Ejercicio 1 (18 pts.)

a)

A[7..0]: $T1+T2+T_{WA}+T3H + NT_W - t6_{(max)} - tor_{(max)} - t_{D(max)} >= t25$

IORQ\: $T2+T_{WA}+T3H + NT_W - t27_{(max)} - tor_{(max)} - t_{D(max)} >= t25$

RD\: $T2+T_{WA}+T3H + NT_W - t24_{(max)} - t_{D(max)} >= t25$

b)

A[7..0]: $50ns+50ns+50ns+25ns - 57ns - 12ns - t_{D(max)} >= 12ns \rightarrow t_{D(max)} =< 94ns$

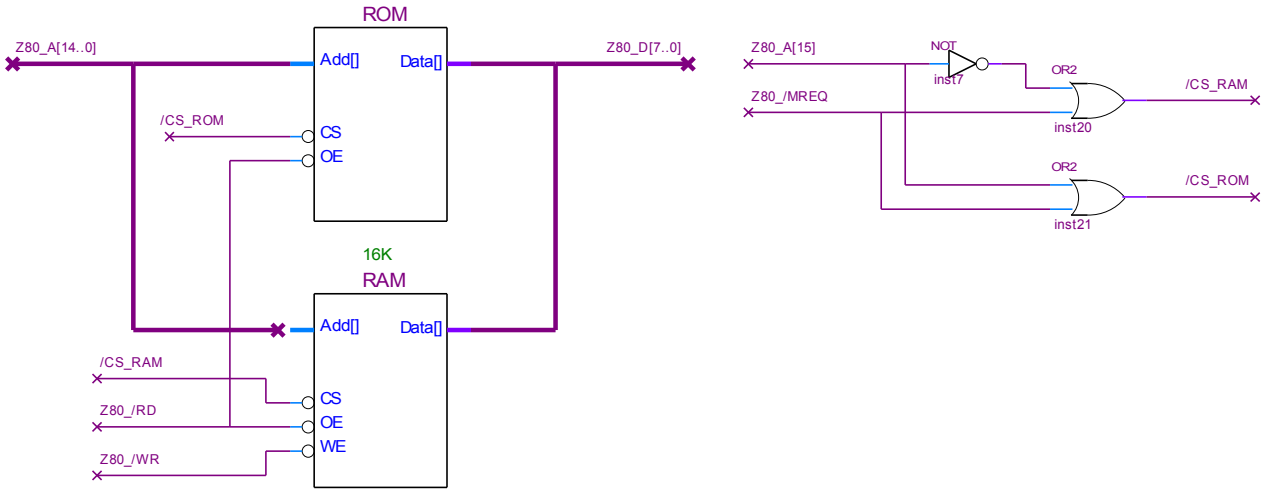
IORQ\: $50ns+50ns+25ns - 40ns - 12ns - t_{D(max)} >= 12ns \rightarrow t_{D(max)} =< 61ns$

