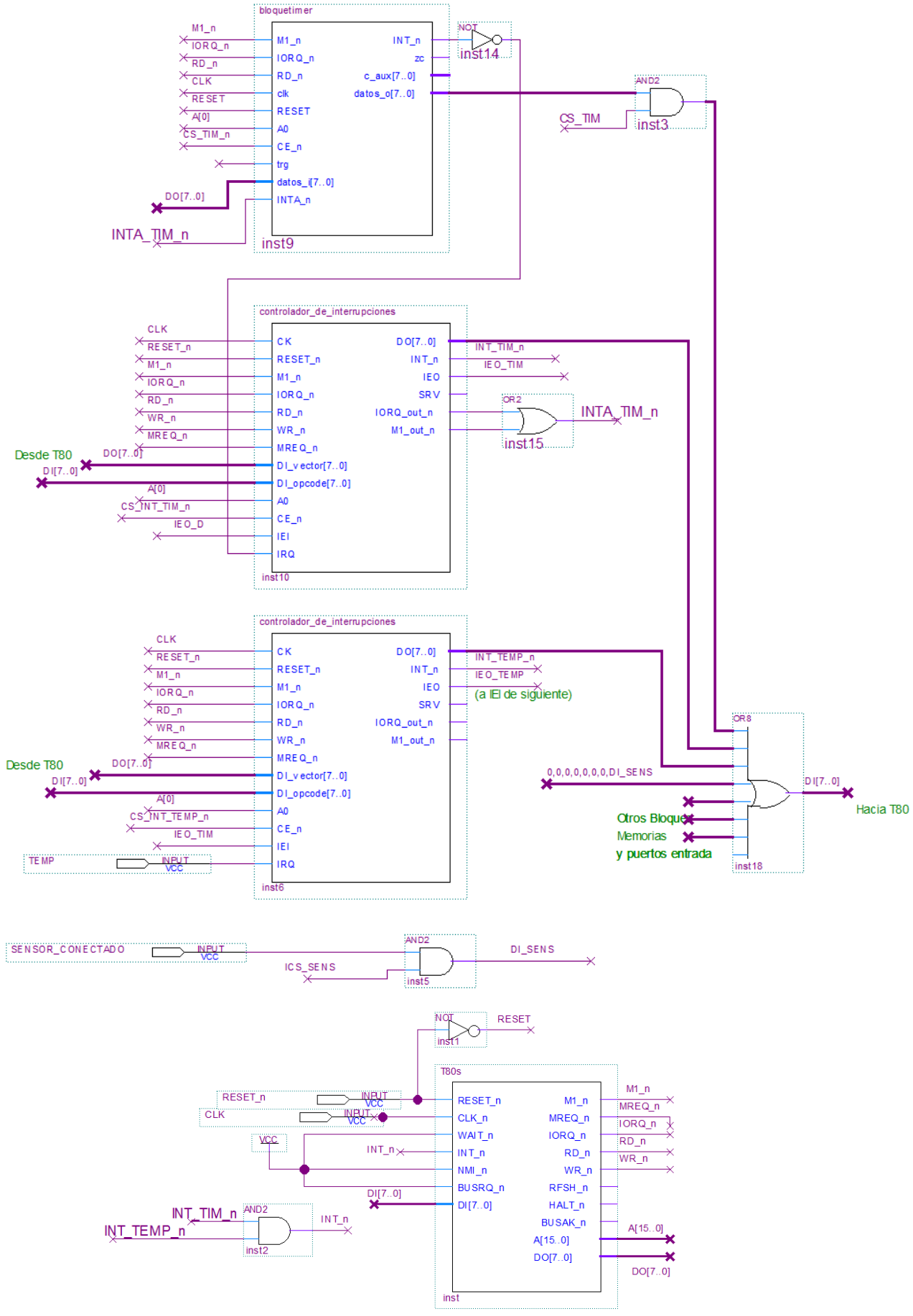
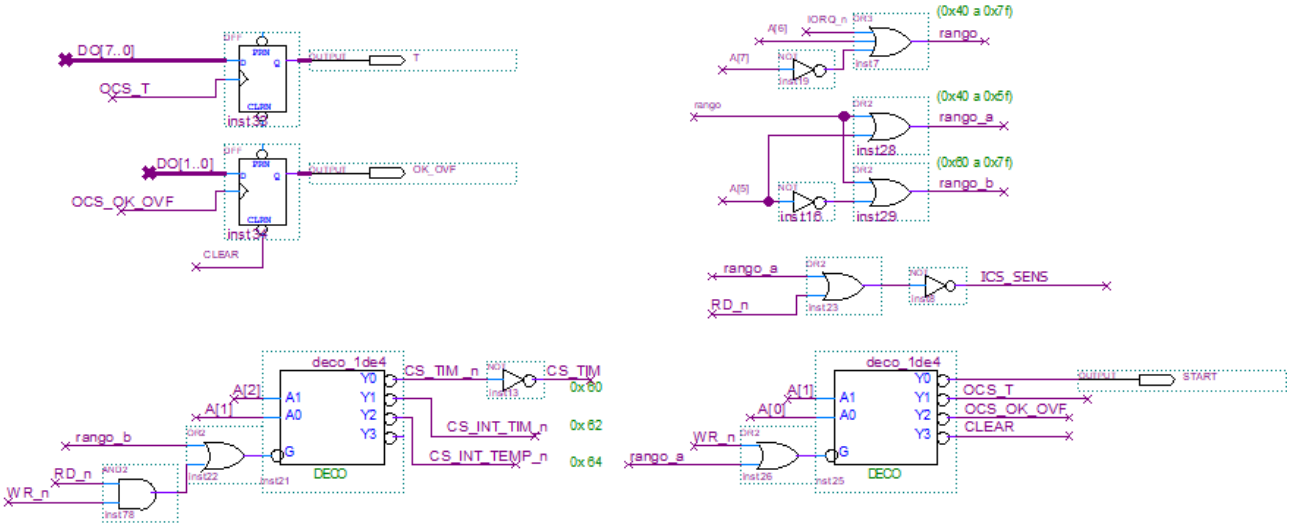
SOLUCIÓN PROBLEMA 1

