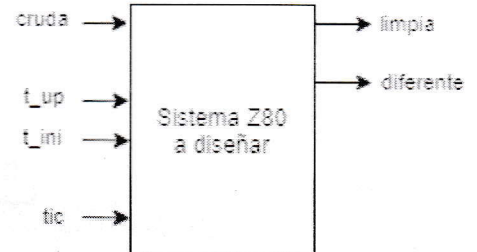


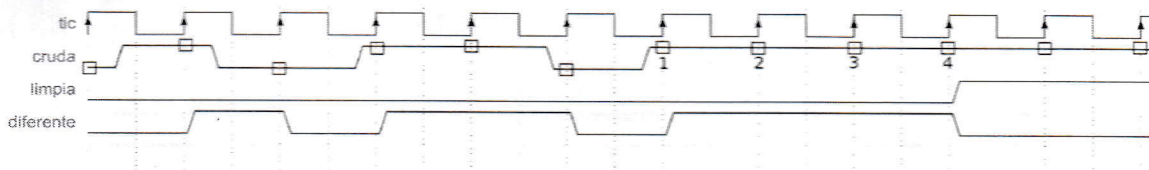
- | | |
|--|--|
| <ul style="list-style-type: none">• Mínimo para aprobar: un problema• Nombre y CI en cada hoja• Numere las hojas, indique el total en la primera | <ul style="list-style-type: none">• Utilice sólo un lado de las hojas• Incluya un solo problema por hoja• Sea prolijo |
|--|--|

PROBLEMA 1

Un sistema basado en Z80 recibe como entrada la señal **cruda** (de un bit) que presenta rebotes y ruido, y debe generar como salida la señal **limpia** libre de rebotes y pulsos espurios. Para eso se muestrea la entrada **cruda** periódicamente. Un cambio en la entrada se traslada a la salida **limpia** recién cuando el nuevo valor se mantuvo durante **N** muestras consecutivas. El valor inicial de **limpia** debe copiarse del valor de **cruda** leído durante la inicialización.



Además el sistema debe dar una salida **diferente** que debe valer 1 mientras los valores muestreados de la entrada sean diferentes a la salida actual. El diagrama de tiempos muestra un ejemplo del comportamiento deseado para una transición de 0 a 1 con rebotes para $N=4$.



Inicialmente $N = N_{min}$ con $N_{min} = 2$, pero puede aumentarse hasta un valor $N_{max} = 128$ mediante el siguiente mecanismo. Cada vez que llegue un flanco de subida en la entrada **t_{up}**, el valor de **N** debe duplicarse, salvo que ya estuviera en el valor **N_{max}** en cuyo caso se mantiene el valor. Por otra parte, cada vez que llegue un flanco de subida en **t_{ini}**, **N** debe volver al valor inicial **N_{min}**.

Para muestrear periódicamente la entrada se cuenta con una señal periódica **tic** que interrumpirá al sistema en **modo 1**.

El programa principal consistirá en un loop infinito que además de atender otras tareas deberá ajustar el valor de **N** de acuerdo a las entradas **t_{up}** y **t_{ini}**.

Se pide:

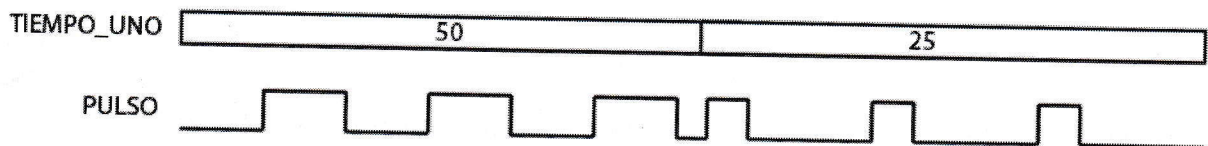
- a) Hardware del sistema incluyendo la memoria y los puertos necesarios para la funcionalidad descripta. El sistema cuenta también con puertos para otras funciones que ocupan el intervalo 0x00-0x3F del espacio de E/S. Para la memoria se cuenta solamente con un chip de ROM de 8KB y un chip de RAM de 16KB, que deben decodificarse sin fantasmas.
- b) Rutina de atención a la interrupción periódica.
- c) Inicialización, loop de programa principal y directivas de reserva de memoria. Las funciones del sistema no descriptas en este problema no utilizan interrupciones. Para atenderlas debe invocarse la subrutina **init_otros()** durante la inicialización, y la subrutina **atiendo_otros()** dentro del loop del programa principal.

