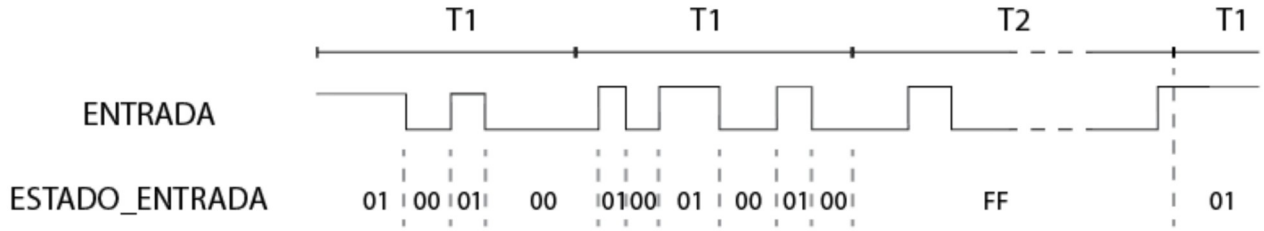
• Nombre y CI en cada hoja • Numere las hojas, indique el total en la primera • Utilice sólo un lado de las hojas	• Incluya un solo problema por hoja • **Sea prolijo** • **Aprobación:** mínimo UN problema

PROBLEMA 1

Se tiene un sistema T80 con un programa principal que utiliza una variable **ESTADO_ENTRADA** cuyo valor depende de la señal **ENTRADA**. Cuando la señal de entrada pasa de 0 a 1, la variable **ESTADO_ENTRADA** debe tomar el valor 0x01 y cuando pasa de 1 a 0 debe cambiar a 0x00. Estos cambios deben provocar una interrupción en el sistema y la rutina de atención a dicha interrupción será la encargada de actualizar la variable.

Por otro lado cada un tiempo T1 se deberá controlar la cantidad de cambios que hubo en la señal ENTRADA. En caso que hayan sido más de 5, se debe de dejar de actualizar la variable durante un tiempo T2 (se garantiza que no ocurrirán más de 255 cambios en un tiempo T1). Durante ese tiempo T2 la variable debe valer 0xFF.



Luego de un reset o luego del tiempo T2, la variable debe valer 0x00 o 0x01 dependiendo si en ese momento ENTRADA es 0 o 1.

Para el control de tiempos debe utilizarse un Timer con un controlador para el manejo de las interrupciones que genera.

No hay otros controladores de interrupciones salvo los que agreguen, pero sí otros puertos que ocupan el rango de I/O 0x40 a 0x9F.

El sistema está equipado con 32K de ROM y 32K de RAM.

El valor de los tiempos es: T1 = 1ms, T2 = 64ms y el reloj del sistema es de 250ns de período. (1ms =1 000 000ns = 250ns * 2^4 * 250, 64 = 2^6).

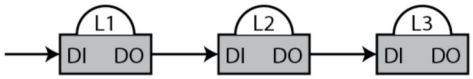
a) Hacer el esquemático de los periféricos y puertos agregados junto con su decodificación. Observación: para generar interrupciones con los flancos de subida y bajada de la señal **ENTRADA**, deben utilizarse 2 controladores de interrupciones.

b) Escribir el código a ejecutarse luego de un reset, el cual debe terminar con la instrucción JP prog_principal. Este código debe inicializar todo lo necesario para la funcionalidad pedida (sistema de interrupciones para trabajar en MODO 2, periféricos, puertos y variables incluida ESTADO_ENTRADA). Las funcionalidades no descritas aquí son inicializadas al comienzo del programa principal. En esta parte también se deberán realizar las declaraciones de reserva de memoria para las variables y tabla de atención a interrupciones.

c) Escribir las rutinas de atención a las interrupciones.

