Universidad autónoma de baja california

Ingeniero en computación

microcontroladores

practica 3

Maestro: José Isabel Garcia Rocha

Erik Garcia Chávez 01275863

Lunes 2 de septiembre del 2024

-teoria de puertos de I/O del atmega 2560

todos los puertos de AVR tienen una funcionalidad de lectura--modificación-escritura. cuando se utilizan como puertos de I/O digitales generales. por lo que la dirección de un pin de puerto se puede cambiar sin cambiar involuntariamente la dirección de ningún otro pin con la ayuda de las instrucciones SBI y CBI. todos los pins de puerto tiene resistencias pull-up seleccionables individualmente con una resistencia invariante de voltaje de suministro.

para los puertos de entrada y salida se tienen 3 registros PORTx-DDRX-PINx donde la x representaría la letra del puerto.

donde:

PORTx -> es el registro de dirección de datos

DDRx -> registro de direcciones

PINx -> pines de entrada del puerto

se accede a los bits DDxn en la dirección de E/S DDRx, a los bits PORTxn en la dirección de E/S PORTx y a los bits PINxn en la dirección de E/S PINx.

el bit DDxn (representa el puerto y el pin), en el registro DDRx selecciona la dirección del pin. si DDRx se escribe en 1 lógico, Pxn se configura como pin de salida en caso contrario se configura como pin de entrada.

si PORTxn se escribe con 1 lógico, cuando el pin está configurado como pin de entrada, se activa las resistencias pull-up. para apagar el pull-up PORTxn se escribe con un cero lógico o el pin se configura como pin de salida.

-formulación de la ecuación:

La ecuación la hice conforme iba haciendo el código, iba ajustando ambos mientras se desarrollaba la rutina de “miRetraso”, ya que primero necesitaria ver una esqueleto, para saber cómo estructurar la formula, después que realice un esqueleto de la rutina, fue cuando empecé a realizar la fórmula con la ayuda de este empecé a ajustar mi código, pero hubieron algunas complicaciones en la forma de entender cómo funcionaba el código, por ahora tengo la fórmula pero se queda corta con los ciclos, pero en el código si da los 1648 ciclos con 103 micro Segundos solicitados, por lo que hay algo en la realización de la fórmula que me estoy saltando.

Donde la formula me da lo siguiente:

```
rcall miRetardo ;4 ciclos
miRetardo:
        clr r25
        clr r26
        clr r24 ;1 ciclo
        ldi r24,5 ;-> 1 -> x
        ldi r26, 7
        ;--------
        ;5 ciclos total
nxt0:
        nop ; ->1x
        nop; 1x
        nop ;1x
        ldi r25, 8 ;1x
        ;-------
        ;5x
nxt1:
        dec r26 ;1zyx
        nop ;1zyx
        nop ;1xyz
        brne nxt1 ; 2xy(z-1)
        ;----------
        ;3xyz + 2xy(z-1)
nxt2:
        ldi r26,7 ;-> 1xy
        nop ; 1xy
        nop ;1xy
```

```
dec r25          ;-> 1xy
brne nxt1 ; 2x(y-1)
dec r24 ;x -> 1x
nop ; 1x
brne nxt0 ;-> 2(x-1)
ret ;5 ciclos
;-------------
;4xy + 2x(y-1) + 2x +2(x-1) +5
```

Lo que nos da al final:

7x + 4xy + 5xyz + 12 donde X= 5, Y= 8 y Z = 7

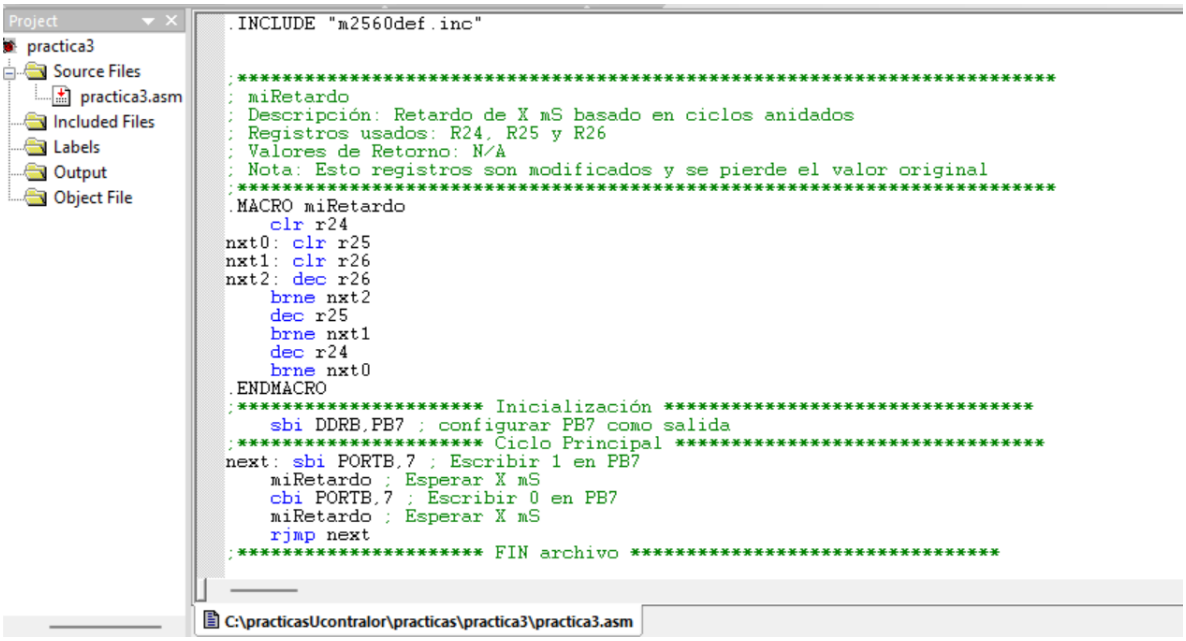
Sustituyendo estos valores nos da:

