ;; deshabilito generación de interr
  ;; por el timer
  ld a, TIM_CTRL_DI
  out (CTC_0), a
  ret
;; -- fin subrutinas caso CTC

;; variables
  org 0x8000
flag_tmin: db 0
v_delta:   db 0
```

b.iii) Inicializacion y directivas

```
;----------
; CONSTANTES caso BLOQUES
; ports timer
base_tim   equ  0x00
off_cte    equ  0
off_ctrl   equ  1
off_flag   equ  0
off_cuenta equ  1

; ports contr. Int.
base_int_next equ  0x02
base_int_tim  equ  0x04
off_vector    equ  0
off_status    equ  1


; ctes timer
TIM_CTE     equ  125-1
   ; ei, x, rst=1, trg auto, pre = 8
TIM_CTRL    equ  10100100B
   ; di, x, reset=1, xxxxx
TIM_CTRL_DI equ  00100000B
VEC_TIM   equ 0
VEC_S0    equ 2

;----------
; CONSTANTES caso CTC
; ports CTC
CTC_0   equ  0
CTC_1   equ  1

; ctes timer
TIM_CTE     equ  125
   ; ei, timer, pre=256, x,
   ; autotrg, sigue cte, sw reset
TIM_CTRL    equ  10100111B
   ; di, no sigue cte
TIM_CTRL_DI equ  00100011B
VEC_CTC     equ 0

; ctes contador
   ; ei, cntr, x, flanco subida,
   ; x, sigue cte, sw reset
COUNTER_CTRL equ 11010111B
COUNTER_CTE  equ 1

; puertos (ambos casos)
p_delta    EQU 6
p_rdy      EQU 6
p_sensores EQU 7
```

```
  ORG 0
ld sp, 0
im 2
ld hl, tabla
ld a, h
ld i, a
  ;; flag_tmin = false
  ;; v_delta = 0
ld a, 0
ld (flag_tmin), a
ld (v_delta), a
  ;; RDY inactivo por hw
  ;; inicializo Contr. Int.
  ;; o CTC
call init_bloques
ei
jp main
```