PROBLEMA 2

Se debe de diseñar un sistema basado en un T80 que genere pulsos de período 2560 ms, con un ciclo de trabajo variable controlado por el usuario (porcentaje del período en que la señal está en 1). Como base de tiempo se usará un Timer que deberá de interrumpir cada $2560 \text{ ms} / 100 = 25.6 \text{ ms}$.



El sistema tiene una entrada de 8 bits TIEMPO_UNO que indica el porcentaje del período que la salida PULSO deberá valer 1, el tiempo restante deberá ser 0. Cuando se detecta un cambio en TIEMPO_UNO (el valor es diferente al anterior), se debe reiniciar la generación de pulsos, incluso reiniciando el Timer. Se asegura que el valor siempre estará entre 1 y 99.



Además se debe dotar al dispositivo de una entrada DETENER que deberá interrumpir con la mayor prioridad y detener toda generación de pulsos y funcionamiento del sistema hasta que sea reinicializado externamente mediante RESET. Cuando se detecta esta condición, la salida PULSO debe valer 0 y DETENIDO debe valer 1.

El software debe de estructurarse en un programa principal que monitorea la entrada TIEMPO_UNO, y dos rutinas de atención a interrupción en **modo 2**: una que atiende al Timer para manejar la salida PULSO y y otra que atiende a la entrada DETENER.

Se pide:

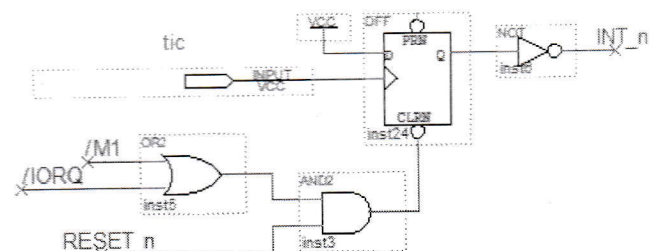
- HW del sistema: controladores de interrupción, Timer y puertos de entrada y salida. Asumir que ya está implementada la memoria con 32K ROM y 32K RAM.
- Código a ejecutar luego de un reset que inicialice procesador, periféricos, variables y puertos de salida. También directivas al ensamblador para reserva de espacio en memoria para variables. El valor inicial del ciclo de trabajo debe leerse durante la inicialización desde el puerto TIEMPO_UNO.
- Programa principal.
- Rutinas de atención a interrupciones.

Notas:

- Período del reloj del sistema $T_{ck} = 250 \text{ ns}$
- Período del pulso a generar = 2560 ms
- $250 \text{ ns} * 100 * 2^{10} = 25.6 \text{ ms}$



