INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INTRODUCCIÓN A LOS MICROCONTOLADORES

HERNÁNDEZ GARCÍA EVA NATALIA

RUIZ ASTORGA ALBERTO ALEJANDRO

"EXAMEN SEGUNDO PARCIAL - RELOJ"

GRUPO: 3CM10

PROFESOR: PÉREZ PÉREZ JOSÉ JUAN

1 | DIAGRAMAS

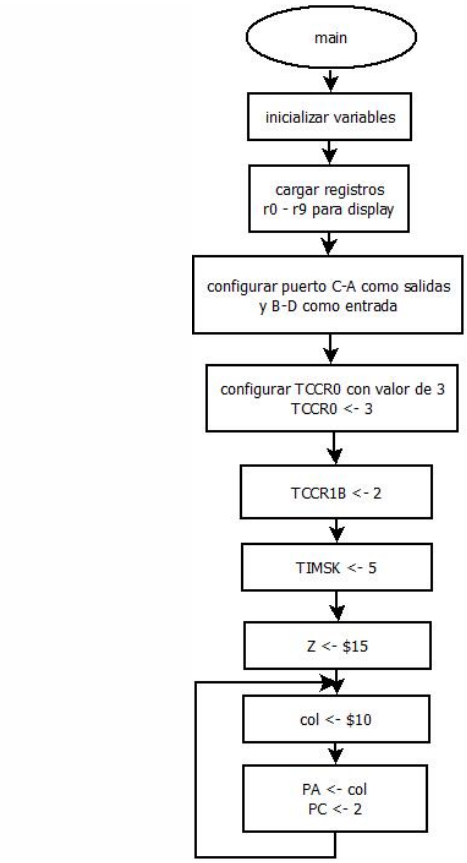


Ilustración 1. Subrutina Main

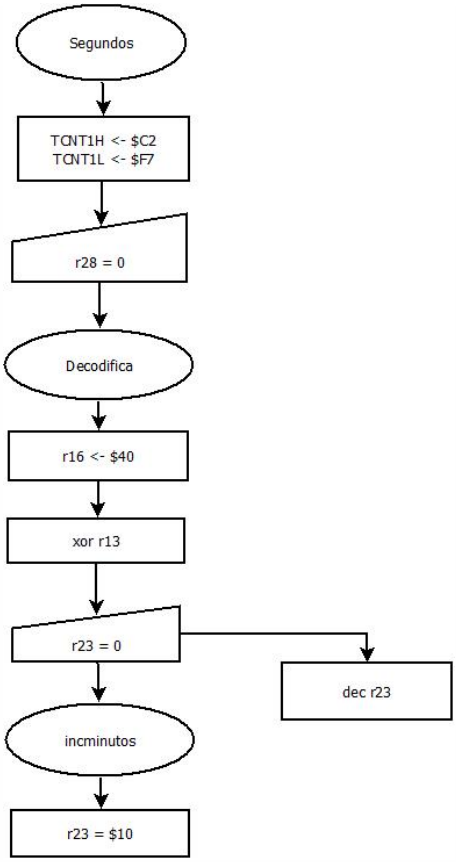


Ilustración 2. Subrutina Segundos.

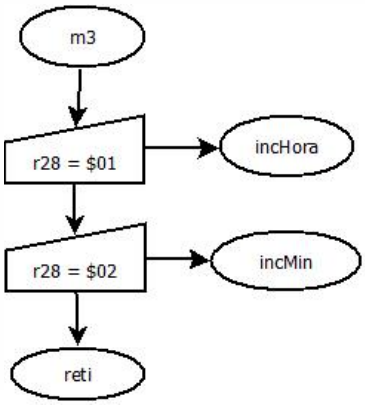


Ilustración 3. Interrupción int2

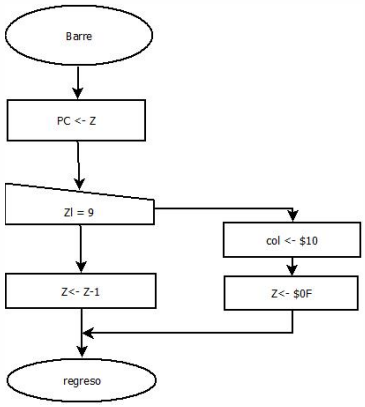


Ilustración 4. Subrutina Barre.