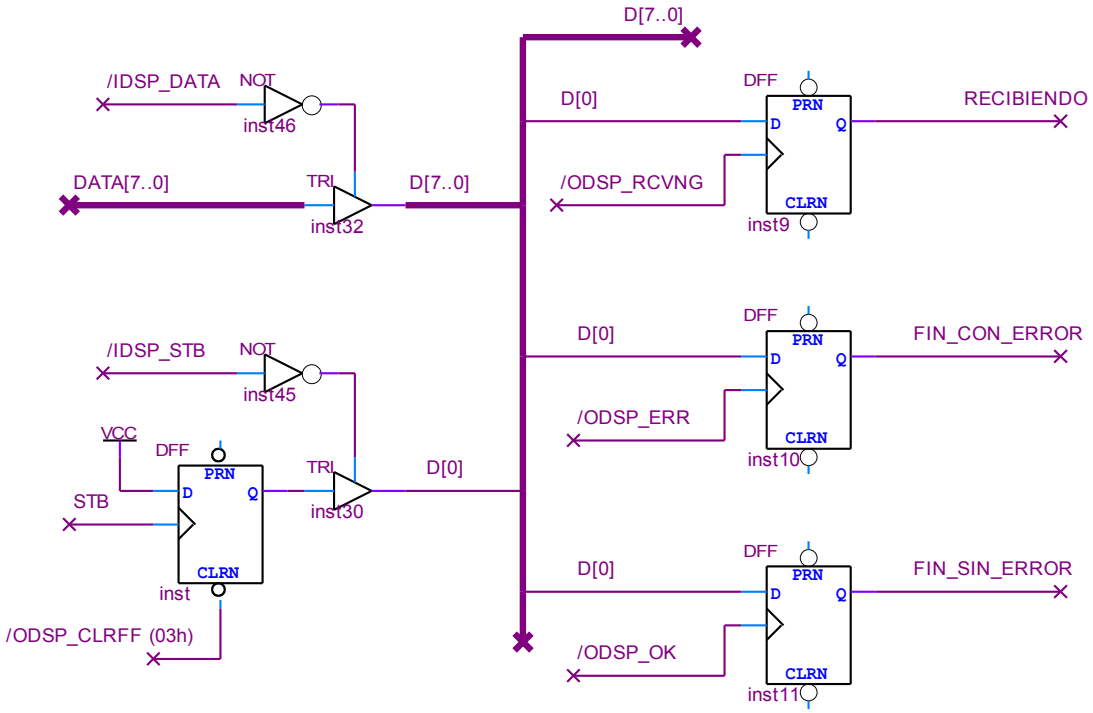
RD\: $50ns+50ns+25ns - 40ns - t_{D(max)} >= 12ns \rightarrow t_{D(max)} =< 73ns$

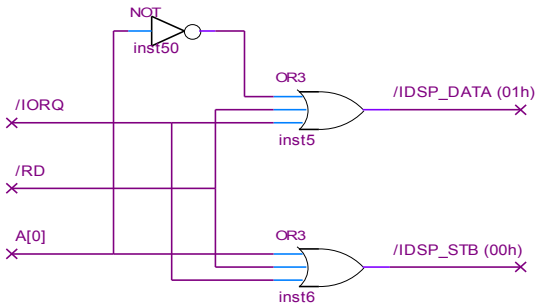
Requerimiento para el dispositivo: $t_{D(max)}$ <= 61 ns

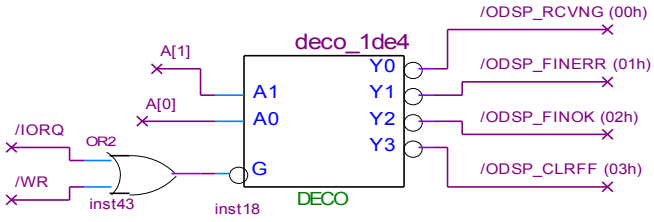
Ejercicio 2 (18 pts.)