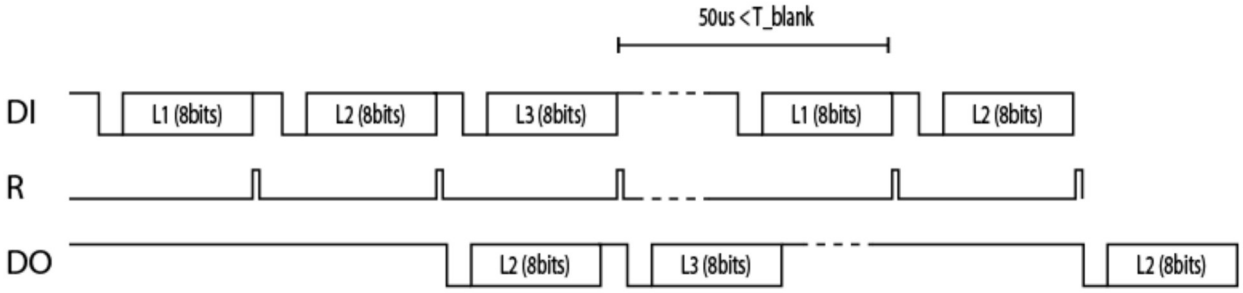
PROBLEMA 2

Los Pixel-Led son utilizados para crear tiras de leds cuya intensidad puede ser controlada en forma individual utilizando solo un cable para transmitir los datos.



No es necesario configurar direcciones, basta con enviar en forma serie uno tras otro los bytes con la palabra de control. El primer byte transmitido es para el Led1, el segundo para el Led 2 y así sucesivamente hasta enviar tantos bytes como leds tenga la tira. Luego de transmitir la ráfaga de bytes, se espera un tiempo mayor a T_blank = 50us y se comienza nuevamente las transmisión.

Cada led recibe una secuencia de bytes por su entrada DI y se queda con el primero que recibe luego de un tiempo T_blank y retransmite los restantes por su salida DO.



Cada Pixel-Led internamente tiene un led rojo y uno verde. El bit más significativo de la palabra de control indica cual debe actualizarse con el nivel indicado por el resto de los bits (0 se actualiza el rojo, 1 el verde).

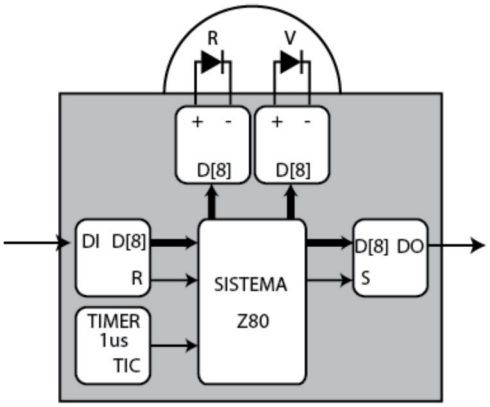
Se desea implementar un Pixel-Led utilizando un sistema basado en un Z80 y periféricos auxiliares.

Se cuenta con un bloque deserializador que recibe bits en forma serie y cada byte recibido lo despliega en su salida D[8], válida solo durante el flanco de subida de R que avisa de la recepción de un nuevo dato.

Se cuenta también con un bloque serializador que envía por su salida el byte que esté en su entrada D[8] en el momento de un flanco de subida en su entrada S.

Para controlar tiempos se cuenta con una señal periódica TIC que genera un flanco de subida cada 1us.

Un conversor D/A genera una corriente proporcional al valor digital de su entrada D[8] de 8 bits, encendiendo el led con su máximo brillo cuando la entrada vale 255 y apagándolo cuando vale 0. Notar que debido al protocolo utilizado solo puede llegarse a 127 ya que se destinan solo 7 bits.

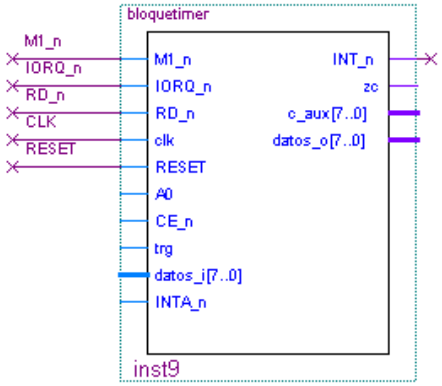


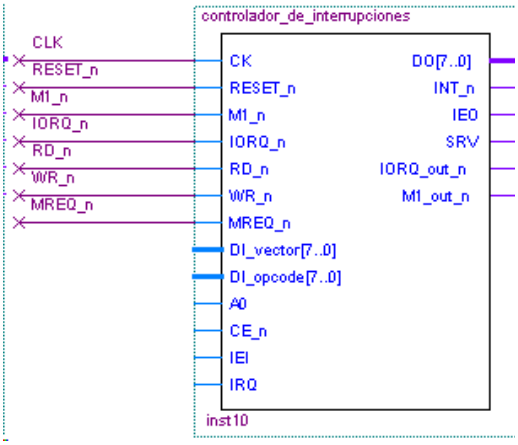
Se pide:

