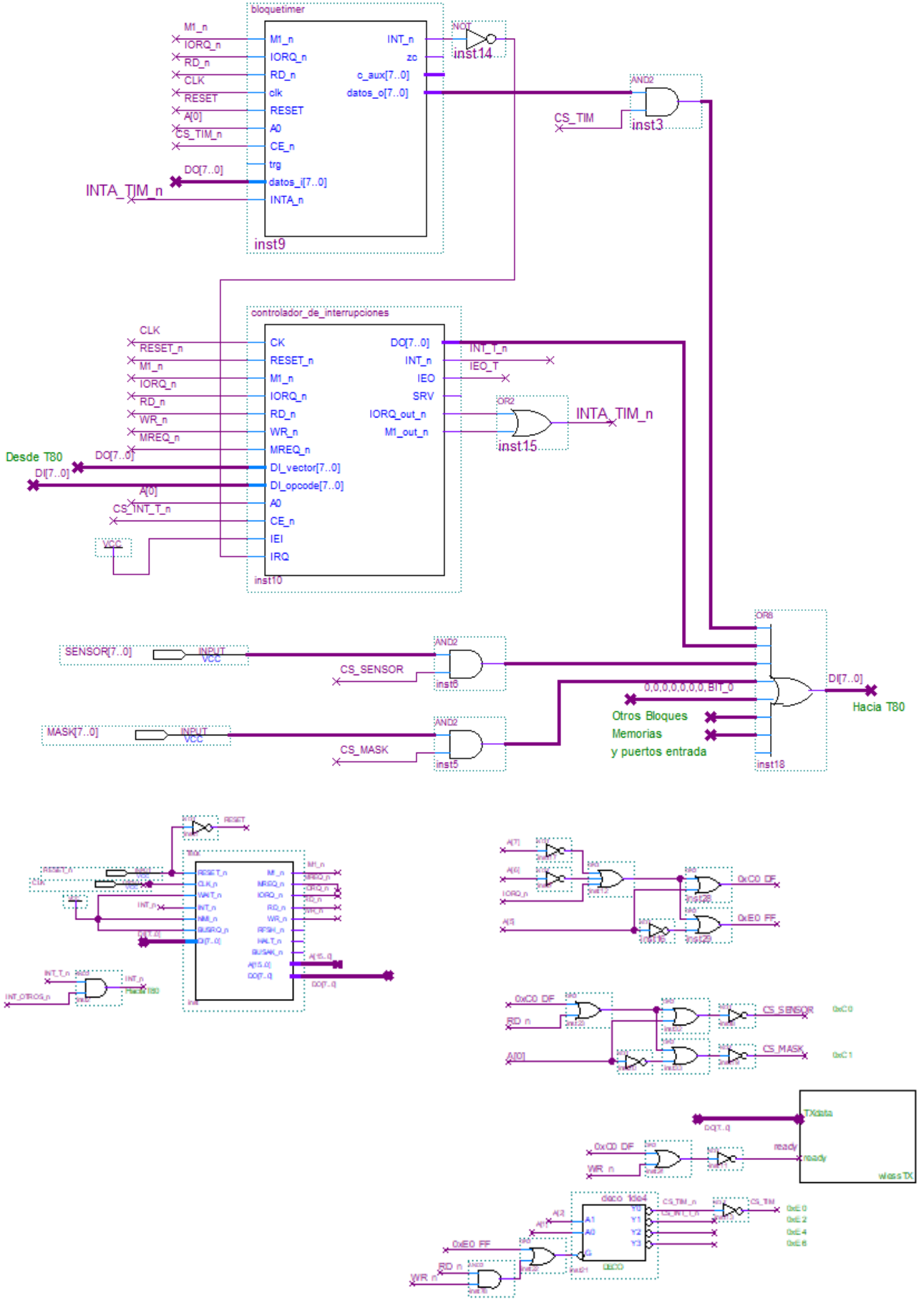
PROBLEMA 1

a) hardware



b) Inicialización

```
PRE_MAX     EQU 12
PRE_MIN     EQU 5
CTE_TIM     EQU 124
;; palabra de control
;;   completar con or con prescaler
;;   ei, x, sw reset, arranque auto
CONTROL_TIM EQU 1x10 0000B

;; dejo libres vectores 0 y 2
VECTOR_CINT EQU 0x04

BASE_TIM    EQU 0xE0
BASE_CINT   EQU 0xE2
SENSOR      EQU 0xC0
pMASK       EQU 0xC1
TXdata      EQU 0xC0

org 0x200
tabla_int:
    dw isr_0
    dw isr_2
    dw isr_timer

org 0x8000
pre_actual:    db
dato:          db
recien_cambio: db
mask:          db

org 0
init:
; sp, modo y tabla en rom
   ld sp, 0
   im 2
   ld hl, tabla_int
   ld a, h
   ld i, a
; controlador de interrup
  out (BASE_CINT+1), a
  ld a, VECTOR_CINT
  out (BASE_CINT), a

; cte del timer
   ld a, CTE_TIM
   out (BASE_TIM), a
;; mascara
   in a, (pMASK)
   ld (mask), a
;; dato
   in a, (SENSOR)
   ld (dato), a
    ;;  transmito dato
   out (TXdata), a
    ;;  pre_actual = PRE_MIN
   ld a, PRE_MIN
   ld (pre_actual), a
    ;;  reprogramo timer
   or a, CONTROL_TIM
   out (BASE_TIM+1), a
    ;;  recien_cambio = true
   ld a, 0xff
   ld (recien_cambio), a
;; init otras tareas y disp.
   call init_otros
;; habilito int y salto a ppal
   ei
   jr prog_ppal
```

c) Prog. ppal

```
; forever{
;   atiendo otros
;   si (recien_cambio == false) {
;     si cambio algun bit habilitado {
;       reprogramo timer con PRE_MIN
;       pre_actual = PRE_MIN
