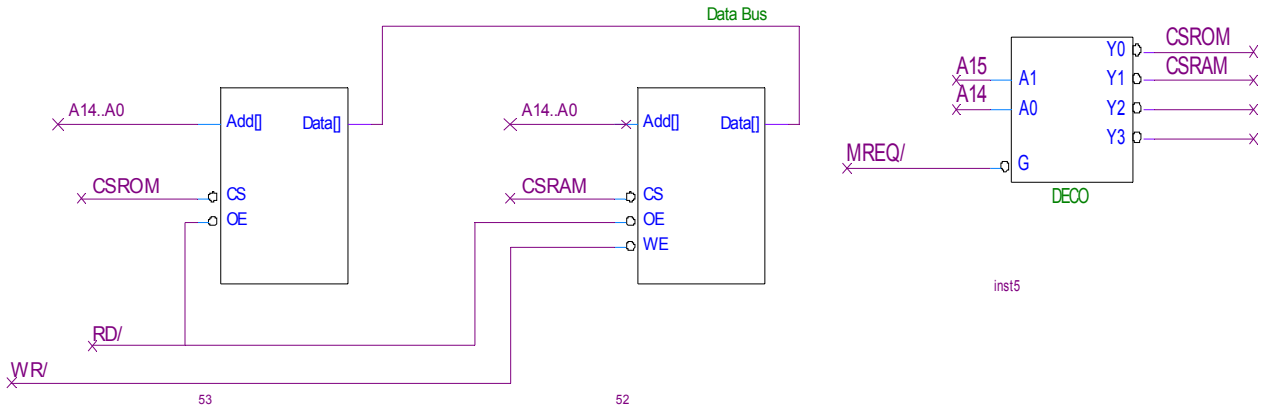


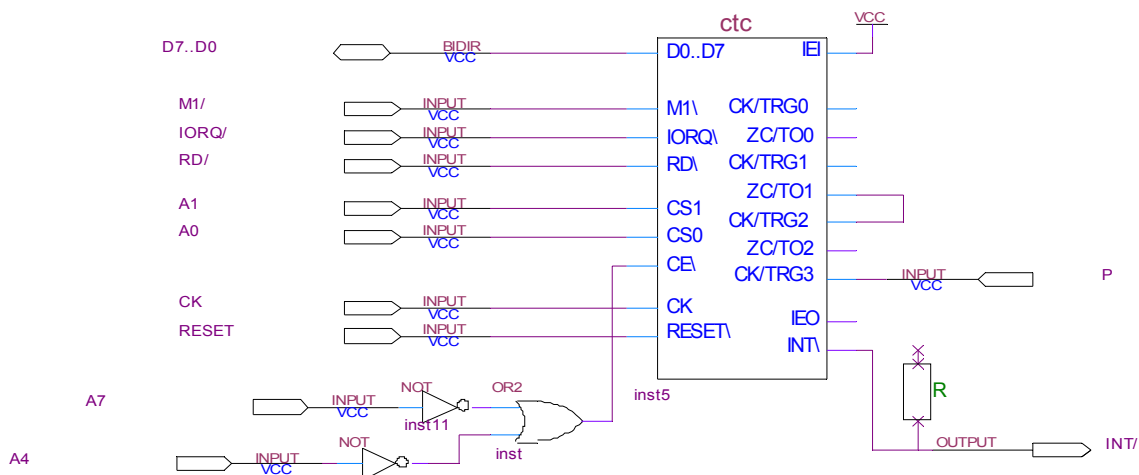
SOLUCIONES

PROBLEMA 1

a) Hardware. Memorias

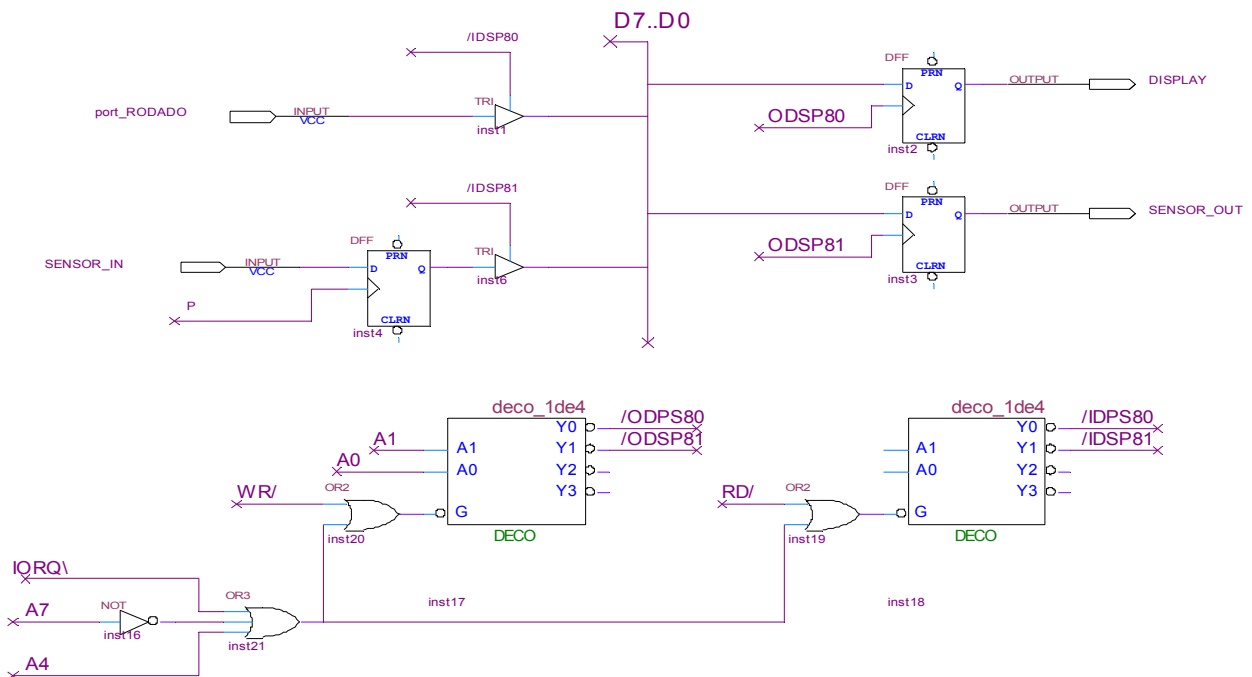


CTC:



PUERTOS: salidas: port_c[8], port_d[8], port_u[8]

entrada: port_RODADO[8]



SOLUCIONES

b) Rutinas de atencion a interrupcion

```

    org 4000h
rutint_pulso:
    ei
    push af
    push bc
    ld a, (sensor_ant)
    ld b, a          ; anterior en b
    in a, (sensor_in)
    ld (sensor_ant), a ; almaceno
    add a, b         ; sumo
    slasra a      ; div 2
    out (sensor_out), a
    ld a, (cuenta_pulso)
    inc a
    ld (cuenta_pulso), a
    pop bc
    pop af
    reti

```

```

rutint_seg:
    ei
    push af
; llamar convert(vueltas)
; cuenta_pulso = 0
    ld a, (cuenta_pulso)
    ld b, a
    xor a
    ld (cuenta_pulso), a
    ld a, b
    call pulsos2kph
; actualizar display
    ld a, (kph)
    out (display), a
    pop af
    reti

```

```

c) Inicializacion y variables
ctc_0      equ 90h