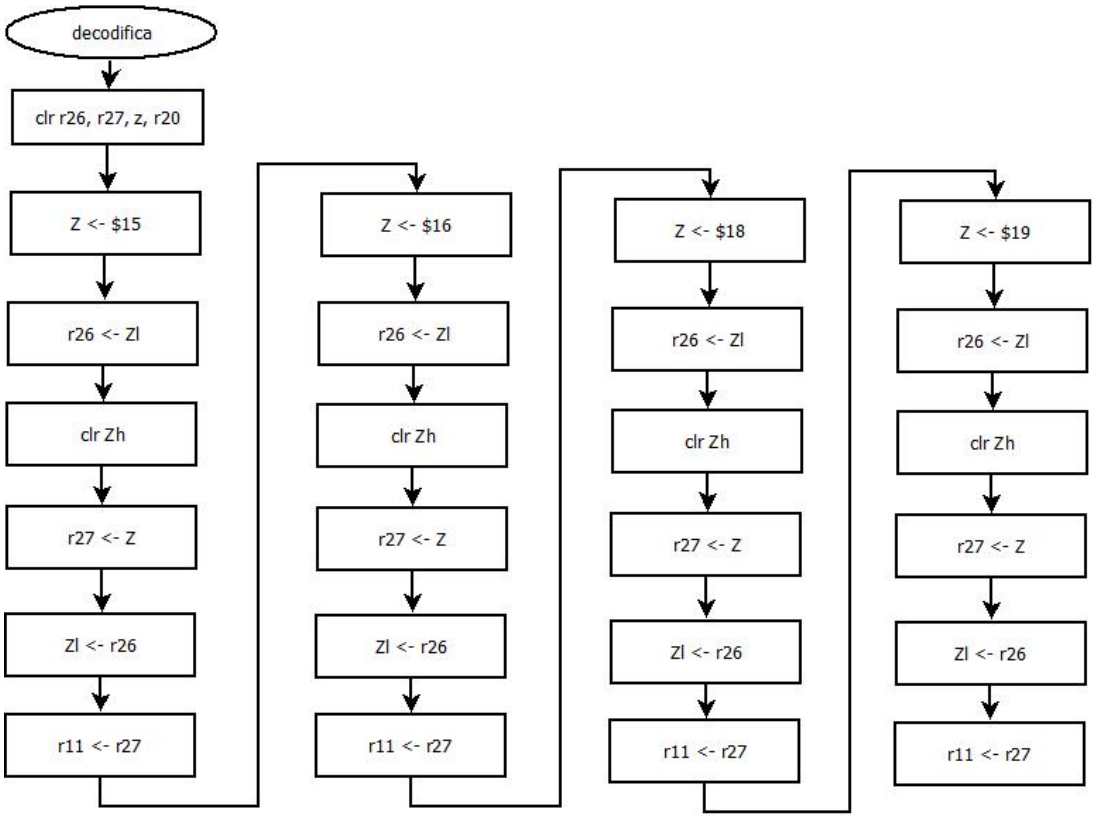


Ilustración 5. Subrutina Decodifica

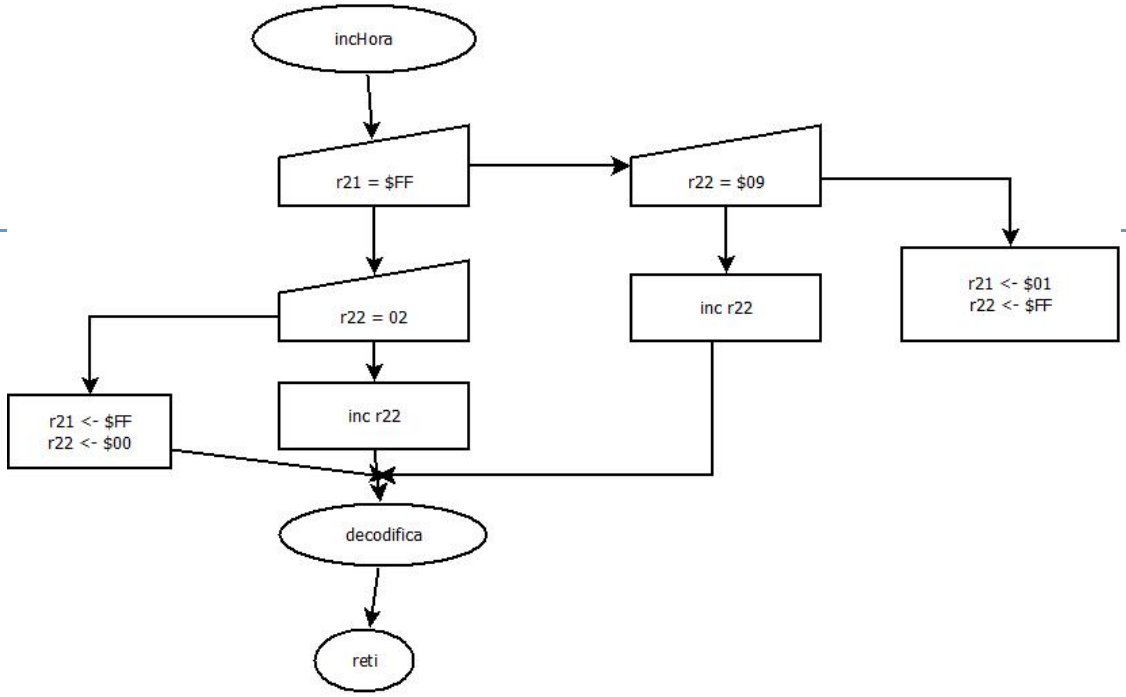


Ilustración 6. Subrutina IncHora

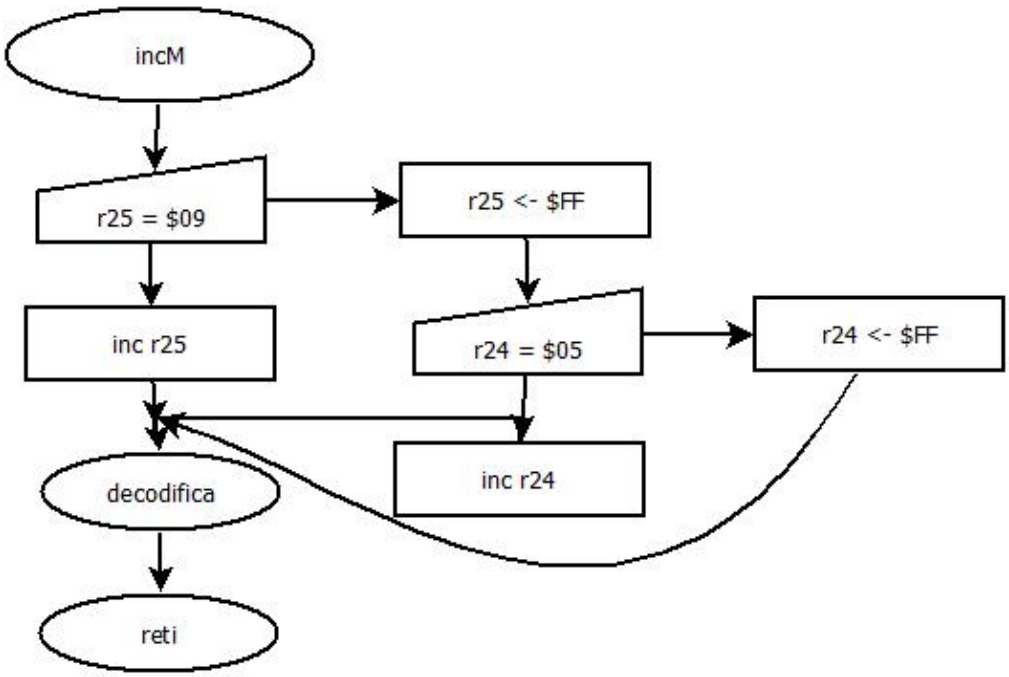


Ilustración 7. Subrutina IncM

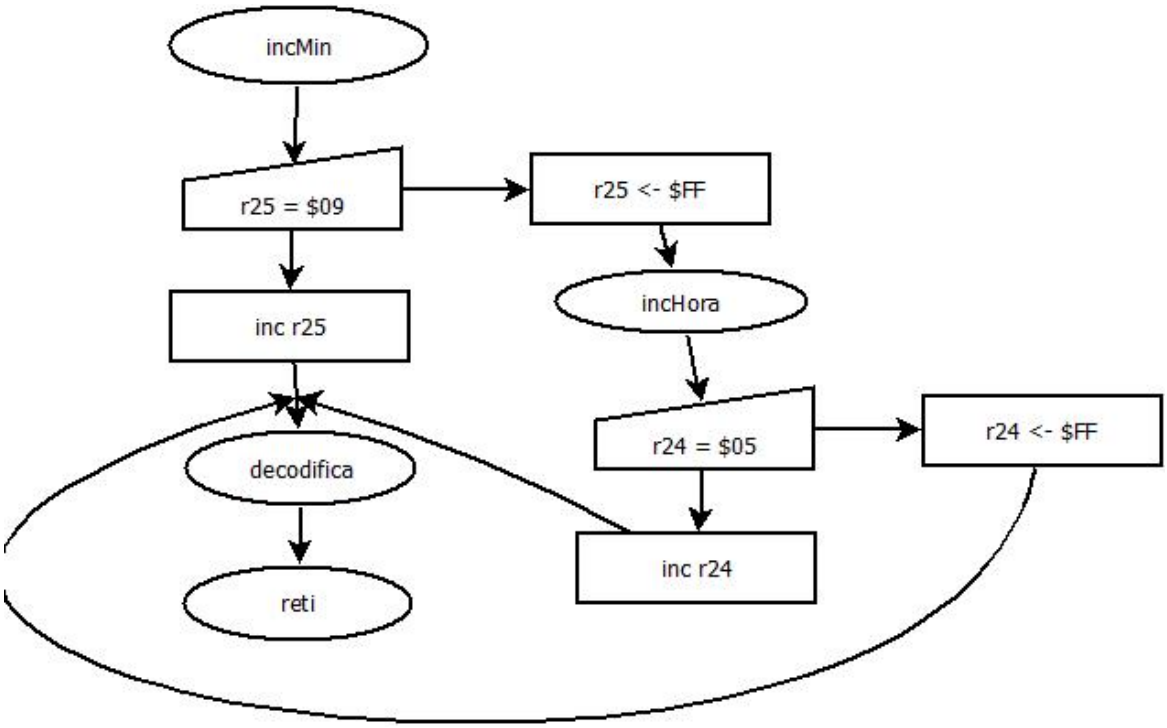