a) Hardware




DO[7..0]
OCS_T
T
inst17
DO[1..0]
OCS_OK_OVF
OK_OVF
inst32
CLEAR
IORQ_n
A[6]
A[7]
rango
(0x40 a 0x7f)
inst7
inst15
rango
rango_a
(0x40 a 0x5f)
inst28
A[5]
(0x60 a 0x7f)
rango_b
inst116
inst29
rango_a
RD_n
inst23
ICS_SENS
deco_1de4
A[2]
A[1]
A1
A0
G
Y0
Y1
Y2
Y3
DECO
CS_TIM_n
CS_TIM
0x60
CS_INT_TIM_n
0x62
CS_INT_TEMP_n
0x64
rango_b
RD_n
WR_n
inst22
inst21
inst78
deco_1de4
A[1]
A[0]
WR_n
rango_a
OCS_T
OCS_OK_OVF
CLEAR
START
inst26
inst25
DECO

b)

```
org 0x3000
iniciar_medida:
   ; preservar registros
     ; borrar OK y OVF
     ; borrar flag de TIMEOUT
     ; si sensor_conectado = 1
     ;    dar pulso en START
     ; restaurar registros
     ; retornar
  push AF
    out (CLEAR), A
    ld A, 0
    ld (TIMEOUT),A
    in A,(SENSOR_CONECTADO)
    bit 0,A
    jp Z, fin_iniciar_medida
      out (START), A
fin_iniciar_medida:
  pop AF
    ret
```

c)

```
org 0x4000
rutint_temp:
   ; habilitar interrupciones
     ; preservar registros
     ; si NO hubo timeout
     ;   leer timer
     ;   reprogramar timer (EI,
sw_reset, arranque con flanco bajada
trg, pre=8)
     ;   temperatura = 60 - valor
leido
     ;   escribir temperatura en
puerto T
     ;   OK=1
     ; restaurar registros
     ; retono de interrupción
  ei
  push AF
  push BC

  ld A,(TIMEOUT)
  cp 0
  jp NZ, fin_rutint_temp
  in A,(TIMER_A1)
  ld B,A
  ld A, CW_TIMER
   ;; reprogramo timer para prox.
medida
   ;;   y evito que gener int por
timeout
  out (TIMER_A1), A
  ld A, MAX_TEMP-1
  sub B ; MAX_TEMP-1 - valor timer
  out (T), A
  ld A, OK
  out (OK_OVF), A
  pop BC
  pop AF
  reti


rutint_timer:
   ; habilitar interrupciones
     ; preservar registros
     ; setear flag de timeout
     ; reprogramar timer (EI,
sw_reset, arranque con flanco bajada
trg, pre=8)
     ; OVF=1
     ; restaurar registros
     ; retono de interrupción
  ei
  push AF

  ld A, 0xFF
  ld (TMEOUT), A
  ld A, CW_TIMER
   ;; reprogramo timer para prox.
medida
   ;;   y evito nuevas int por
timeout
  out (TIMER_A1), A
  ld A, OVF
  out (OK_OVF), A

  pop AF
  reti
```