ctc_1      equ ctc_0+1
ctc_2      equ ctc_0+2
ctc_3      equ ctc_0+3
vector_ctc equ 0
; canal 3: EI, counter, rising,
; sige cte (11x1x101)
prog_word3 equ 11010101B
cte3       equ 1
; canal 1: DI, timer, pre=16,
; auto. trg, sige cte (000x1101)
prog_word1 equ 00000±101B
cte1       equ 250
; canal 2: EI, counter,
; sige cte (11xxx101)
prog_word2 equ 11000101B
cte2       equ 250
display    equ 80h
sensor_out equ 81h
p_RODADO   equ 80h
sensor_in  equ 81h

```

```

    org 8000H
cuenta_pulso: DB
kph:          DB
RODADO:       DB
sensor_ant:   DB

```

```

    org 2000h
; tabla interrupciones en ROM
tabla_vec:
    DW      rutint_pulso ; canal 0
    DW
    DW      rutint_seg   ; canal 2
    DW
; Inicializacion
; stack
; display, sensor_out
; variables: cuenta_pulso, kph
; y sensor_ant en cero
; RODADO desde port_RODADO
; modo 2 interrupciones
; registro I
; tabla (en rom)
; vector CTC
; canal CTC pulso (counter y cte=1)
; canales CTC para interrupcion periódica 1
seg

