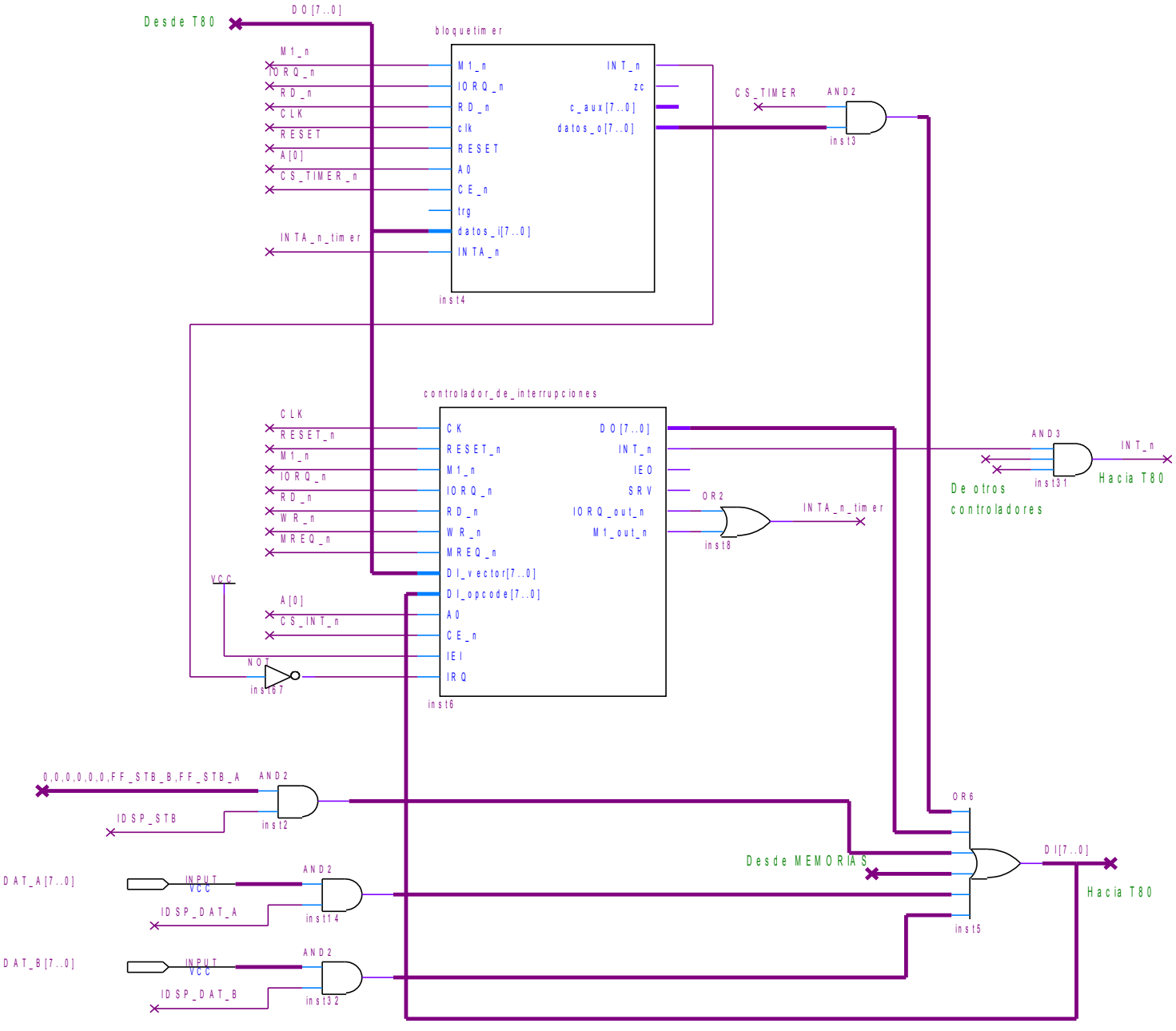
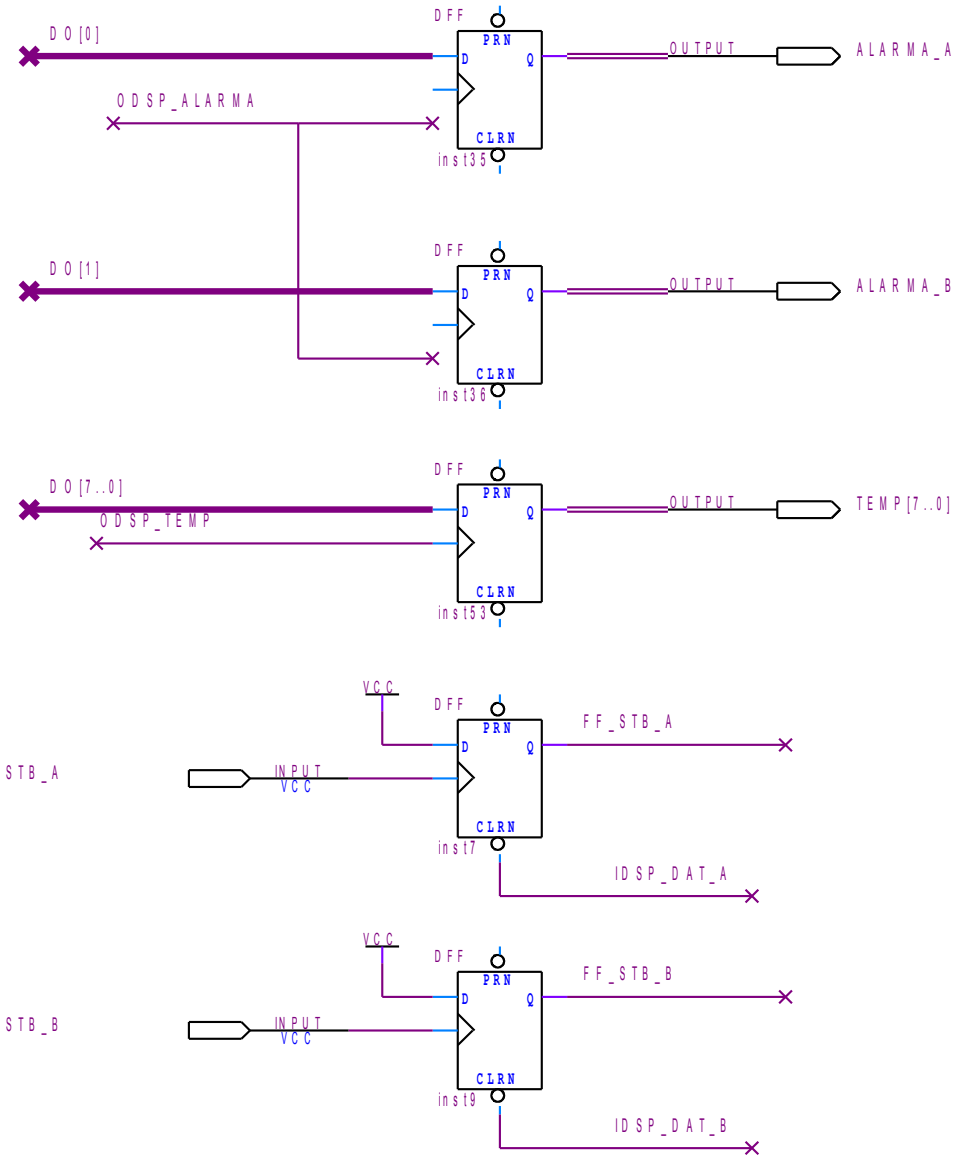
