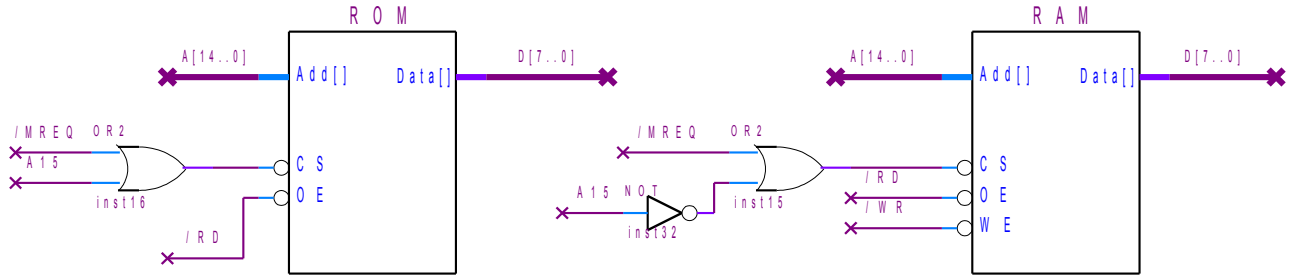
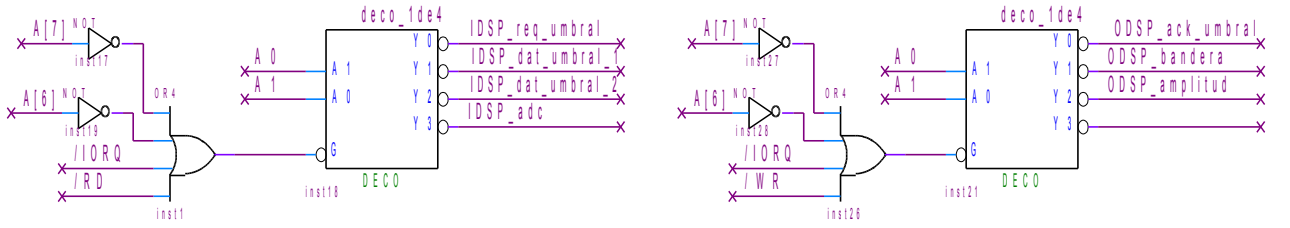
SOLUCIÓN PROBLEMA 1