```

```

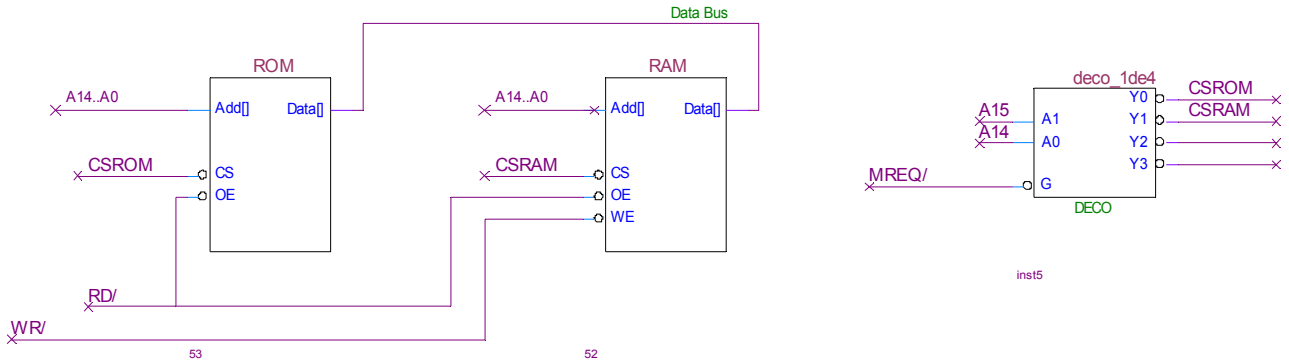
    org 0
    jp inicio

    org 100
inicio:
; stack pointer
    ld sp, 0000h
    ld a, 0
; inicializo puertos
    out (display), a
    out (sensor_out), a
; inicializo variables
    ld (cuenta_pulso), a
    ld (kph), a
    ld (sensor_ant), a
; leo diametro ruedas
    in a, (p_RODADO)
    ld (RODADO), a
; modo 2 interrupciones
    im 2
; registro I (tabla en rom)
    ld a, tabla_vec / 256
    ld i, a
; vector CTC
    ld a, vector_ctc
    out (ctc_0), a
; canal CTC pulso (counter y cte=1)
; P con menor prioridad para evitar que
; se modifique la cuenta de pulsos dentro
; de la interrupción periódica (1 seg)
    ld a, prog_ctc3
    out (ctc_3), a
    ld a, cte3
    out (ctc_3), a
; canales CTC para int. 1 seg
    ld a, prog_ctc1
    out (ctc_1), a
    ld a, cte1
    out (ctc_1), a
    ld a, prog_ctc2
    out (ctc_2), a
    ld a, cte2
    out (ctc_2), a
    ei
    jp ppal