Ilustración 8. Subrutina IncMin

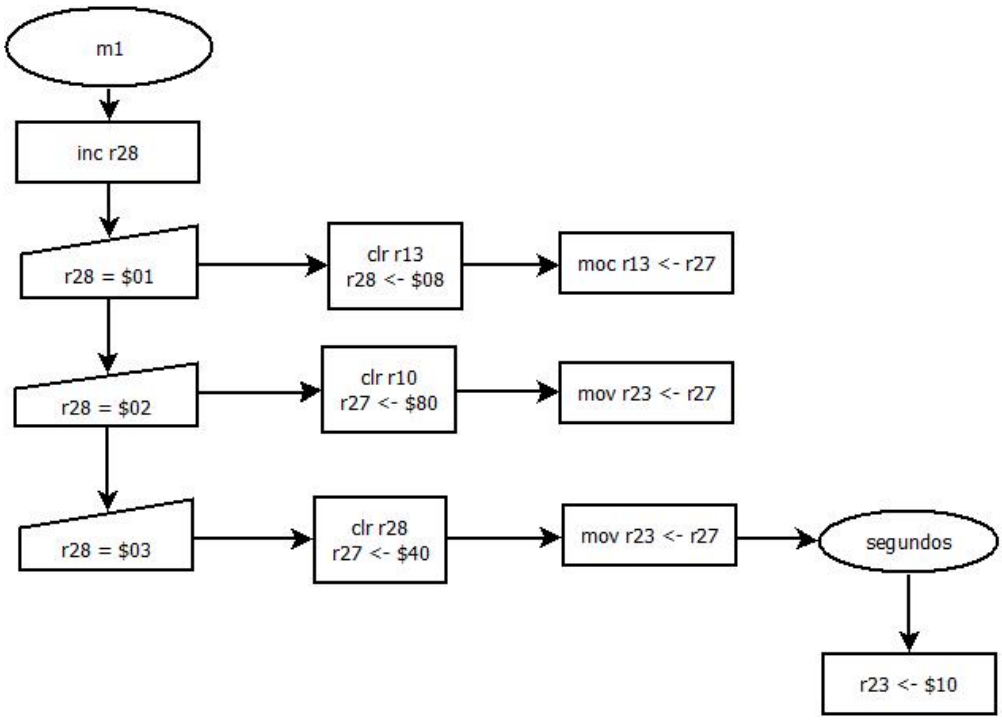


Ilustración 9. Subrutina interrupción int0

2 | CÓDIGO

```
.include"m8535def.inc"
.def d = r16
.def aux = r20
ldi aux, 0
.def col = r17
.def aux2 = r18
.def aux3 = r19
```

Declaración de variables y librería

```
rjmp main
rjmp m1
rjmp m2
.org $008
rjmp segundos
rjmp barre
.org $12
rjmp m3
```

Saltos a interrupciones

```
main:
    ldi d,$01
    mov r21,d
    ldi d,$02
    mov r22,d
    ldi d,$00
    mov r24,d
    ldi d,$00
    mov r25,d
```