7(5) + 4(5*8) + 5(5*8*7) + 12 = 1607

Es en donde digo que algo debe estar mal en el desarrollo de la fórmula, pero no logro ver en donde está mi error

Desarrollo de la práctica:

Se tiene la macro del listado 1:



Ahora se carga el .hex a la paca Arduino con Xloader, después el programa se emepzara a ejecutar.

Después de x tiempo el led empieza a prender

con un delay de 103 micro Segndos y la rutina prestarBit.

El inicio de la practica en donde se establece la configuración del puerto.

```
.INCLUDE "m2560def.inc"
;---------------------------------

;garcia chavez erik
;01275863
;laboratorio microControladores
;ciclo 2024-2


;---------------------------------
rcall miRetardo

sbi DDRE, PE1 ;PE1 como salida

sbi PORTE, PE1 ;escribe en 1 PE1

clr r16

sts UCSR0B, r16
```

Rutina principal:

En esta rutina se hace la función principal del programa de la practica 3, se escriben en puertos en este caso se está escribiendo y borrando 1, lo que permite el prender o apagar un led por ejemplo del puerto B, después de llama a la subrutina la cual va a hacer una espera de 103 uSegundos.

```
;principal
next:

    sbi PORTB, 7 ; escribe 1 en PB7
    rcall miRetardo

    cbi PORTB, 7 ;escribe 0 en PB7

    rcall miRetardo

    ;prestar bits

    ldi r18, 0x55
    rcall prestarBits
    ldi r18, 0x41
    rcall prestarBits
    ldi r18,0x42
    rcall prestarBits
    ldi r18, 0x43
    rcall prestarBits
    rjmp next
```

Subrutina miRetardo:

Esta es la función que hace la espera de esperar 103 uSegundos, su única función es perder ese tiempo, se hace uso de 3 ciclos, donde sus variables son R24-> X, R25-> Y y R26-> Z, en este caso estoy usando R24 para mi función de retardo, pero igual se usa en la rutina de prestarBits, por lo que en esa rutina hago uso de R18.

Lo que quiere hacer son ciclos while anidados, pero están dispersos.

Antes de llegar a la subrutina se esta tomando en cuenta los 4 ciclos que tarda con rcall,

Al inicio se van a quedar 5 ciclos los cuales limpian los registros, después asigno a R24 a mi ciclo principal 5,

Next0: es mi ciclo principal, la que engloba las otras, la idea es algo como

```
While(){
        While(){
                While(){
                }
        }
}
```

Aunque en mi código se ve un poco disperso. Primero cargo R24 que es mi X,asi como R26 que es mi Z, r24 si es necesario que sea afuera del ciclo, ya que si se vuelve a cargar entonces el ciclo seria infinito porque constantemente se estaría cargando de nuevo. R26, le cargo su valor, en parte para quemar ciclos, después ingresa a los ciclos, NXT0, es mi ciclo principal, aunque este contiene R25 que representa Y mi segundo ciclo, NXT1. Representa mi ciclo 3 de Z que es el que se va a estar repitiendo la mayor cantidad de veces. Primero NXT1. Se repetirá hasta que R26 sea igual a 0, cuando esto pase iría a NXT2 que es el ciclo 2. En donde se vuelve a cargar el valor de Z=R26, decrementa R25 en 1, si este no es igual a 0, vuelve a NXT1. Donde R26 se vuelve a restar así hasta que R25 sea igual a 0, si este el caso, se resta a R24 y si este no es igual a 0 vuelve a entrar en el ciclo en NXT0. Donde se vuelve a cargar R25=Y, y el mismo proceso hasta que R24 se igual a 0, en donde sale y con un RET=5 ciclos, vuelve a la dirección en PC+1.

```
miRetardo:
    clr r25
    clr r26
    clr r24 ;1 ciclo
    ldi r24,5 ;-> 1 -> x
    ldi r26, 7
    ;--------
    ;5 ciclos total
nxt0:
    nop ; ->1x
    nop; 1x
    nop ;1x
    ldi r25, 8 ;1x
    ;-------
    ;4x

nxt1:
    dec r26 ;1zyx
    nop ;1zyx
    nop ;1xyz
    brne nxt1 ; 2xy(z-1)
    ;----------
    ;3xyz + 2xy(z-1)

nxt2:
    ldi r26,7 ;-> 1xy
```

```
nxt2:
    ldi r26,7 ;-> 1xy
    nop ; 1xy
    nop ;1xy
    dec r25 ;-> 1xy
    brne nxt1 ; 2x(y-1)
    ;---------------------
    ;4xy + 2x(y-1)

    dec r24 ;x -> 1x
    nop ; 1x
    brne nxt0 ;-> 2(x-1)

    ret ;5 ciclos

    ;-------------
    ;2x+2(x-1)+5

    ;--------------------------
    ;7x + 4xy + 5xyz + 12
```