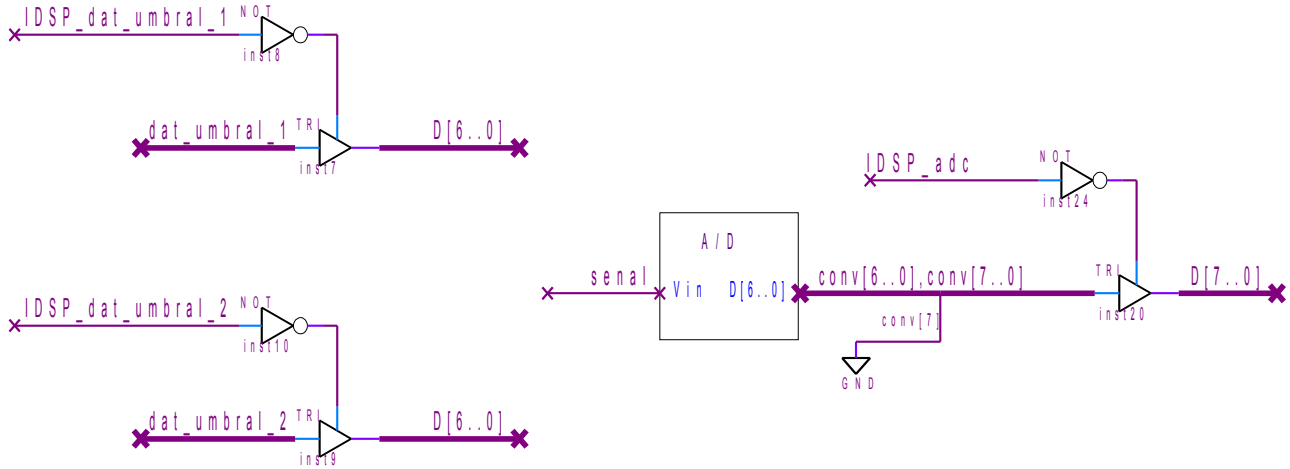
a) Hardware

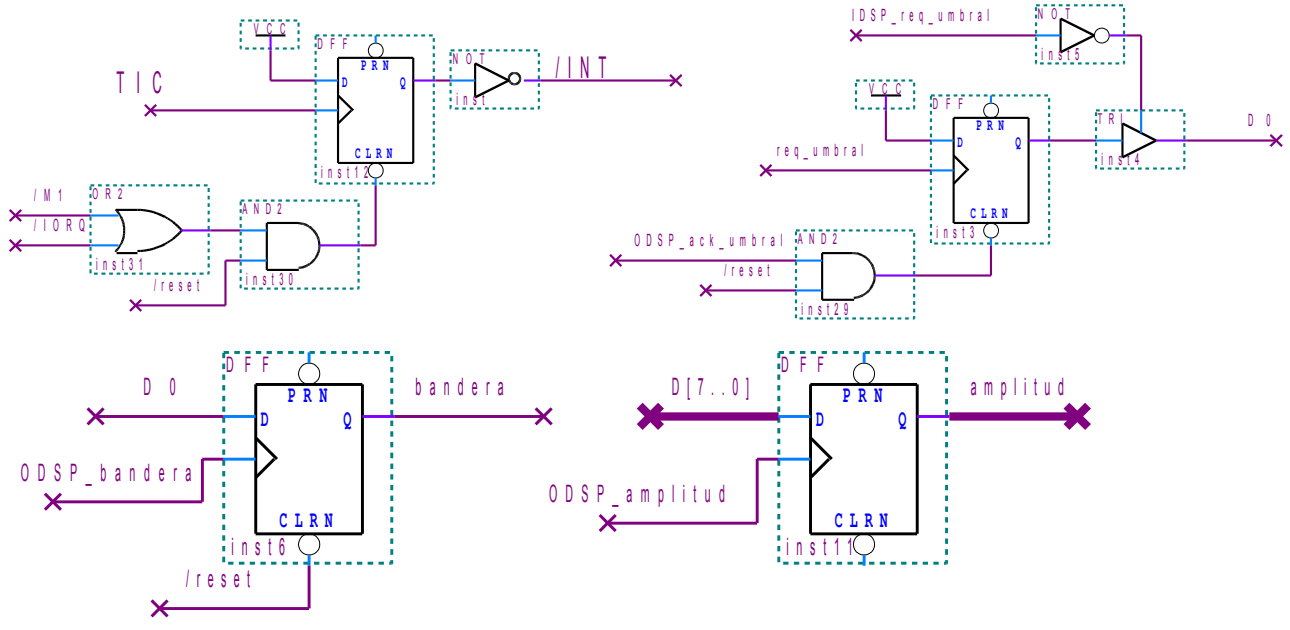
-- Memorias:



-- Puertos, decodificación e Interrupciones.







b) Inicialización y reserva mem.

```
adc      EQU 0xC3
datumb_2 EQU 0xC2
datumb_1 EQU 0xC1
req_umb  EQU 0xC0
ack_umb  EQU 0xC0
bandera  EQU 0xC1
amplitud EQU 0xC2
umb1_ini EQU 20
umb2_ini EQU 5

.Org 0x8000
umbral_1 DB
umbral_2 DB
Dat_ant  DB
Estado   DB
Máximo   DB

.Org 0x0000
       LD SP, 0x0000
       LD A, umb1_ini
       LD (umbral_1), A
       LD A, umb2_ini
       LD (umbral_2), A
       LD A, 0x00
       LD (Dat_ant), A
       LD (Estado), A
       LD (Máximo), A
       OUT (amplitud), A
       IM 1
       EI
       JP Loop
```

c) Loop programa principal

```
; for (true) {
; Leo req_umbral
; if (req_umbral == 1){
;      actualizo umbral_1
;      actualizo umbral_2
;      doy pulso ack_umbral
;      }
; }
.Org 0x100
; dejo lugar para la rutina de
; atención a la interrupción
Loop:  IN A, (req_umb)
       BIT 0, A
       JP Z, Loop
       IN A, (datumb_1)
       AND 0x7F
       LD (umbral_1), A
       IN A, (datumb_2)
       AND 0x7f_
       LD (umbral_2), A
       OUT (ack_umb), A
       JP Loop
```