a) Hardware







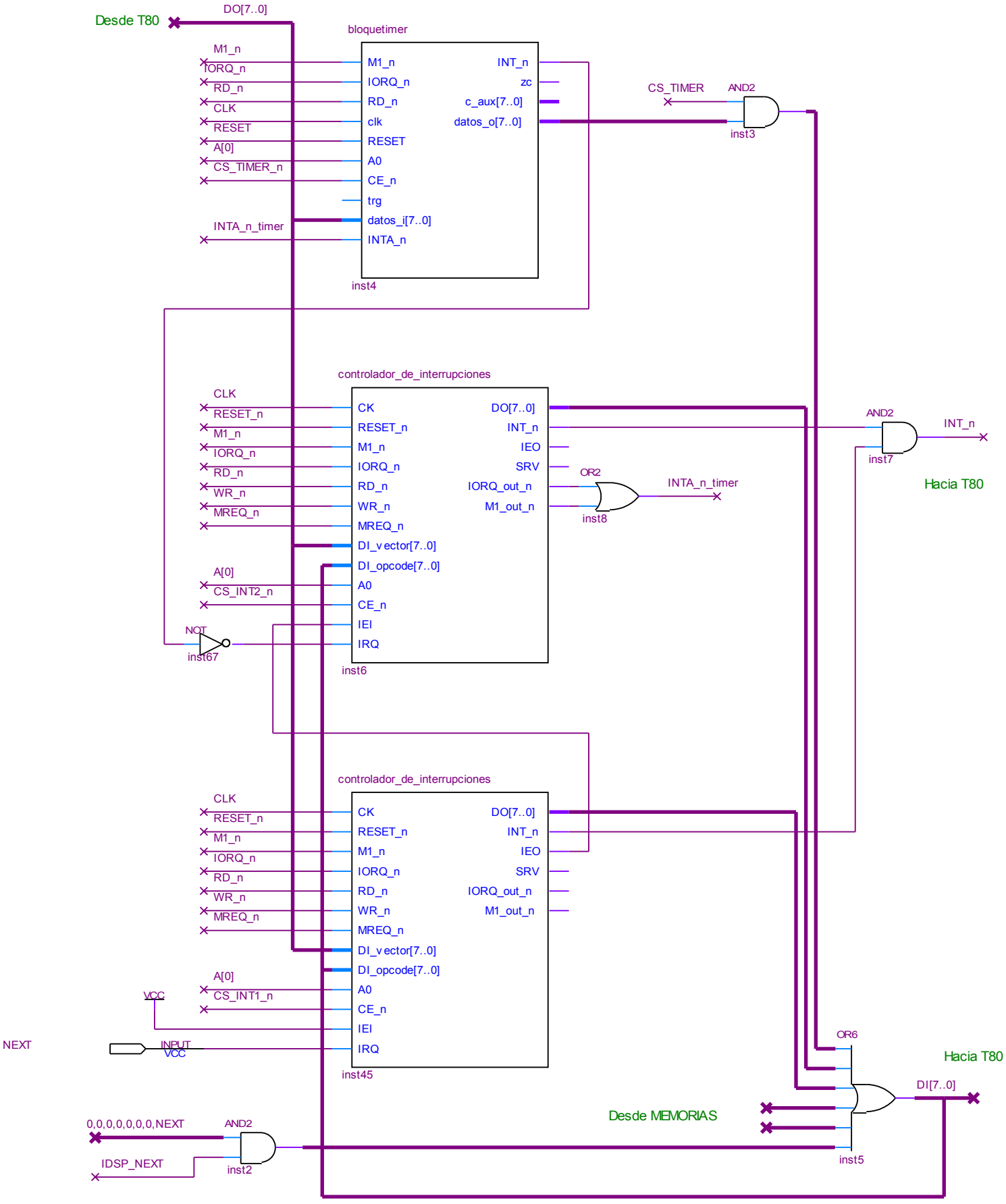


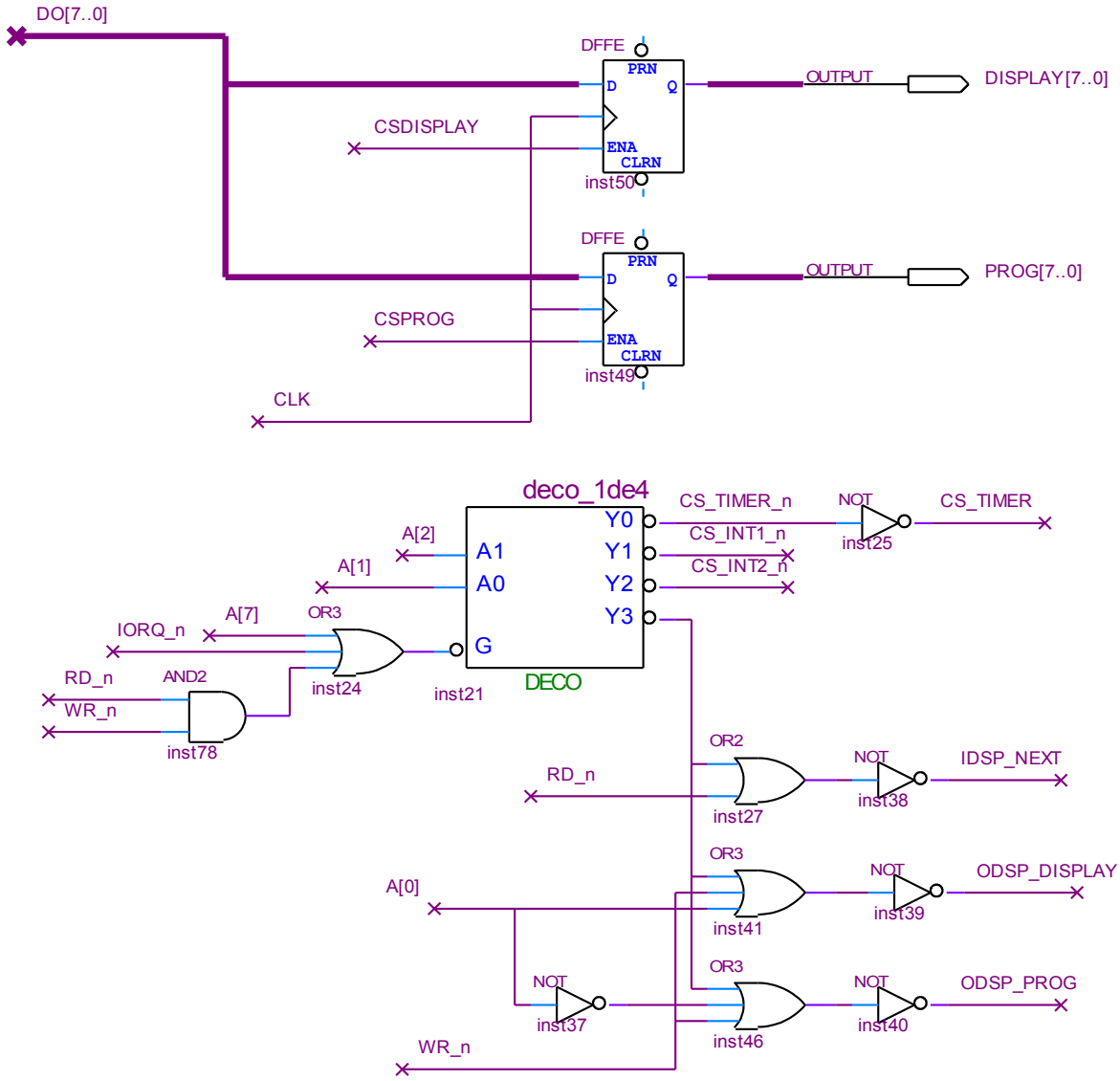
b) Software

```
DATA            equ     0
FLAG            equ     1
RECIBIENDO      equ     0
FIN_CON_ERROR   equ     1
FIN_SIN_ERROR   equ     2
CLEAR_FLAG      equ     3

org 0x0000
        LD SP, 0        ;; inicialización
        LD A, 0
        OUT (RECIBIENDO), A
        OUT (FIN_CON_ERROR), A
        OUT (FIN_SIN_ERROR), A
        OUT (CLEAR_FLAG), A

inicio:
        CALL leo_dato           ;; byte de inicio
        CP 0xFF
        JP NZ, inicio

        LD A, 0
    ;; desactiva salidas FIN por inicio de mensaje
        OUT (FIN_CON_DATO), A
        OUT (FIN_SIN_DATO), A
    ;; inicializa variable de error
        LD (error_check), A
        LD A, 1
    ;; activa salida recibiendo
        OUT (RECIBIENDO), A

    ;; cantidad de bytes de datos
        CALL leo_dato
        LD B, A
        LD HL, error_check
loop_data:
        CALL leo_dato           ;; datos
        XOR (HL)
        LD (HL), A
        DJNZ loop_data

        CALL leo_data           ;; byte de error
        LD C, A
        LD A, 0
        OUT (RECIBIENDO), A     ;; desact.  recibiendo
        LD A, C

        CP (HL)         ;; comparación de
                        ;; error recibido con el
                        ;; calculado
        JP NZ, hay_error

        LD A, 1
        OUT (FIN_SIN_ERROR), A
        JP inicio

hay_error:
        LD A, 1
        OUT (FIN_CON_ERROR), A
        JP inicio

;; subrutina que espera que se active la bandera
;; FLAG (flanco de subida en STB), lee el
;; dato y lo devuelve en el registro A
org 0x0200
leo_dato:
        IN A, (FLAG)    ;; espera por bandera
        AND 1
        JP Z, leo_dato
        IN A, (DATA)            ;; lectura de dato
        OUT (CLEAR_FALG), A     ;; borro bandera
        RET

org 0x8000
error_check: DB                 ;; variable para
                                ;; cálculo de error
```

Ejercicio 3 (64 pts.)