```

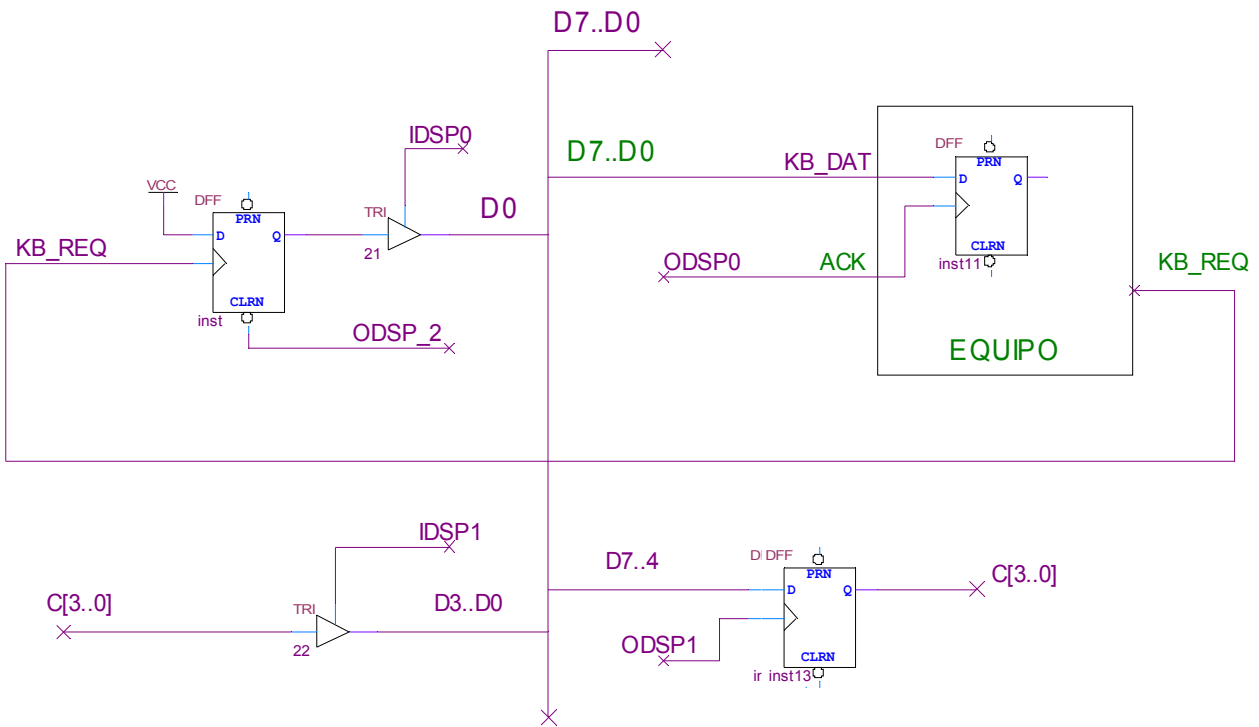
Solución problema 2

a) Hardware
Memoria



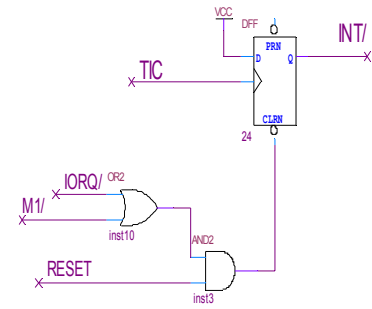
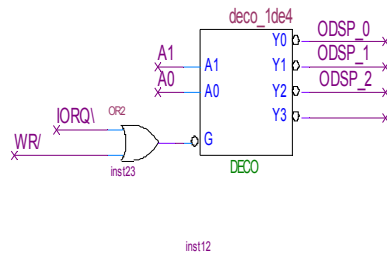
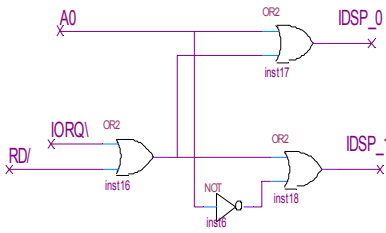
Puertos

El equipo solo espera que KB_DAT esté válido en el flanco de subida de ACK por lo que debe memorizarlo internamente y no hace falta poner el registro del puerto de salida del sistema Z80.



Decodificación y solicitud interrupción

SOLUCIONES



b)

```

; scan()
; for i=0 to 3
;     sel fila i
;     leo cols
;     si cols <> 1111b entonces
;         retorno(<fila i><cols>)
;     fin si
; fin for
; retorno( 0FFh no hay tecla presionada)
    
```

```

; kbd_val = scan()
; restaurar regs
; (borrar peticion), ei, ret
    
```

```

filas equ 01h
cols equ 01h
    
```

```

org 38h
rutint: push af
        push bc
        call scan
        ld (KBD_VAL), a
    
```

```

ORG 1000h
scan:
    
```

```

        push bc
        ld c, 01110111b
for:    ld a, c           ; fila en c
        out (filas), a
        in (cols), a
        and 0Fh
        cp 0Fh
        jr z, finsi
        ld b, a         ; col en b
        ld a, 0F0h
        and a, c       ; nibble alto = fila
        or a, b        ; nibble bajo = col
        jr mevoy
finsi:  rrc c          ; rotación 8 bits y lsb al Cy
        jr c, for
        ; si cy = 0 fue la cuarta
        ; fila sin tecla apretada
        ld a, 0FFh
mevoy:  pop bc
        ret
    
```

```

d)
kb_req equ 0
kb_data equ 0
borro_ff equ 2
    
```

```

org 0000h
ld sp, 0000h
im 1
ld a, 0FFh
ld (KBD_VAL), a
out (borro_ff), a
ei
loop:  in a, (kb_status)
        bit 0, a
        jr z, loop
        out (borro_ff), a
        ld a, (KBD_VAL)
        out (kb_data), a
        jp loop
    
```

```

c)
; rutint
; preservar registros
    
```

```

e)
ORG 8000h
KBD_VAL: DB
    
```