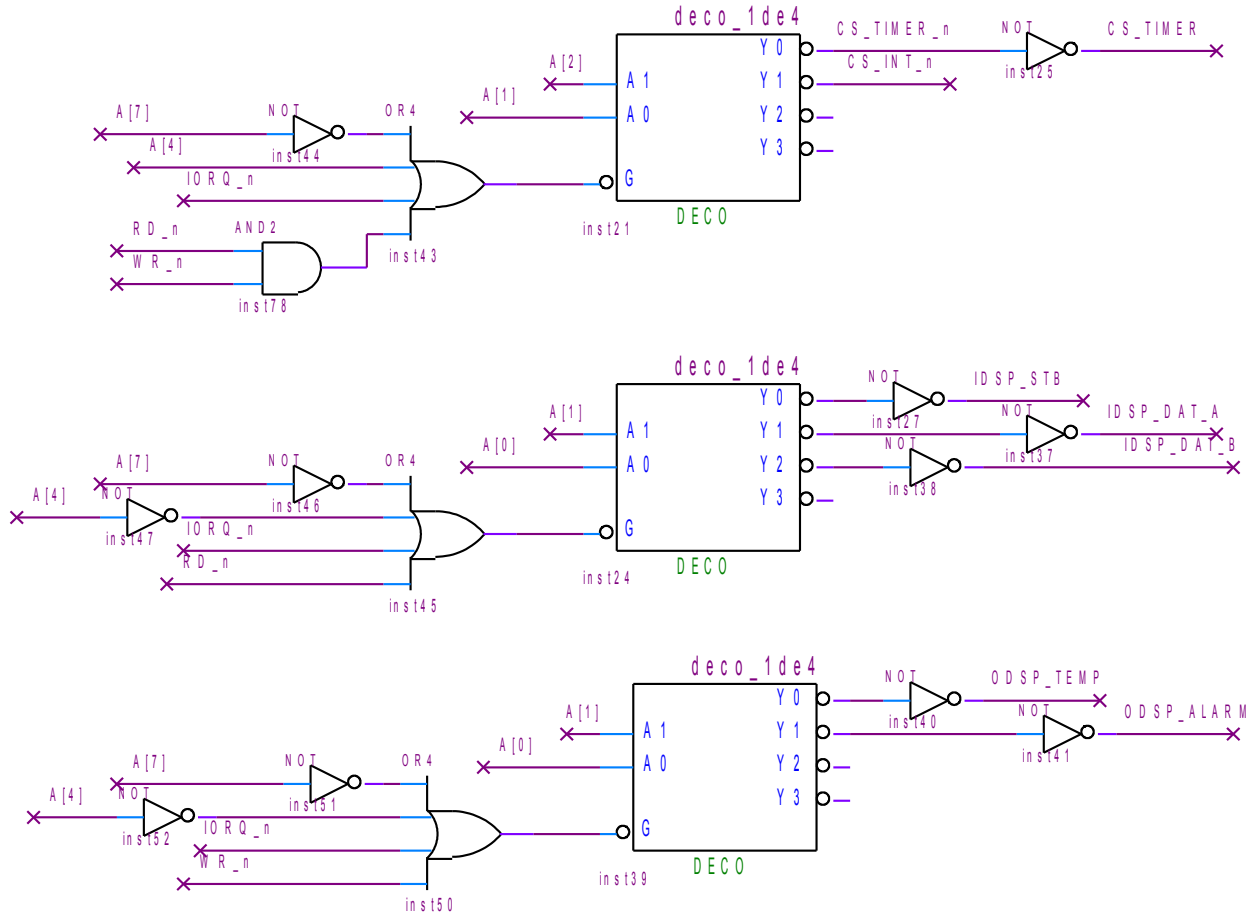
PROBLEMA 1 – Solución