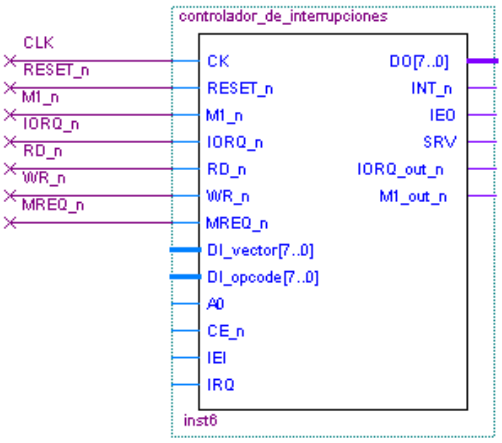
a) Todo el hardware del sistema incluyendo memoria y puertos para interactuar con los periféricos y recibir la señal TIC que debe interrumpir el sistema. Para la memoria se dispone solo de un chip de ROM de 4K y uno de RAM de 16K y deberán decodificarse sin fantasmas.

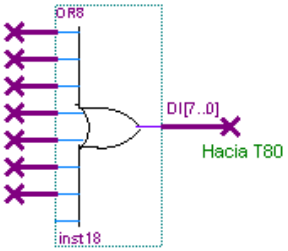
b) Rutina de atención a la interrupción TIC que actualizará una variable en memoria que deberá ser utilizada por el programa principal para saber si transcurrió T_blank desde el último byte recibido.

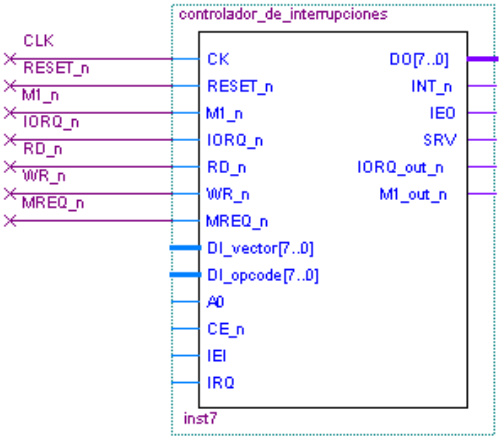
c) Programa principal que inicializa el sistema para trabajar en modo 1 y luego se queda en bucle esperando recibir los bytes. Cada vez que transcurre un tiempo mayor a T_blank sin recibir bytes, se deberá esperar por el primer byte, que es el que tiene el nivel para él y se deberá retransmitir los restantes bytes de la ráfaga.


bloquetimer
M1_n
IORQ_n
RD_n
CLK
RESET
M1_n
IORQ_n
RD_n
clk
RESET
A0
CE_n
trg
datos_i[7..0]
INTA_n
INT_n
zc
c_aux[7..0]
datos_o[7..0]
inst9