Llenado de registros r21- r25 , utilizados para hacer los incrementos según corresponda.

```
    ldi d,$67
    mov r9,d

    ldi d,$7F
    mov r8,d

    ldi d,$07
    mov r7,d

    ldi d,$7D
    mov r6,d

    ldi d,$6D
    mov r5,d

    ldi d,$66
    mov r4,d

    ldi d,$4F
    mov r3,d

    ldi d,$5B
    mov r2,d

    ldi d,$06
    mov r1,d

    ldi d,$3F
    mov r0,d
```

Llenado de registros r0 – r9 con la decodificación del display.

```
/*  r15,r14,r13,r12,r11,r10 registros para usar en el display */
    ldi d, low(ramend)
    out spl,d
    ldi d, high(ramend)
    out sph,d
    ser d
    out ddrc,d
    out ddra,d
    out portb,d
    out portd,d
    ldi d,2
    out mcucr,d
    ldi d,$e0
    out gicr,d
    sei
```

Se declaran como entradas el puerto B y D, y como salida el puerto C y A. Se inicializan los registros MCUCR y GICR para las interrupciones

```
    ldi d, $40
    mov r13,d

    clr r28
    clr r10
    clr r27
    ldi r23, 10
```

Se llena el registro r13 con el decodificación para el guion y mostrarse en el display.

```
inicio:
    ldi aux,3
    out TCCR0, aux
    ldi aux,2
    out TCCR1B, aux
    ldi aux,5
    out TIMSK, aux
    sei
```

Se inicializa el timer

```
    clr zh
    ldi zl,$0F
    ldi col,$10

uno:
    out portc,col
    ld aux,z
    out porta,aux
rjmp uno

barre:
    ldi aux2,192
    out TCNT0,aux2
    out portc,zh
    lsr col

    cpi zl, $09
    ;breq dos
    brcs dos
    dec zl
    ldi aux3,0

reti

dos:
    ldi col,$10
    ldi zl, $0F
    reti

reti
```

Llenado de los registros utilizados para cada uno de los display. Se usa la variable z para irlos recorriendo. En este caso zl se inicializa en 15 ya que los registros van de r10 – r15. Se utiliza otro registro para hacer el barrido el cual se inicializa contemplando solo 5 displays.

```
segundos:

    ldi d, $C2
    out TCNT1H, d
    ldi d, $F7
    out TCNT1L, d

        cpi r28,$00; Compare :
        brbc 1,esig4; Branch :

            rcall decodifica

            ldi d, $40
            eor r13, d


            cpi r23,$00; Compa
            brbc 1,enop; Branc


                rcall incMin
                ldi r23 , 10

            enop: dec r23

    esig4:
reti
```

Se inicializa el timer; r28 se utiliza como bandera para simular la interrupción del tiempo en el reloj. Se utiliza una xor para hacer el parpadeo del guion; r23 es utilizado como contador.

```
decodifica:

        clr r26
        clr r27
        clr zh
        clr zl
        clr r10

    //Para R11
        ldi zl,$15
        mov r26,zl

        clr zh
        ld r27,z

        clr zh
        mov zl, r27

        clr zh
        ld r27, z

        mov zl,r26

        mov r11,r27

        clr r26
        clr r27
        clr zh
        clr zl
        clr r10

    //Para R12

        ldi zl,$16
        mov r26,zl

        clr zh
        ld r27,z

        clr zh
        mov zl, r27

        clr zh
        ld r27, z

        mov zl,r26

        mov r12,r27

        clr r26
        clr r27
        clr zh
        clr zl
        clr r10

    //Para R14

        ldi zl,$18
        mov r26,zl

        clr zh
        ld r27,z

        clr zh
        mov zl, r27

        clr zh
        ld r27, z

        mov zl,r26

        mov r14,r27

        clr r26
        clr r27
        clr zh
        clr zl
        clr r10

    //Para R15

        ldi zl,$19
        mov r26,zl

        clr zh
        ld r27,z

        clr zh
        mov zl, r27

        clr zh
        ld r27, z

        mov zl,r26

        mov r15,r27

        clr r26
        clr r27
        clr zh
        clr zl
        clr r10

reti
```

Accede a los registros r0 – r9, según sea el caso a través de la variable zl y muestra en el display al restarle 10 a dicha variable, ya que los registros que se muestran en el display son de r10 -15 y los usados para incrementar son de r20 – r25. Se hacen limpieza de registros.

```
m1:

inc r28

        cpi r28,$01
        brbc 1,enopM1_1
            clr r13
            ldi r27,$08
            mov r13,r27

        enopM1_1:
            cpi r28,$02;
            brbc 1,enopM1_2
                clr r10
                ldi r27,$80
                mov r13,r27

        enopM1_2:
            cpi r28,$03;
            brbc 1,enopM1;
                clr r28
                ldi r27, $40
                mov r13, r27
                rcall segundos
                ldi r23, 10
        enopM1: //rjmp segundos

reti
```