Parte a) Conexionado puertos





b) Rutinas atención interrupciones

```
isr_next:
; incremento v_display mod 8
; p_display = v_display
; ini_timer
; faltan = 2
   ei
   push af
   ld a, (v_display)
   inc a
   and 7
   ld (v_display), a
   out (P_DISPLAY), a

   ld a, TIM_CTE
   out (base_tim + off_cte), a
   ld a, TIM_CTRL_SLOW
   out (base_tim + off_ctrl), a

   ld a, 2
   ld (faltan), a
   pop af
   reti
```

```
rutint_timer:
; si pulsador inactivo{
;    prog = display
;    desactivo timer
; }else{
;    si !faltan {
;        faltan = faltan - 1
;    }else{
;    incremento display mod 8 y muestro
;    }
;}
   ei
   push af
   push hl
   in A,(P_NEXT)
   bit 0, a
   jp z, activo
            ; si pulsador inactivo{
            ;    prog = display
            ;    desactivo timer
   ld A,(v_display)
   out (P_PROG), A
   ld a, CNT_CTRL_DI
   out (base_tim + off_ctrl), a
   jr fin_isr_timer
```

```
activo:
            ; }else{
            ;   si faltan {
            ;     decrem. faltan
   ld a, (faltan)
   cp 0
   jr z, else_faltan
   dec a
   ld (faltan), a
   jr fin_isr_timer
else_faltan:
;    }else{
;        incremento mod 8
;        y muestro display

   ld a, (v_display)
   inc a
   and 7
   ld (v_display), a
   out (P_DISPLAY), a

fin_isr_timer:
   pop hl
   pop af
   reti
```

Parte c) Inicialización y directivas

```
; ports timer
base_tim   equ  0x00
off_cte    equ  0
off_ctrl   equ  1
off_flag   equ  0
off_cuenta equ  1

; ports controladores interrupciones
base_int_next equ  0x02
base_int_tim  equ  0x04
off_vector    equ  0
off_status    equ  1


; ctes timer
; 256ms = (1/4MHz)*2^pre * (TIM_CTE+1)
; 256ms = (1/4MHz)* 4096 * 250

TIM_CTE     equ  249
   ; ei, x, rst=1, trg auto, pre = 12
TIM_CTRL    equ  10101100B
   ; di, x, reset=1, xxxxx
TIM_CTRL_DI equ  00100000B
VEC_NEXT  equ 0
VEC_TIM   equ 2

P_NEXT    EQU 6
P_DISPLAY EQU 6
P_PROG    EQU 7
```

```
       org 0
init:
    ld SP,0000h
    im 2
    ld hl, tab_int
    ld a, h
    ld i, a
    ld A, 00
    ld (v_display), A

    out (P_DISPLAY), A
    out (P_PROG), A

  ;; inic cont. interrupciones
    ld a, VEC_NEXT
      out (base_int_next+off_vector), a
    ld a, VEC_TIM
      out (base_int_tim+off_vector), a

      out (base_int_next+off_status), a
      out (base_int_tim+off_status), a

    ei

      jp ppal

   ;; ejemplo programa principal
ppal:
   jp ppal

;;;;;;;;;;;;;;;;;;;;;;

   ;; tabla interrupciones en ROM
   org 0x100
tab_int:
   dw isr_next
   dw rutint_timer



   ;; variables en comienzo RAM
org 0x8000

v_display:  db 0
faltan:     db 0
```

Solución alternativa:
Se podría haber conectado NEXT también a la entrada trg del TIMER y programarlo para que arranque por un flanco en trg.

En ese caso el timer debe programarse en la inicialización para que arranque con el flanco de bajada del primer pulso.

Luego en la rutina de interrupción del timer, cuando se detecta que se libera el pulsador NEXT en lugar de detener al timer se lo debe reprogramar para que arranque con el siguiente flanco de bajada en trg.