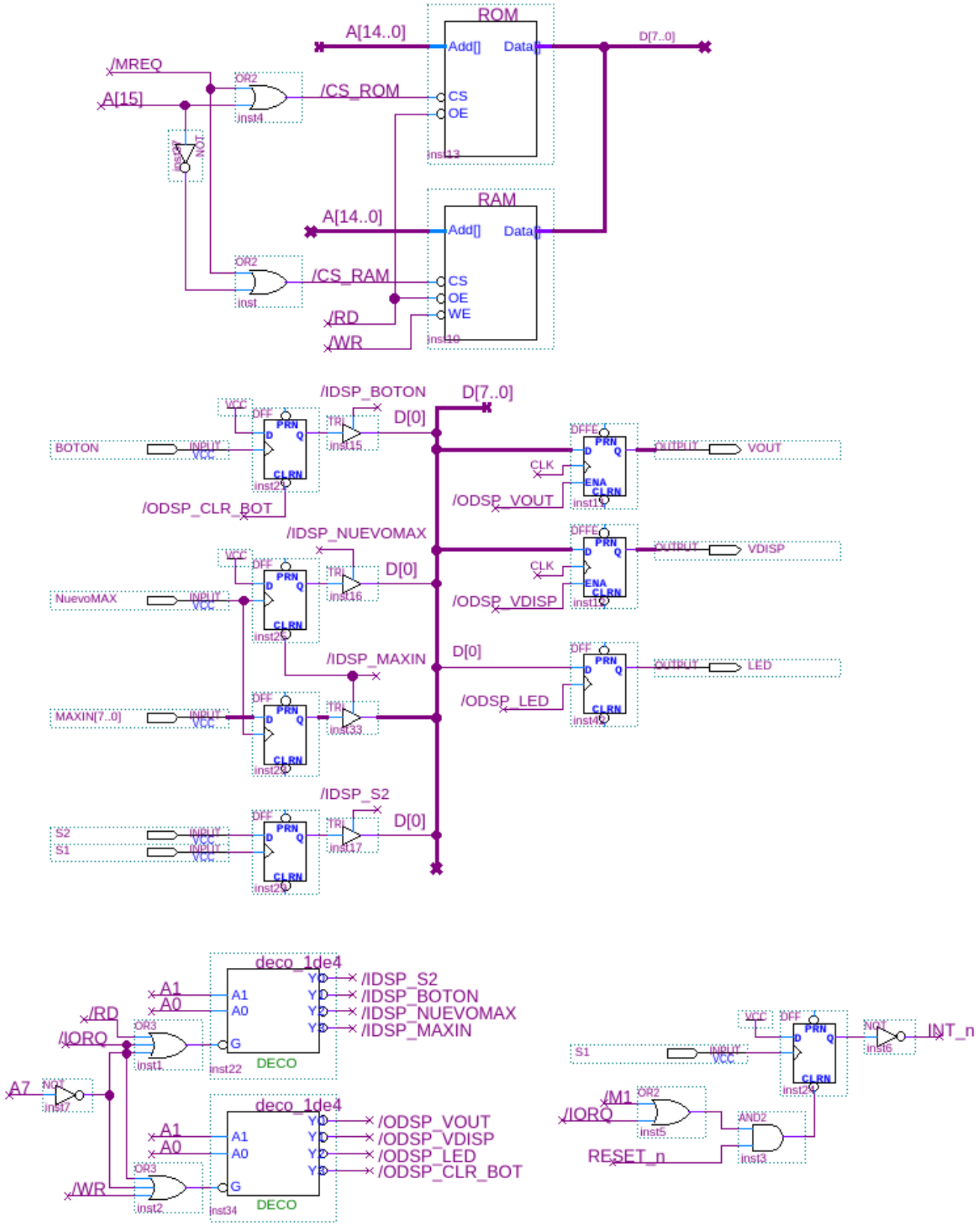
d)

```
org 0x0000
; SP= 0x0000
; I=0x90
; cargar direcciones rutinas en
tabla de int
; programar VI=6 en CI_1
; borrar int pendientes CI_1
; programar VI=8 en CI_2
; borrar int pendientes CI_2
; cargar constante MAX_TEMP-1 en
timer
; programar timer (EI, sw_reset,
arranque con flanco bajada trg,
pre=8)
; borrar puertos OK y OVF y escribir
0 en puerto T
; llamar a init_otros
; habilitar interrupciones
; saltar a programa_ppal

  ld SP, 0x0000
  ld A, tabla_int /256
  ld I,A
  ld HL, (tabla_int + VI_TIMER)
  ld (HL), rutint_timer
  ld HL, (tabla_int + VI_TEMP)
```

```
  ld (HL), rutint_temp
  ld A, VI_TIMER
  out (CI_1_A0), A
  out (CI_1_A1), A ; borro
peticiones
  ld A, VI_TEMP
  out (CI_2_A0), A
  out (CI_2_A1), A ; borro
peticiones
  ld A, MAX_TEMP-1
  out (TIMER_A0), A
  ld A, CW_TIMER
  out (TIMER_A1), A
  out (CLEAR), A
  call init_otros
  ei
  jp programa_ppal

;puertos
CLEAR            equ 0x43
SENSOR_CONECTADO equ 0x40
START            equ 0x40
OK_OVF           equ 0x42
T                equ 0x41
TIMER_A0 equ 0x60
TIMER_A1 equ 0x61
CI_1_A0  equ 0x62
CI_1_A1  equ 0x63
CI_2_A0  equ 0x64
CI_2_A1  equ 0x65

;constantes
MAX_TEMP equ 60
OK       equ 0x01
OVF      equ 0x02

CW_TIMER equ 1011 1000 ; EI, trigger
bajada, reset sw, arranca por
trigger, pre=8
VI_TIMER equ 6
VI_TEMP  equ 8

org 0x8000
TIMEOUT DB

org 0x9000
tabla_int:
   DW   ;
   DW   ;
   DW   ;
   DW   ; timer
   DW   ; temp
```