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

- T_CTE : control de timer en dirección 80
- T_CTRL : control de timer en dirección 81
- CI_VI : vector de interrupciones en dirección 82
- P_STB : entrada en dirección 90, bit0=STB_A, bit1=STB_B
- P_DAT_A : entrada en direccion 91. Borra FF de STB_A
- P_DAT_B : entrada en direccion 92. Borra FF de STB_B
- P_TEMP : salida en dirección 90 - P_ALARM : salida en dirección 91, bit0=ALARMA_A, bit1=ALARMA_B
- 32K ROM y 32K RAM




DFF
PRN
D
Q
CLRN
inst35
DO[0]
ODSP_ALARMA
OUTPUT
ALARMA_A
DFF
PRN
D
Q
CLRN
inst36
DO[1]
OUTPUT
ALARMA_B
DFF
PRN
D
Q
CLRN
inst53
DO[7..0]
ODSP_TEMP
OUTPUT
TEMP[7..0]
VCC
DFF
PRN
D
Q
CLRN
inst7
STB_A
INPUT
VCC
FF_STB_A
IDSP_DAT_A
VCC
DFF
PRN
D
Q
CLRN
inst9
STB_B
INPUT
VCC
FF_STB_B
IDSP_DAT_B



Parte b)

```
; valor de flag de timeout
TIME_OUT_FLAG EQU 0xFFh

org ALGUN_LUGAR_ROM
RUTINT_TIMER:
  ei
  push af
  ld a, TIME_OUT_FLAG
  ld (TIME_OUT), a
  pop af
  reti
```

Parte c)