d) Rutina atención interrupciones

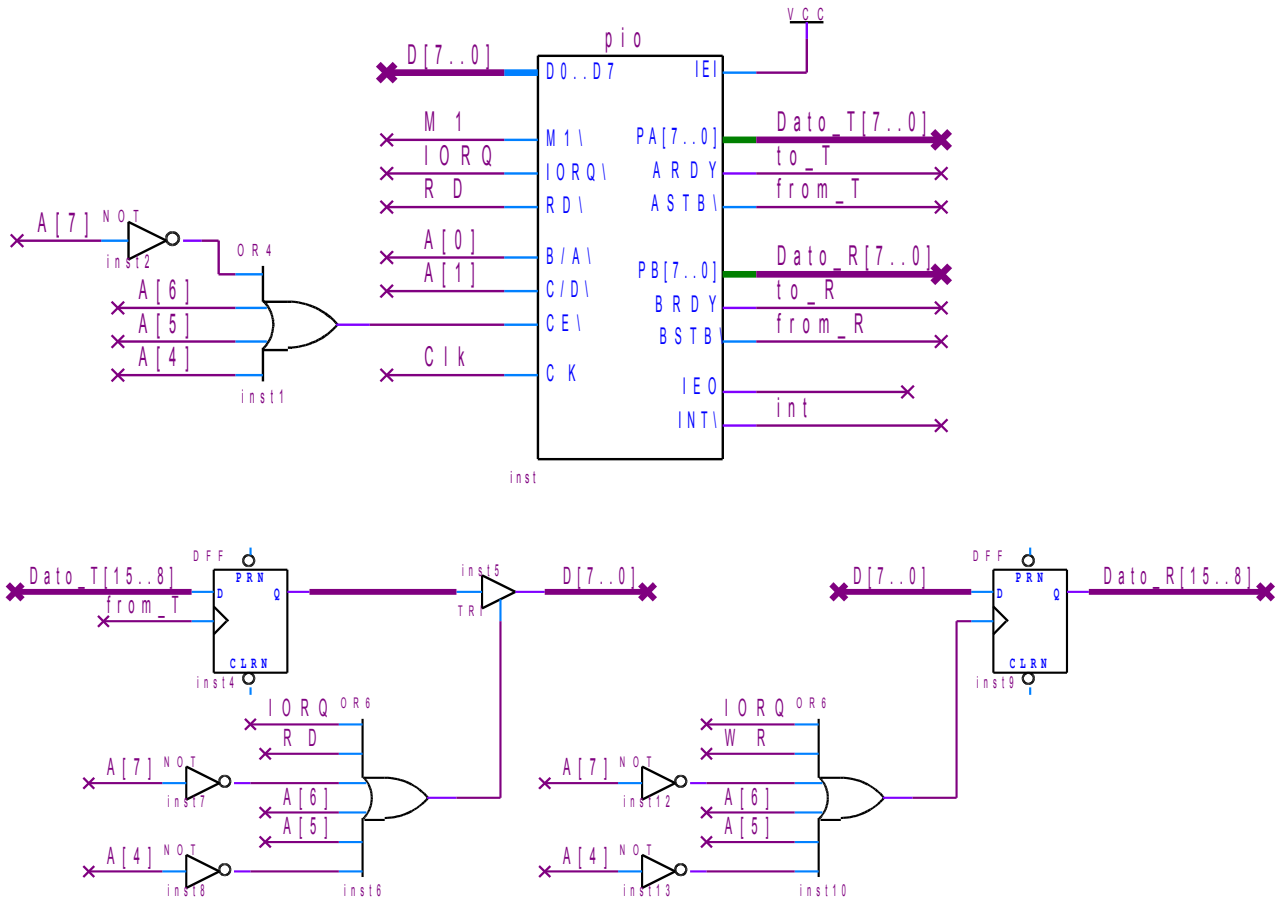
```
; preservo registros
; Dato_leido = puerto_adc
; if (Estado == reposo){
;   if (Dato_leido – Dat_ant – umbral1 > 0){
;     bandera = 0xFF
;     Estado = pico (0xFF)
;     Máximo = Dato_leído
;    }
; } else{
;      if ( (Dato_leido < umbral2) and
;           (Dat_ant < umbral2)){
;      bandera = 0x00
;      Estado = reposo  (0x00)
;      amplitud = Máximo
;   }else{
;     if (Dato_leído > Máximo){
;       Máximo = Dato_leído
;     }
;   }
; }
; Dat_ ant = Dato_leído
; restauro registros y retorno

; Nota: Tener en cuenta que tanto los datos
; leídos del ADC como los umbrales siempre
; son positivos y menores que 128.

.Org 0x38
       PUSH AF
       PUSH BC
       PUSH HL
       LD A, (Dat_ant)
       LD B, A
       IN A, (adc)
       LD C, A; Guardo en C dato leído de ADC
       LD A, (Estado)
       AND 0xFF
       JP NZ, Pico
       LD HL, umbral_1
       LD A, C
       SUB B        ; dato_leido - dat_ant
       CP (HL)      ; menos umbral1
       JP M, Fin
          ; Si positivo → comienzo de pico
          ; si no reposo
       LD A, 0xFF
       OUT (bandera), A
       LD (Estado),A
       LD A, C
       LD (Máximo), A
       JP Fin
Pico:  LD A, C
       LD HL, umbral_2
       CP (HL)
       JP P, Máx
          ; Si es positivo no se cumple
          ; cond. de salida de pico
       LD A, B      ; dat_ant
       CP (HL)
       JP P, Máx
          ; Se cumple cond. de salida de pico
       LD A, 0x00
       OUT (bandera), A
       LD (Estado), A
       LD A, (Máximo)
       OUT (amplitud), A
       JP Fin
Máx:   LD A, (Máximo); comparo maximo
       CP C          ; con C (dato_leido)
       JP P, Fin
       LD A, C       ; maximo = Dato_leido
       LD (Máximo), A
Fin:   LD A, C
       LD (Dat_ant), A
       POP HL
       POP BC
       POP AF
       EI
       RET
```