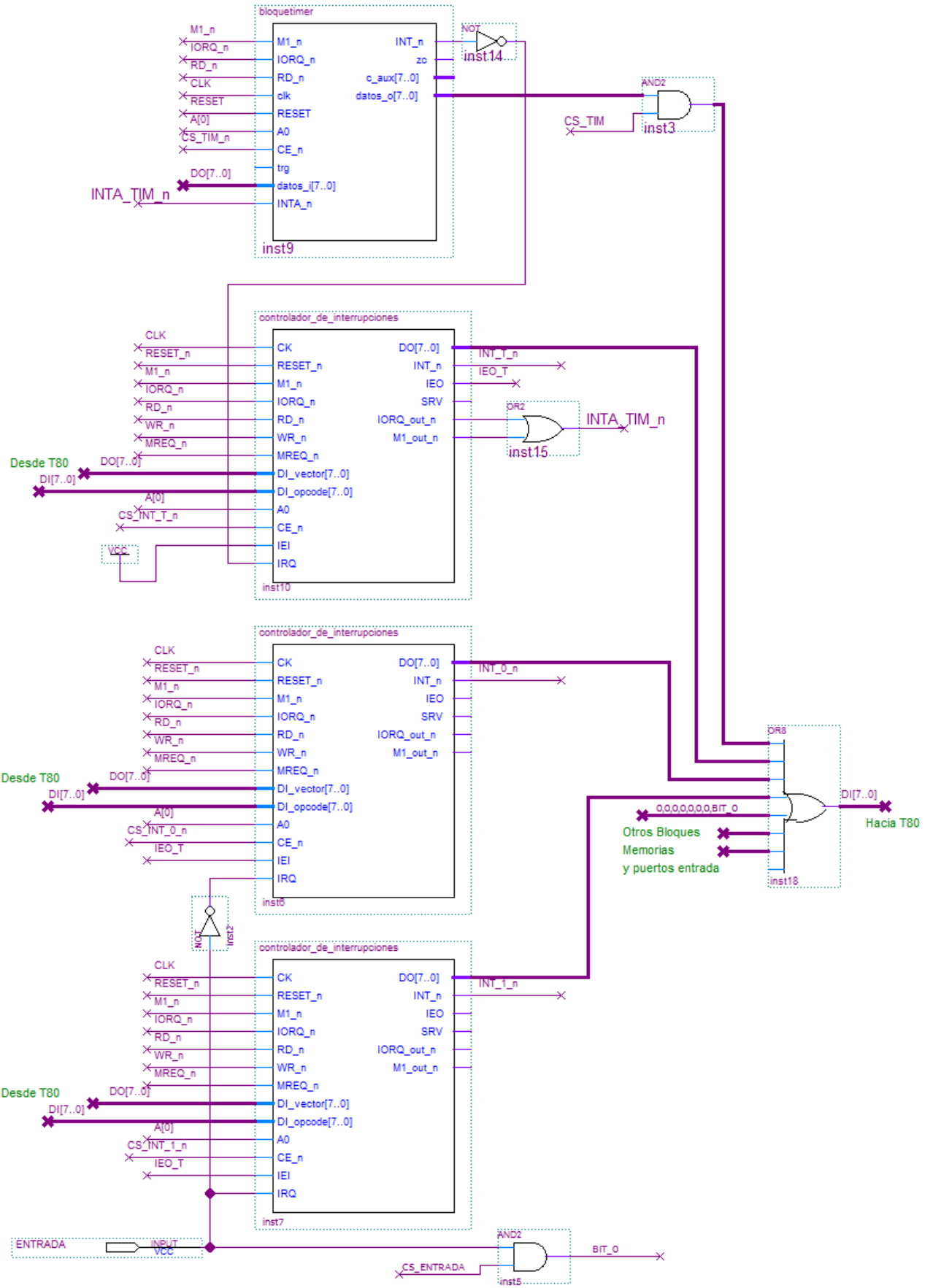

controlador_de_interrupciones
CLK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
CK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
DI_vector[7..0]
DI_opcode[7..0]
A0
CE_n
IEI
IRQ
DO[7..0]
INT_n
IEO
SRV
IORQ_out_n
M1_out_n
inst10