Se usa el r28 como bandera para saber si se trata de un incremento en horas, en minutos o bien reactivar el tiempo del reloj.

```
m2:

reti

m3:
        cpi r28,$01;
        brne enopM3_1
            rcall incHora

    enopM3_1:
        cpi r28,$02;
        brne enopM3
            rcall incM

    enopM3: //rjmp segundos

reti

incM:
        cpi r25,$09;
        brbc 1,enopM;
            ldi r25, $FF

            cpi r24,$05;
            brbc 1,enopMD;
                ldi r24, $FF

            enopMD:
                inc r24

        enopM:
            inc r25

rcall decodifica
ret
```

Dependiendo del estado del registro r28, hace el incremento ya sea en minutos o en horas.

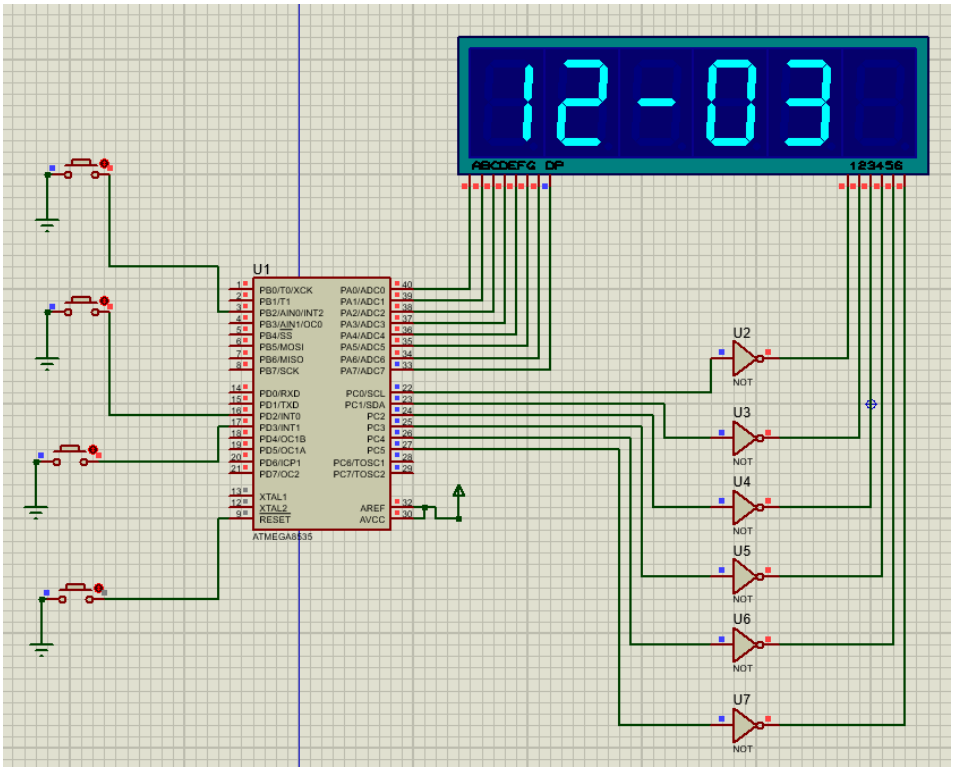
diferencia de incMin, este no hace la llamada a la subrutina inHora ya que solo nos interesa incrementar de 00 al 59 sin modificar horas.

```
incMin:
        cpi r25,$09;
        brbc 1,enopMin;
            ldi r25, $FF

            cpi r24,$05;
            brbc 1,enopMinD
                ldi r24, $FF
                rcall incHora

            enopMinD:
                inc r24

        enopMin:
            inc r25

rcall decodifica
ret

incHora:
            cpi r21,$FF;
            brbc 1,enopHorD

                cpi r22,$09;
                brbc 1,enopHor12_09;
                    ldi r21, $01
                    ldi r22, $FF

            enopHorD:
                cpi r22,$02;
                brbc 1,enopHor12_09;
                    ldi r21, $FF
                    ldi r22, $00

            enopHor12_09:
                inc r22
rcall decodifica
ret
```