a) Hardware



b) rutina de atención a interrupción c) inic., prog. Ppal y directivas

org 0x0038

.text

isr:

```
;; preservó estado
;; si (cruda != limpia_actual) {
;;     diferente = 1
;;     cont--
;;     si(cont == 0){
;;         port_limpia = cruda
;;         limpia_actual = cruda
;;         diferente = 0
;;         cont = N
;;     }
;; }else{
;;     cont = N
;; }

push af
push bc
in a, (cruda)
and 0x01
ld b, a
ld a, (limpia_actual)
cp b
jr z, else_cruda
;; si(cruda != limpia_actual) {
;;     diferente = 1
;;     cont--
ld a, 1
out (diferente), a
ld a, (cont)
dec a
ld (cont), a
jr nz, fin_si_cont_0
;; si(cont == 0){
;;     port_limpia = cruda
;;     limpia_actual = cruda
;;     diferente = 0
;;     cont = N
;; }
ld a, b
out (limpia), a
ld (limpia_actual), a
ld a, 0
out (diferente), a
ld a, (N)
ld (cont), a
fin_si_cont_0:
jr fin_si_cruda
else_cruda:
;; }else{
;;     cont = N
ld a, (N)
ld (cont), a
fin_si_cruda:
pop bc
pop af
ei
ret
```

```
t_up      equ 0x80
t_ini     equ 0x81
cruda     equ 0x82
limpia    equ 0x80
diferente equ 0x81
clr_t_up  equ 0x82
clr_t_ini equ 0x83

N_min     equ 2
N_max     equ 128
```

inicio:

```
;; stack en tope de RAM
ld sp, 0
im 0
;; limpia = cruda
in a, (cruda)
and 0x01
ld (limpia_actual), a
out (limpia), a
;; diferente = 0
ld a, 0
out (diferente), a
;; borro FFs
out (clr_t_up), a
out (clr_t_ini), a
;; N = N_min, cont = N
ld a, N_min
ld (N), a
ld (cont), a
;; init_otros
call init_otros
;; habilito interrupciones
ei
jp main
```

org 0x0038

isr:

;; incluida en b)

...



main:

```
;; forever{
;;   si t_up{
;;     borro ff t_up
;;     si N != N_max{
;;       N = 2 * N
;;     }
;;   }
;;   si t_ini{
;;     borro ff t_ini
;;     N = N_min
;;   }
;;   call atiando_otros()
;; }
```

```
in a, (t_up)
bit 0, a
jr z, finssi_t_up
;; si t_up{
;;   borro ff t_up
;;   si N != N_max{
;;     N = 2 * N
;;   }
out (clr_t_up), a
ld a, (N_max)
ld b, a
ld a, (N)
cp b
jr z, finssi_N
```

```
sla a
ld (N), a
finssi_N:
finssi_t_up:
in a, (t_ini)
bit 0, a
jr z, finssi_t_ini
;; si t_ini{
;;   borro ff t_ini
;;   N = N_min
;; }
out (clr_t_ini), a
ld a, N_min
ld (N), a
finssi_t_ini:
call atiando_otros
jr main
```

```
atiando_otros:
init_otros:
ret
```