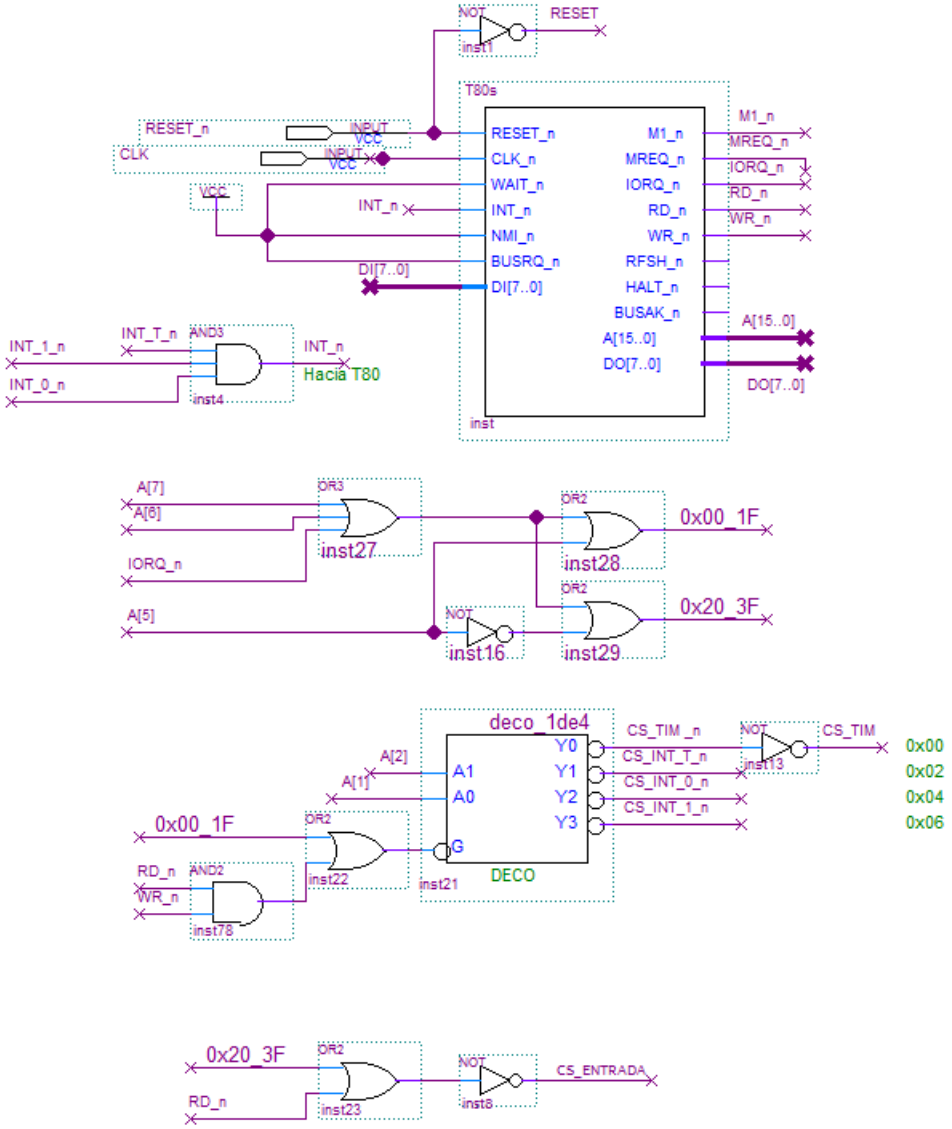

controlador_de_interrupciones
CLK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
CK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
DI_vector[7..0]
DI_opcode[7..0]
A0
CE_n
IEI
IRQ
DO[7..0]
INT_n
IEO
SRV
IORQ_out_n
M1_out_n
inst6


OR8
DI[7..0]
Hacia T80
inst18


controlador_de_interrupciones
CLK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
CK
RESET_n
M1_n
IORQ_n
RD_n
WR_n
MREQ_n
DI_vector[7..0]
DI_opcode[7..0]
A0
CE_n
IEI
IRQ
DO[7..0]
INT_n
IEO
SRV
IORQ_out_n
M1_out_n
inst7

PROBLEMA 1 – SOLUCION





b)

```
TIMER_CTE  equ 0x00
TIMER_CTRL equ 0x01
CI_TIMER   equ 0x02
CI_ENT0    equ 0x04
CI_ENT1    equ 0x06
ENTRADA    equ 0x00
;;int hab, SW reset, pre= 2^4=16
CW_T1 equ 1010 0100
;;ídem excepto pre= 2^10=1024
CW_T2 equ 1010 1010
; 250ns *  2^4 * 250 =  1ms
; 250ns * 2^10 * 250 = 64ms
CTE        equ 249
CONTROLANDO_T1 equ 1
CONTROLANDO_T2 equ 2
MAX_CAMBIOS    equ 4
VI_ENT0        equ 0x00
VI_ENT1        equ 0x02
VI_TIMER       equ 0x04

org 0x0000
   ld SP, 0
   im2
   ld A, tabla_int/256
   ld I,A
     ;;; controladores interr
   ld A, VI_ENT0
   OUT (CI_ENT0), A
    ; borra peticiones pendientes
   OUT (CI_ENT0+1), A
   ld A, VI_ENT1
   OUT (CI_ENT1), A
   OUT (CI_ENT1+1), A
```

```
   ld A, VI_TIMER
   OUT (CI_TIMER), A
   OUT (CI_TIMER+1), A
     ;;; timer
   ld A,CW_T1
   out (TIMER_CTRL), A
   ld A,CTE
   out (TIMER_CTE), A
     ;;; variables
   ld A, 0
   ld (CAMBIOS_ENTRADA), A
   ld A, CONTROLANDO_T1
   ld (CONTROL_TIEMPO), A
   in A,(ENTRADA)
   and 0x01
   ld (ESTADO_ENTRADA), A
   ei
   jp prog_ppal

org 0x0800
tabla_int:
  DW RI_entrada_a_0
  DW RI_entrada_a_1
  DW RI_Timer

org 0x4000
prog_ppal: ....

org 0x8000
ESTADO_ENTRADA:  DB
;;; cantidad de cambios durante T1
CAMBIOS_ENTRADA: DB
 ; 1= controlando T1, 2= controlando T2
CONTROL_TIEMPO:  DB
```