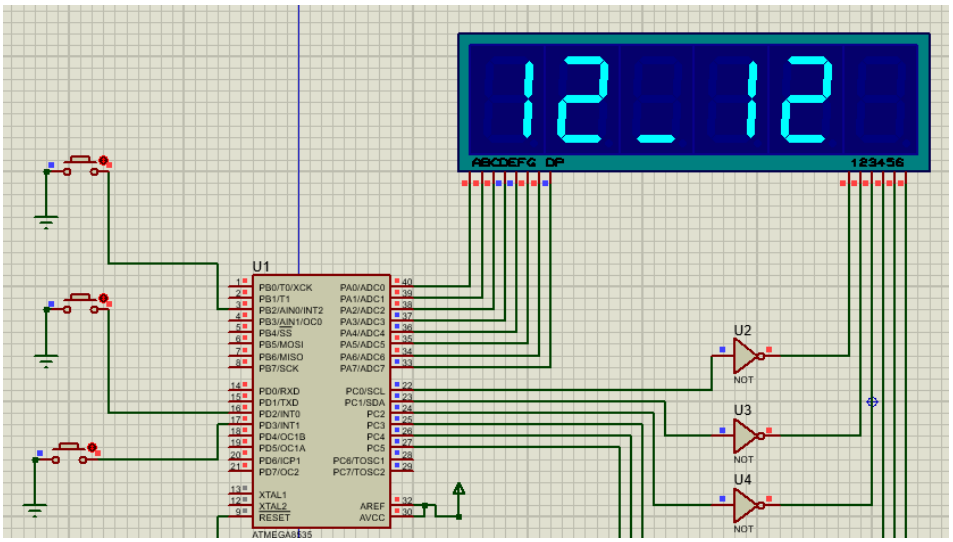
Incrementa el minuto haciendo las validaciones correspondientes por cada cifra, incrementa las decenas hasta llegar al 5, y las unidades hasta llegar a 9. Al incrementar las decenas va haciendo la llamada a la subrutina incHora.

Incrementa la hora haciendo las validaciones correspondientes por cada cifra, si se trata de 1 en las decenas para 10,11 y 12 , y si se trata de -1 para el rango de 1-9. En el caso de unidades va haciendo el incremento normal hasta llegar a 10.

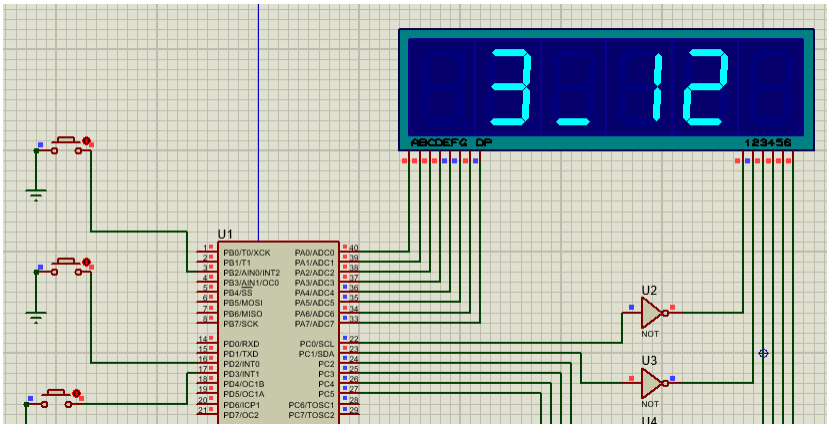
3 | SIMULACIÓN



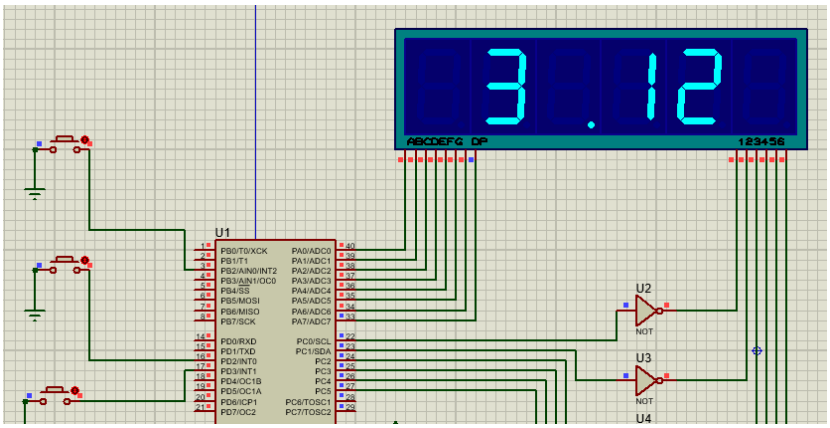
Pantalla 1. Ejecución inicial, el reloj corre a partir de 12:00