;       guardo dato
;       transmito dato
;       recien_cambio = true
;     }
;   }
; }

prog_ppal:
   call atiendo_otros
   ld a, (recien_cambio)
   or a
   jr nz fin_forever
    ;; si (recien_cambio == false) {
   in a, (SENSOR)
   ld b, a
   ld a, (dato)
   xor a, b
   and a, (mask)
   jr z, fin_si_cambio
    ;; si cambio algun bit {
    ;;  reprogramo timer con PRE_MIN
   ld a, PRE_MIN
   or a, CONTROL_TIM
   out (BASE_TIM+1), a
    ;; // timer recien arranca
    ;; // no habrá interrupciones
    ;; // mientras modifico variables
    ;;  pre_actual = PRE_MIN
   ld a, PRE_MIN
   ld (pre_actual), a
    ;;  recien_cambio = true
   ld a, 0xff
   ld (recien_cambio), a
    ;;  guardo dato
   ld a, b
   ld (dato), a
    ;;  transmito dato
   out (TXdata), a
fin_si_cambio:
fin_forever:
   jr prog_ppal
```

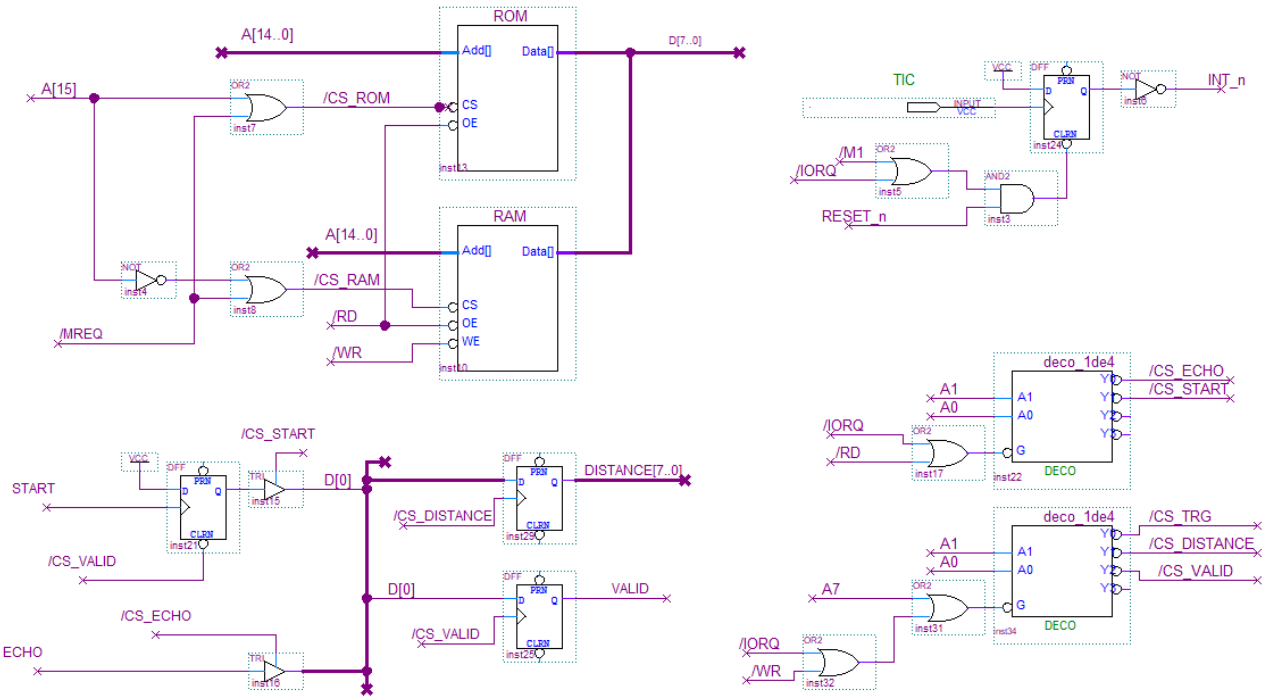
d) Rutina atención interrup

```
: isr_timer()
; preservar estado
; si pre_actual != PRE_MAX {
;    pre_actual++
;    reprogramo timer
;   }
; transmitir dato
; recien_cambio = false
; restaurar estado


isr_timer:
   ei
   push af
   ld a, (pre_actual)
   cp PRE_MAX
   jr z, fin_si
    ;; si pre_actual != PRE_MAX {
    ;;    pre_actual++
    ;;    reprogramo timer
   inc a
   ld (pre_actual), a
   or a, CONTROL_TIM
   out (BASE_TIM+1), a

finsi:
    ;; transmitir dato
   ld a, (dato)
   out (TXdata), a
    ;; recien_cambio = false
   ld a, 0
   ld (recien_cambio), a
   pop af
   reti
```