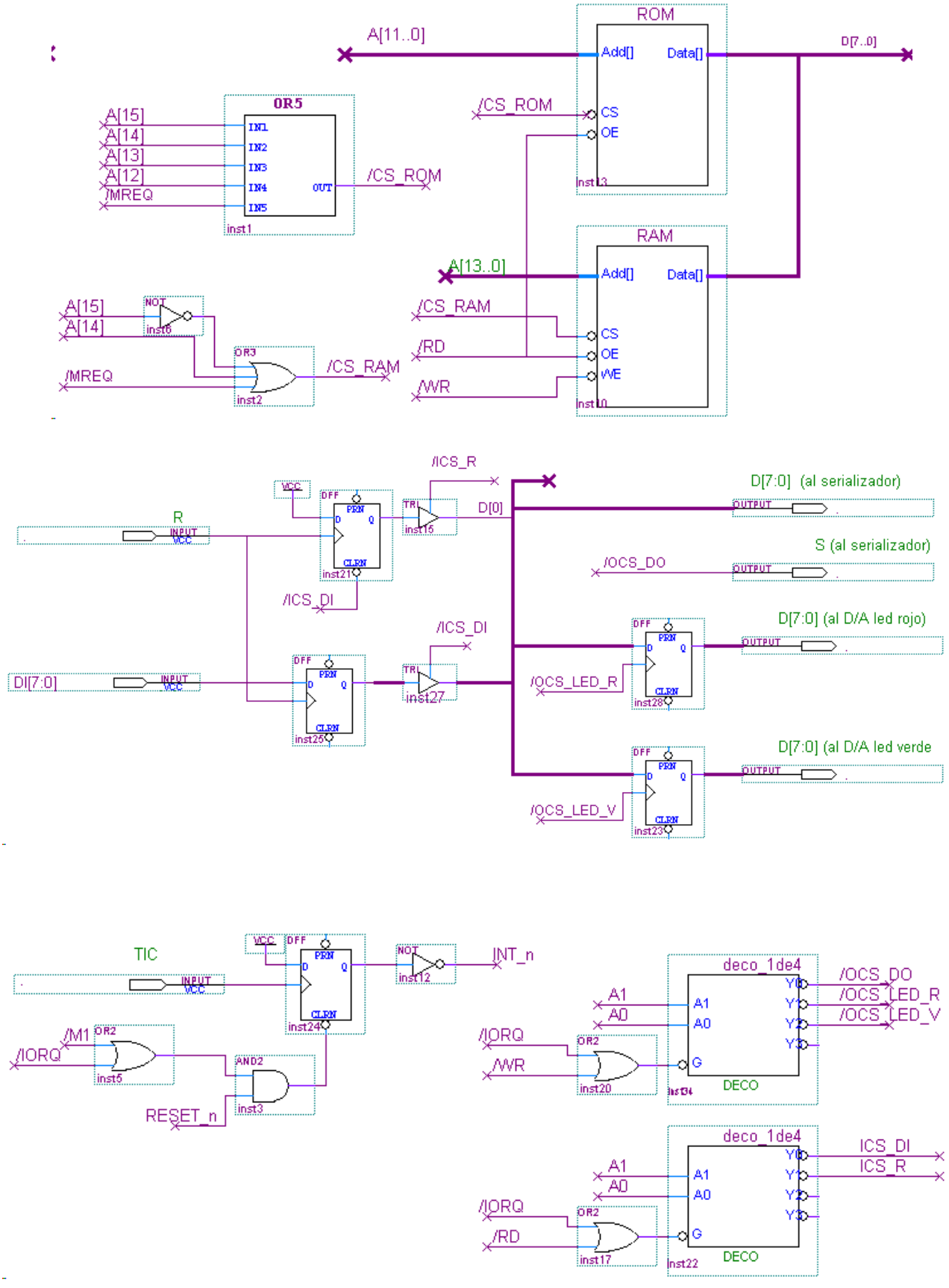
c)

```
; si controlando_T1 {
;    ESTADO_ENTRADA=0
;    Inc CAMBIOS_ENTRADA
;    }
RI_entrada_a_0:
   ei
   push AF
   ld A,(CONTROL_TIEMPO)
   cp CONTROLANDO_T2
   jp Z, fin_RI_entrada_a_0
   ld A, (CAMBIOS_ENTRADA)
   inc A
   ld (CAMBIOS_ENTRADA), A
   ld A, 0x00
   ld (ESTADO_ENTRADA),A
fin_RI_entrada_a_0 :
   pop AF
   reti

RI_entrada_a_1:
; si controlando_T1 {
;    ESTADO_ENTRADA=1
;    Inc CAMBIOS_ENTRADA
;    }
   ei
   push AF
   ld A,(CONTROL_TIEMPO)
   cp CONTROLANDO_T2
   jp Z, fin_RI_entrada_a_1
   ld A, (CAMBIOS_ENTRADA)
   inc A
   ld (CAMBIOS_ENTRADA), A
   ld A, 0x01
   ld (ESTADO_ENTRADA),A
fin_RI_entrada_a_1:
   pop AF
   reti

RI_TIMER:
; si (controlando_T1) {
;      si (menos de MAX_CAMBIOS){
;          cambios = 0
;       }else{
;          estado_entrada = 0xFF
;          arranco timer T2
;          control_tiempo = controlando_T2
;       }
; }else{
;    cambios = 0
;    estado_entrada = entrada
;    arranco timer T2T1
;    control_tiempo = controlando_T2T1
; }
   ei
   push AF
   ld A,(CONTROL_TIEMPO)
   cp CONTROLANDO_T1
   jp Z, termino_T1
```

```
   jp termino_T2
termino_T1:
   ld A, (CAMBIOS_ENTRADA)
   cp MAX_CAMBIOS
   jp M, muchos_cambios
pocos_cambios:
   ld A, 0
   ld (CAMBIOS_ENTRADA),A
   jp fin_ri_timer
muchos_cambios:
   ld A, 0xFF
   ld (ESTADO_ENTRADA), A
   ld A,CW_T2
   out (TIMER_CTRL), A
   ld A,CTE
   out (TIMER_CTE), A
   ld A, CONTROLANDO_T2
   ld A,(CONTROL_TIEMPO)
   jp fin_ri_timer
termino_T2:
   ld A, 0
   ld (CAMBIOS_ENTRADA), A
   in A,(ENTRADA)
   and 0x01
   ld (ESTADO_ENTRADA), A
   ld A,CW_T1
   out (TIMER_CTRL), A
   ld A,CTE
   out (TIMER_CTE), A
   ld A, CONTROLANDO_T1
   ld (CONTROL_TIEMPO), A
fin_ri_timer:
   pop AF
```