SOLUCIÓN PROBLEMA 2

a) Hardware.

El protocolo descrito para los pares de señales **from_T**, **to_T** y **from_R**, **to_R** coincide con el comportamiento de las señales **RDY** y **STB** del PIO. Conecto dispositivo T a puerto A porque debe tener mayor prioridad.

Memorizo el byte alto de **Dato_T** con flanco de subida de **from_T** en un registro y agrego puertos de 8 bits para byte alto de **Dato_T** y **Dato_R**



b) Inicialización y reserva memoria

```
PIOconf_A  EQU 0x82
PIOconf_B  EQU 0x83
DatoT_low  EQU 0x80
DatoT_high EQU 0x90
DatoR_low  EQU 0x81
DatoR_high EQU 0x90

.Org 0x1000; Tabla en ROM
tabla_int:
       DW rutint_T
       DW rutint_R

.Org 0x8000; En RAM
R_Busy DB 0x00

.Org 0x0000
       LD SP, 0x0000
;Config. PIO, canal A, dispositivo T
       LD A, 01001111; Modo 1, entrada
       OUT (PIOconf_A), A
       LD A, 00000000; Vector int.
       OUT (PIOconf_A), A
       LD A, 10000011; Habilit. int.
       OUT (PIOconf_A), A
;Config. canal B, dispositivo R
       LD A, 00001111; Modo 0, salida
       OUT (PIOconf_B), A
       LD A, 00000010; Vector int.
       OUT (PIOconf_B), A
       LD A, 10000011; Habilit. Int.
       OUT (PIOconf_B), A

       IN A, (DatoT_low)  ; para subir RDY

       LD A, 0x00
       LD (R_Busy), A
       LD A, 0x10
       LD I, A
       IM 2
       EI
       CALL iniotros
Loop:  CALL otros
       JP Loop
```

c) Rutina interrupción Dispositivo R

```
;      preservo registros
;      si GetFifo devuelve dato
;            escribo dato en puerto
;      else
;            R_Busy = 00h
;      fin_si
;      restauro registros
;      habilito int, retorno

       .Org 0x0200
rutint_R:
       PUSH AF
       PUSH BC
       CALL GetFifo ; entre que verifico cola
          ; vacía y escribo R_Busy no deben
          ; entrar interrupciones del otro
          ; dispositivo
       JP Z, enviar
       LD A, 0x00
       LD (R_Busy), A
       EI
       JP finR
enviar: EI
       LD A, B
       OUT (DatoR_high), A
       LD C, A
       OUT (DatoR_low), A
finR:  POP BC
       POP AF
       RETI
```

d) Rutina interrupción Dispositivo T

```
;      preservo registros
;      leer dato de puerto
;      si (R_Busy = 0) entonces
;            escribo dato en puerto
;            R_Busy = FFh
;      else
;            PutFifo(dato)
;      fin_si
;      restauro registros
;      habilito int, retorno
rutint_T:
       PUSH AF
       PUSH BC
       IN A, (DatoT_high)
       LD B, A
       IN A, (DatoT_low)
       LD C, A
       LD A, (R_Busy)
       BIT 0, A
       JP NZ, busy
       LD A, 0xFF
       LD (R_Busy), A
       LD A, B
       OUT (DatoR_high), A
       LD A, C
       OUT (DatoR_low), A
       JP finR
busy: CALL PutFifo
finR: POP BC
       POP AF
       EI
       RETI
```