PROBLEMA 2



b) Atención interrupción

```
E_REPOSO     EQU 0x00
E_MIDIENDO   EQU 0x01
E_VALIDO     EQU 0x02
E_TIMEOUT    EQU 0x03
I_ECHO       EQU 0x00
TIMEOUT      EQU 250

org 0x8000  ;; BASE_RAM
ESTADO_INT     DB
CONTADOR_TICS DB

org 0x0038
rutint:
  push AF
  ld A, (ESTADO_INT)
  cp E_MIDIENDO
  jp Z, est_midiendo

est_distinto_midiendo:
  jp fin

est_midiendo:
  in A, (I_ECHO)
  cp 0x00
  jp no_echo

llego_echo:
  ld A, E_VALIDO
  ld (ESTADO_INT), A
  jp fin

no_echo:
  ld A, (CONTADOR_TIC)
  inc A
  cp TIMEOUT
  jp Z, si_timeout

no_timeout:
  ld (CONTADOR_TIC), A
  jp fin

si_timeout:
  ld A, TIMEOUT_FLAG
  ld (CONTADOR_TIC), A
  ld A, E_TIMEOUT
  ld (ESTADO_INT), A

fin:
  pop AF
  ei
  ret
```

c) Prog. principal

```
 O_VALID    EQU 0x02
 I_START    EQU 0x01
 O_DISTANCE EQU 0x01
 O_TRG      EQU 0x00

 org 0x0000
 ld SP, 0x0000
 im 1

 ld A, 0x0
 out (O_VALID) ; borra valid y FF START

 ld A, E_REPOSO
  ;; estado rutina diferente a E_MIDIENDO
 ld (ESTADO_INT), A

 ei

loop:
 in A,(I_START)
 bit 0,A
 jp Z, loop

comienzo_medir:
 ld A, E_MIDIENDO
  ;; indico a rutint que debe medir
 ld (ESTADO_INT), A
 out (O_TRG), A  ; doy pulso trigger
 ld A, 0x00
 out (O_VALID), A  ; desactivo VALID
```

```
 ld (CONTADOR_TIC), A ; resteo contador

espero_rutint:
 ld A,(ESTADO_INT)
 cp E_MIDIENDO
 jp Z, espero_rutint
 cp E_VALIDO
 jp Z, calculo_distancia

ocurrio_timeout:
 ld A,(CONTADOR_INT)
 out (O_DISTANCE), A
 ld A, 0x01
 out (O_VALID), A ; activo VALID
 ld A, E_REPOSO
 ld (ESTADO_INT), A
 jp loop

calculo_distancia:
 ld A,(CONTADOR_INT) ; 1 TIC = 4cm
 SLA A
 SLA A ; A en cm
 SRA A ; d = 2d / 2
 out (O_DISTANCE), A
 ld A, 0x01
 out (O_VALID), A ; activo VALID
 ld A, E_REPOSO
 ld (ESTADO_INT), A
 jp loop
```