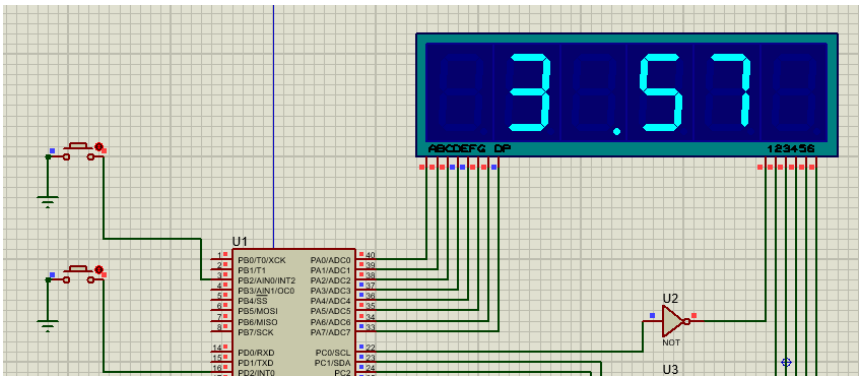
Pantalla 2. Se activa por primera vez la interrupción 3, cambia el guion por guion bajo e indica que se pueden modificar las horas.



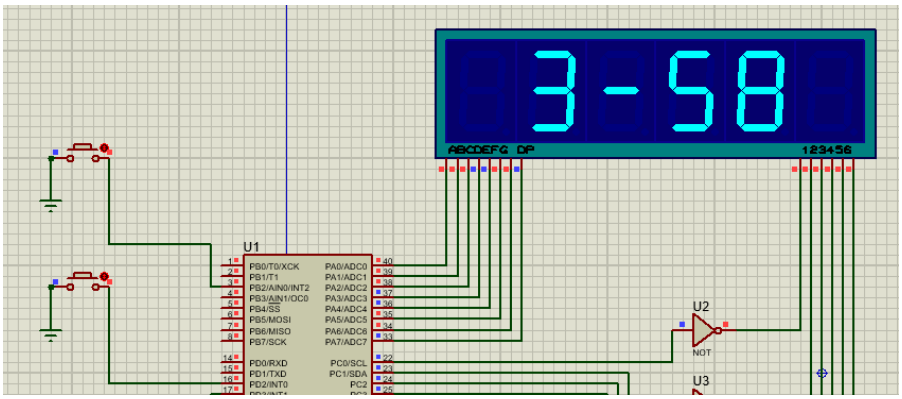
Pantalla 3. Se activa la interrupción 2 n veces, siendo cada una un incremento.
En este caso al haber activado por primera vez la interrupción 3, solo se pueden incrementar las horas.



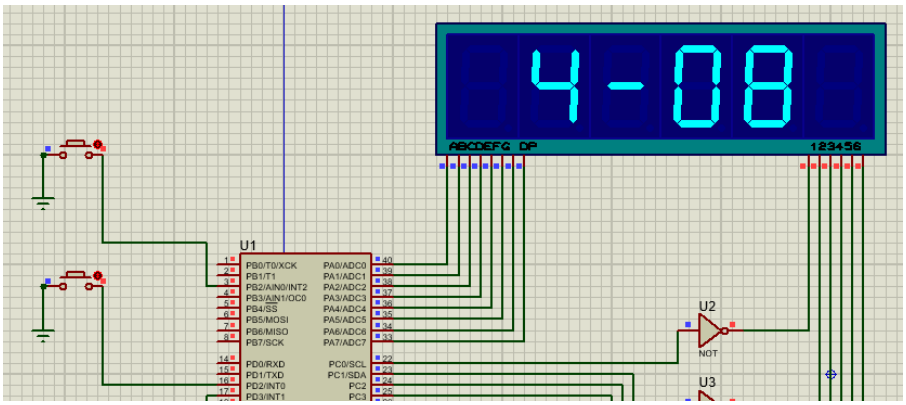
Pantalla 4. Se activa por segunda vez la interrupción 3, cambia el guion bajo por punto e indica que se pueden modificar los minutos.



Pantalla 5. Se activa la interrupción 2 n veces, siendo cada una un incremento.
En este caso al haber activado por segunda vez la interrupción 3, solo se pueden incrementar los minutos



Pantalla 6. Se activa por tercera vez la interrupción 3, esto reactiva el reloj.



Pantalla 7. Ejecución del reloj después de los cambios realizados.