SOLUCIÓN PROBLEMA 2

a) Hardware



b) Inicialización, programa principal y reserva de memoria

```
; puertos E/S
S2          equ 0x80
BOTON       equ 0x81
NUEVOMAX    equ 0x82
MAXIN       equ 0x83
VOUT        equ 0x80
VDISP       equ 0x81
LED         equ 0x82
CLR_BOT     equ 0x83
;constantes
MIN_V       equ 0

.data ; en comienzo RAM 0x8000
vaux:       db
max_v:      db

.text ; inicio de ROM 0x0000
im 1
ld sp, 0
ld a, 0
ld (vaux), a
out (VOUT), a
out (LED), a
ei

main:
; forever{
;   atiendo otros
;   si Boton{
;       clr FF Boton
;       VOUT = vaux
;       LED = 0
;    si NuevoMax{
;       max_v = MAXIN
;       si vaux>max_v {
;          vaux = max_v
;       }
;    }
; }
    call atiendo_otros
    in a, (BOTON)
    bit 0, a
    jr z, miro_vmax
  actualizo_vout:
    out (CLR_BOT), a
    ld a, (vaux)
    out (VOUT), a
    ld a, 0
    out (LED), a
  miro_vmax:
    in a, (NUEVOMAX)
    bit 0, a
    jr z, main
actualizo_vmax:
    ld a, (vaux)
    ld b, a
    in a, (MAXIN)
    ld (max_v), a
    cp b         ;MAXIN-vaux
    jp P, main   ;no es mayor, sigo
    ld (vaux), a ;es mayor, saturo
     jr main
```

c) Rutina de atención a la interrupción ----------

```
; isr_modificar_volt:
; preservar registros
; flag = 1
; LED = 1
; si S2=0 entonces
;   vaux = min(MAX_V, vaux + 1)
; else
;   vaux = max(MIN_V, vaux - 1)
; end si
; restaurar

    org 0x0038
isr_modificar_volt:
    push af
    push bc
    ld a, 1
    out (LED), a
    in a, (S2)
    bit 0, a
    jr nz, antihorario
horario:
    ld a, (max_v)
    ld b, a
    ld a, (vaux)
    inc a
    cp b
    jp M, fin_si ;MAX>vaux
satura_max:
    ld a, b   ; saturo en MAX_V
    jr fin_si
antihorario:
    ld a, (vaux)
    dec a
    ld b, MIN_V
    cp b
    jp P, fin_si ;MIN<=vaux
satura_min:
    ld a, b   ; saturo en MIN_V
fin_si:
    ld (vaux), a
    out (VDISP), a
    pop bc
    pop af
    ret
```