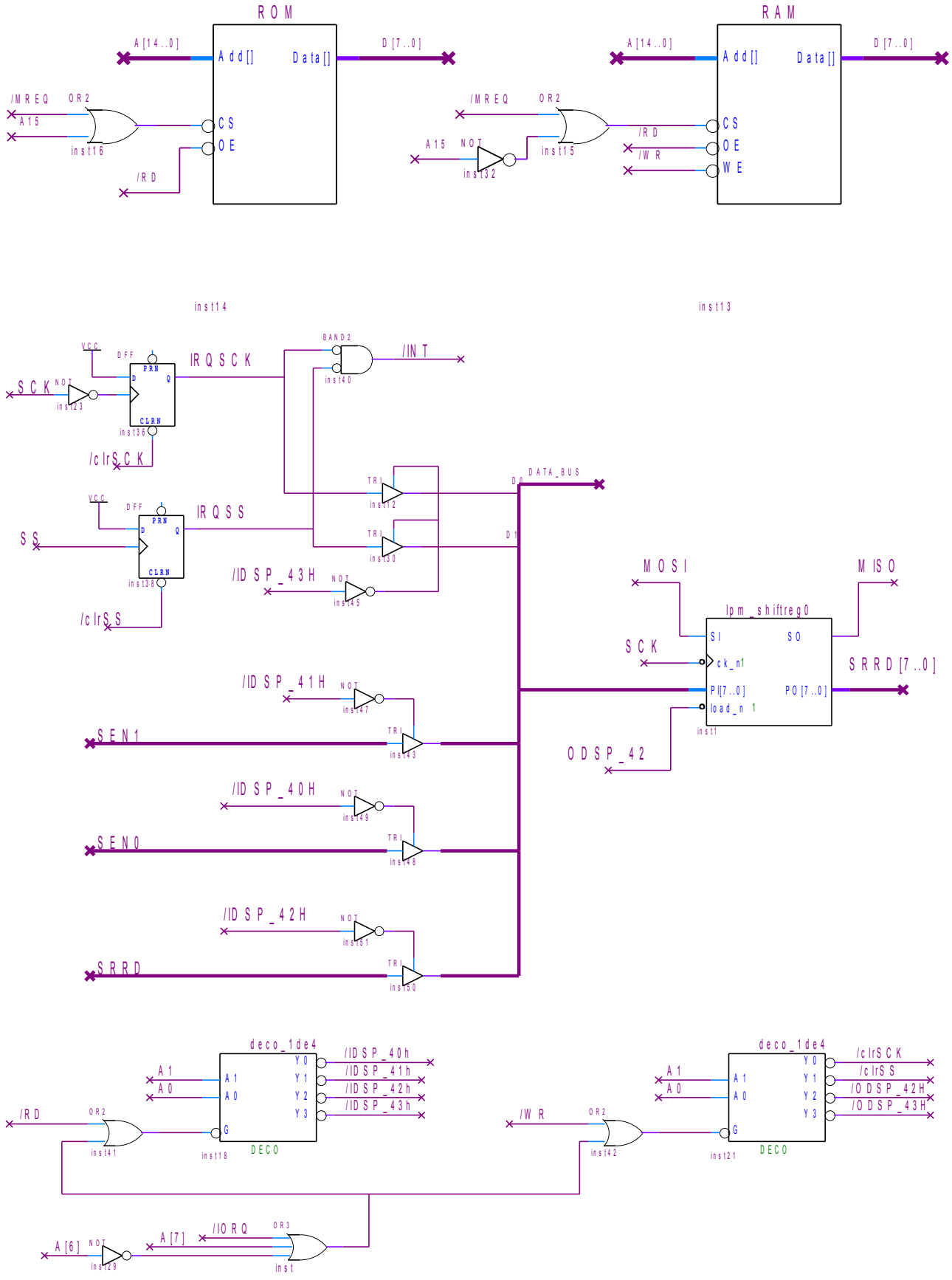
```
 - espero que llegue un dato (P_STB[1:0] != 00)
 - si llegan 2 a la vez (P_STB[1:0] == 11)
   - leo medida de A y borro flag P_STB[0]
   - leo medida de B y borro flag P_STB[1]
   - promedio
   - actualizo temp
   - borro alarmas
 - sino
   - borro flag de TIME_OUT
   - arranco timer
   - espero a que llegue la otra medida o TIME_OUT
   - case P_STB[1:0]
     - 01
       - leo medida de A y borro flag P_STB[0]
       - actualizo temp
       - prendo Alarma B
     - 10
       - leo medida de B y borro flag P_STB[1]
       - actualizo temp
       - prendo Alarma A
     - 11
       - leo medida de A y borro flag P_STB[0]
       - leo medida de B y borro flag P_STB[1]
       - promedio
       - actualizo temp
       - borro alarmas
   - vuelvo a esperar medidas
```

```
PPAL:
  in A, (P_STB)
  and A, 0x03h
  jp Z, PPAL

  cp A, 0x03h
 ;;;; llegaron al mismo tiempo
  jp Z, DOS_MEDIDAS

  ld A, TIME_OUT_FLAG
  cpl
  ld (TIME_OUT), A
  ; arranco cuenta de timer
  call START_TIMER

LOOP_ESPERA_MEDIDA:
  in A, (P_STB)
  and A, 0x03h
  cp 0x03h
  ; llego la segunda medida
  jp Z, DOS_MEDIDAS
  ld A, (TIME_OUT)
  cp TIME_OUT_FLAG
  jp NZ, LOOP_ESPERA_MEDIDA

NO_LLEGO_MEDIDA:
  in A, (P_STB)
```

```
  bit 0, A
  jp NZ, SOLO_MEDIDA_A
SOLO_MEDIDA_B:
  in A, (P_DAT_B) ; borra flag de STB_B
  out (TEMP), A
  ld A, 0x02
  out (ALARMA), A
  jp PPAL

SOLO_MEDIDA_A:
  in A, (P_DAT_A) ; borra flag de STB_A
  out (TEMP), A
  ld A, 0x01
  out (ALARMA), A
  jp PPAL

DOS_MEDIDAS:
  in A, (P_DAT_A) ; borra flag de STB_A
  ld B, A
  in A, (P_DAT_B) ; borra flag de STB_B
  add A, B
  srl A           ; promedio
  out (TEMP), A
  ld A, 0x00
  out (ALARMA), A
  jp PPAL

;;;;;;;; init
DELTA_T equ 250

org 0x8000
  TIME_OUT db

org 0x0000
  ld SP, 0000
  ld A, TABLA_INT / 256
  ld I, A
  ld A, 0
  out (TEMP), A
  in A, (P_DAT_A) ; borra flag de STB_A
  in A, (P_DAT_B) ; borra flag de STB_B
  ld A, 0x04  ; 3er VI
  out (CI_VI), A
  call ini_otros
  im2
  ei
  jp PPAL

START_TIMER:
  push AF
  ld A, DELTA_T
  out (T_CTE), A
  ; prescaler = 2^8 = 256
  ld A, 1x10 1000
  pop AF
  ret

org 0x0800
TABLA_INT:
  DW rutint_1
  DW rutint_2
  DW RUTINT_TIMER
```