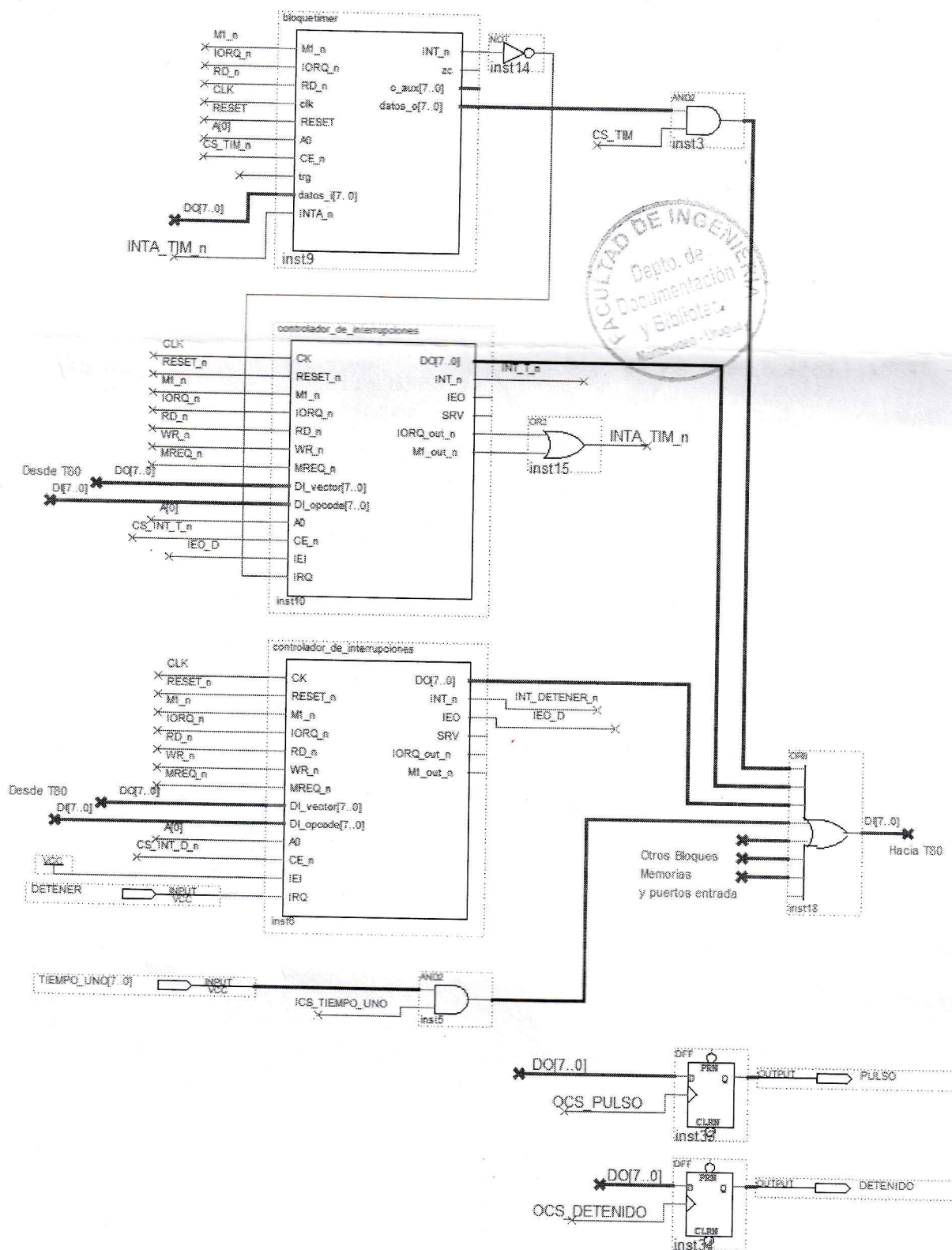
```
.data
cont: db 0
limpia_actual: db 0
N: db 0
```