PROBLEMA 2 – SOLUCION

a)



b)

```
org 38h
 ;; preservo registros
 push AF

 ;; incremento variable de tiempo
 ld A, (TIMER)
 inc A
 ld (TIMER), A

 ;; restauro registros
 ;; habilito int y retorno
 pop AF
 ei
 reti
```

c)

```
DI equ 0x00
R  equ 0x01
DO equ 0x00
LED_ROJO equ 0x02
LED_VERDE equ 0x02

org 0000h
 ld SP,0xA000
 im1
 ei

; espero primer byte luego de un
reset o T_blank
esperoR_inicial:
 in A,(R)
 bit 0,A
 jp Z, esperoR_inicial

 ; envío valor recibido a conversor
AD correspondiente.
 in A,(DI)
 bit 7,A
 jp Z, ROJO
VERDE:
 and 0x7F
 out (LED_VERDE), A
jp reinicio_tiempo
ROJO:
 out (LED_ROJO), A

reinicio_tiempo:
 ; reinicio cuenta de tiempo.
 ld A, 0
 ld (TIMER), A

esperoR_siguientes:
 ; si no ha llegado pulso en R
 ;  verifico tiempo transcurrido
 ;  - si es < T_blank: vuelvo a
esperar pulso R
 ;  - si es >= T_blank: voy a
esperar R de primer byte
 ; si llegó pulso en R:
 ;  - retransmito byte
 ;  - reinicio cuenta de tiempo
 in A,(R)
 bit 0,A
 jp NZ, retransmito
 ld A,(TIMER)
 cp 50
 jp M, esperoR_siguientes
 jp esperoR_inicial

retransmito:
 in A,(DI)
 out (DO), A
 ; reinicio cuenta de tiempo.
 ld A, 0
 ld (TIMER), A
 jp esperoR_siguientes

org 8000h
TIMER DB
```