PROBLEMA 2 – Solución

a) Hardware



b) Atención a interrupción

```
   org 0x38     ; modo 1

rutint:
   push af      ; preservar estado
   push bc
   in a, (flag) ; leer flags
   bit 1, a
   jr z, else
   ;--  si FlagSS entonces
   out (clrSS), a  ; borro FFSS
   call rut_SS
   jr fin_si
   ;-- else es FlagSCK
else:
   out (clrSCK), a ; borro FFfin
   call rut_fin
finsi:
   pop bc
   pop af
   ei
   ret

rut_SS:
; inc anterior mod 2
; leer sensor(anterior)
; escribir ShiftReg
; arrancar contador

; inc anterior mod 2
    ld a, (anterior)
    inc a
    and 00000001B
    ld (anterior), a
; leer sensor(anterior)
    add a, SENSOR0
    ld c, a
    in a, (c)
; escribir ShiftReg
    out (SRWR), a
; arrancar contador
    ld a, 8
    ld (cont_flancos), a
    ret

rut_fin:
; dec contador
; si (octavo flanco) entonces
;    leer ShiftReg
;    guardar en memoria
;    frenar contador
; fin_si

    ld a, (cont_flancos)
    dec a
    ld (cont_flancos), a
    jr nz fin_si_octavo
        ;-- si (octavo flanco) entonces
    in a, (SRRD)
    ld (CONFIG), a

fin_si_octavo:
    ret
```

c) Inicialización y directivas

```
org 0
    ld sp, 00        ; stack y modo int
    im 1
    out (clrSS), a   ; borro FFs
petición
    out (clrfin), a
    ld a, 1
    ld (anterior), a ; sensor anterior =
1

    call init_otros   ;ini resto sistema
    ei
    jp ppal         ; hab. int y jp ppal

    ;-- esto debe caber antes de 0x38

org 0x38
    ;-- aquí van las subrut.de parte b)
    ...

    ;-- a continuación el programa
principal e init_otros
ppal:
    ...

org 0x8000          ; comienzo ram
CONFIG:       DB
cont_flancos: DB
anterior:     DB

;; definicion de ctes
  clrSCK    equ 0x40
  clrSS     equ 0x41
  SRWR      equ 0x42

  SENSOR0   equ 0x40
  SENSOR1   equ SENSOR0+1
  SRRD      equ 0x42
  flag      equ 0x43
```