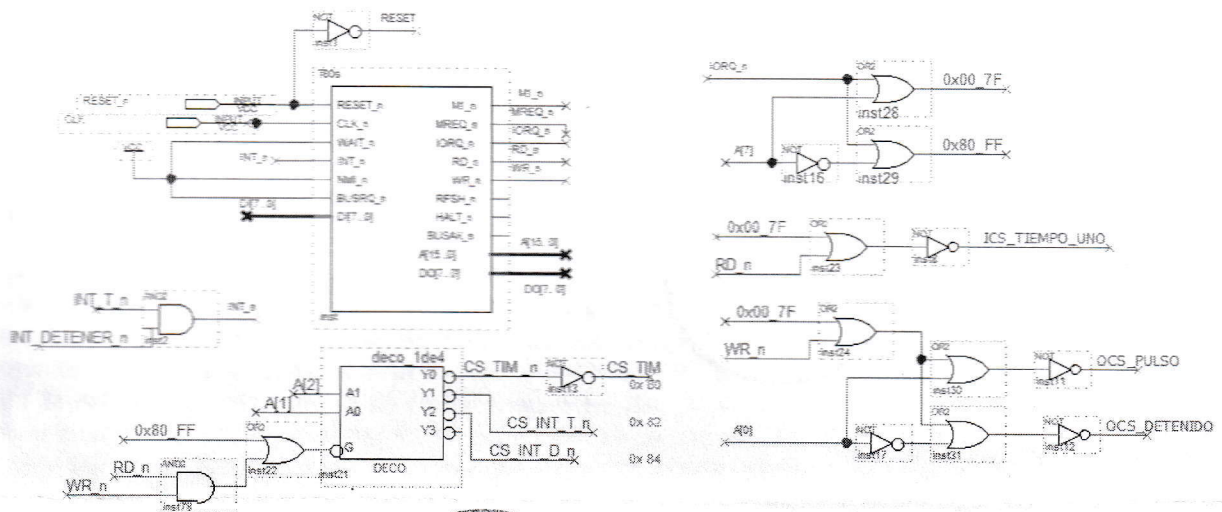
.end



SOLUCIÓN PROBLEMA 2

a) Hardware





b)

PULSO_O EQU 00h
DETENIDO_O EQU 01h

TIEMPO_UNO_I EQU 00h

TIMER_CTE_IO EQU 80h
TIMER_CTL_IO EQU 81h

CI_TIMER_VW EQU 82h
CI_TIMER_ST EQU 83h

CI_DETENER_VW EQU 84h
CI_DETENER_ST EQU 85h

VI_DETENER EQU 00h
VI_TIMER EQU 02h

TIMER_CTE EQU 99 ; 100-1
; int |X| reset | auto | pres = 10
TIMER_CTL EQU 1010 1010b

ORG 8000h
TIEMPO_PULSO_REF DB
CUENTA DB
FLAG_DETENIDO DB

ORG 3000h
TABLA_INT:
DW rutint_detener
DW rutint_timer

org 0000h

; cosas necesarias para interrupción

ld SP, 0000h
ld HL, TABLE_INT
ld A, H
ld I, A
IM 2

; inicializo variables y puertos salida

ld A, 0
ld (CUENTA), A
ld (FLAG_DETENIDO), A
out (DETENIDO_O), A
ld A, FFh ; empiezo generando pulso.
out (PULSO_O), A

; porcentaje inicial de tiempo en 1
in A, (TIEMPO_UNO_I)
ld (TIEMPO_PULSO_REF), A

;timer

out (TIMER_CTE_IO), TIMER_CTE
out (TIMER_CTL_IO), TIMER_CTL

; controladores de interrupciones

ld A, VI_DETENER
out (CI_DETENER_VW), A
out (CI_DETENER_ST), A

ld A, VI_TIMER
out (CI_TIMER_VW), A
out (CI_TIMER_ST), A

ei
jp ppal

c)

org 0100h
ppal:
; verifico si esta detenido
ld A, (FLAG_DETENIDO)
cp FFh
jp Z, detenido

; verifico si cambio
in A, (TIEMPO_UNO_I)
cp (TIEMPO_PULSO_REF)
jp Z, ppal

cambio:
ld (TIEMPO_PULSO_REF), A
;timer
out (TIMER_CTE_IO), TIMER_CTE



out (TIMER_CTL_IO), TIMER_CTL
;salidas
ld A, FFh
out (PULSO_O), A
; variable cuenta
ld A, 0
ld (CUENTA), A

jp ppal

detenido:
ld A, 00h
out (PULSO_O), A

jp ppal

d)

org 1000h
rutint_detener:
ei
push AF

ld A, FFh
ld (FLAG_DETENIDO), A
out (DETENIDO_O), A

pop AF
reti

rutint_timer:
ei
push AF

; verifico si esta detenido
ld (FLAG_DETENIDO), A
cp FFh
jp fin_rutint_timer

ld A, (CUENTA)
inc A
LD (CUENTA), A
cp 100
jp Z, reinicio_ciclo

ld B, A
ld A, (TIEMPO_PULSO_REF)
cp A, B
jp P, salida_0
jp salida_1

reinicio_ciclo:
ld A, 0
ld (CUENTA), A

; luego pongo salida en 1
salida_1:
ld A, FFh
out (PULSO_O), A
jp fin_rutint_timer

salida_0:
ld A, 00h
out (PULSO_O), A

fin_rutint_timer:
pop AF
reti

8/8