Rutina prestar bits:

Una vez que se haya establecido el puerto E como salida, ahora toca escribir en el pin 1 del puerto E. hago uso de un registro más que va a controlar que solo se recorra una palabra de 8 bits. Hace uso de un ciclo, en el cual primero llama a la rutina que tarda los 103 uSegundos, después hago una rotación hacia la derecha para sacar el bit menos significativo (LSB), este bit se va a carry, con la ayuda de 2 branch, uno comprueba si el bit en carry está en 1 y el otro si está en 0, en cualquiera de los 2 casos que sea brincara a una etiqueta en donde establecemos con SBI o limpiamos con CBI el pin 1 del puerto E, después vuelve a la etiqueta “segundoPaso” en donde hace un retardo de 103 uSegundos, después decrementa el contador de r19, si R19 no está en el bit más significativo MSB, vuelve a hacer el ciclo mostrando en el pin 1 del puerto E los datos de la palabra. Pero si está en el MSB, se pone en 1 el pin 1 espera 103 uSegundos y sed regresa al ciclo principal.

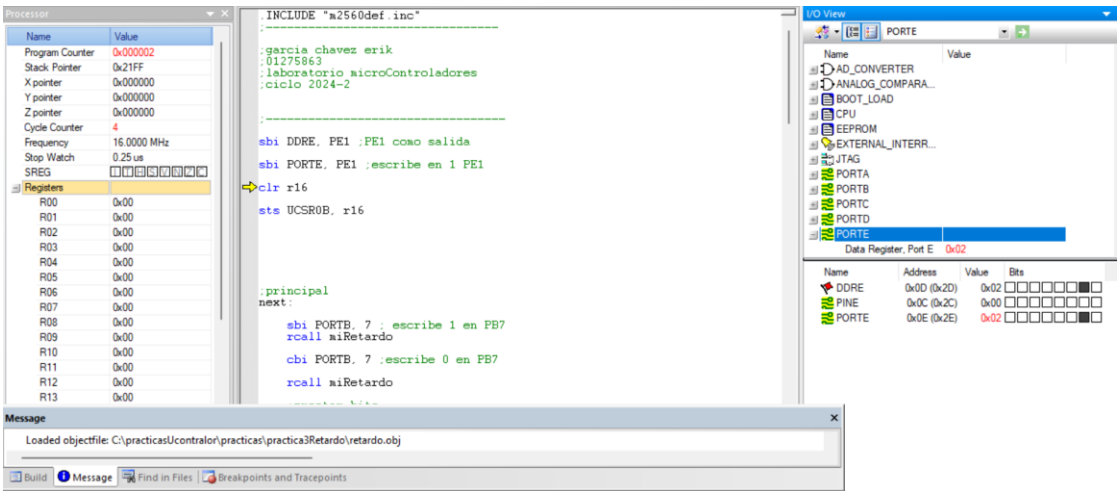
```
prestarBits:

    ;-------------------------------------------------------------------

    ;el registro DDRx se utiliza para configurar un pin como entrada o salida. Seguido,
    ;esta el registro PORTxn que habilita con 5v o deshabilita con 0v. Luego, tenemos el registro PINxn
    ;que sirve para leer el estado un pin configurado previamente como entrada.


    ;DDRX: Escribiendo en cada bit (▮1▮ lógico = Salida | ▮0▮ lógico = Entrada).
    ;PORTX: Escribiendo en cada bit la salida tendrá un nivel de Voltaje (1=5v|0=0v).
    ;PINX: Este registro es de solo Lectura (5v =▮1▮ lógico | 0v =▮0▮ lógico).
    ;-------------------------------------------------------------------


    ldi r19,0x08
    cbi PORTE,1
    ;poner en 0 logico a PE1, el bit 1 del puerto E
ciclo:


    rcall miRetardo ;espera 103 uSegundos

    ;pasar el LSB de r24 a carry, por el MSB entra  un 0
    ror r18

    brcs carrySet ;<- hace un test con la bandera de acarreo si la bandera esta encensida ponemos PE1 en SET

    brcc carryClear
```

```
    brcc carryClear


segundoPaso:

    rcall miRetardo

    dec r19

    cpi r19,0x00

    brne ciclo

    nop
    sbi PORTE,PE1

    rcall miRetardo


    ret


;rutinas de apoyo
carrySet:


    sbi PORTE,PE1

    jmp segundoPaso
```

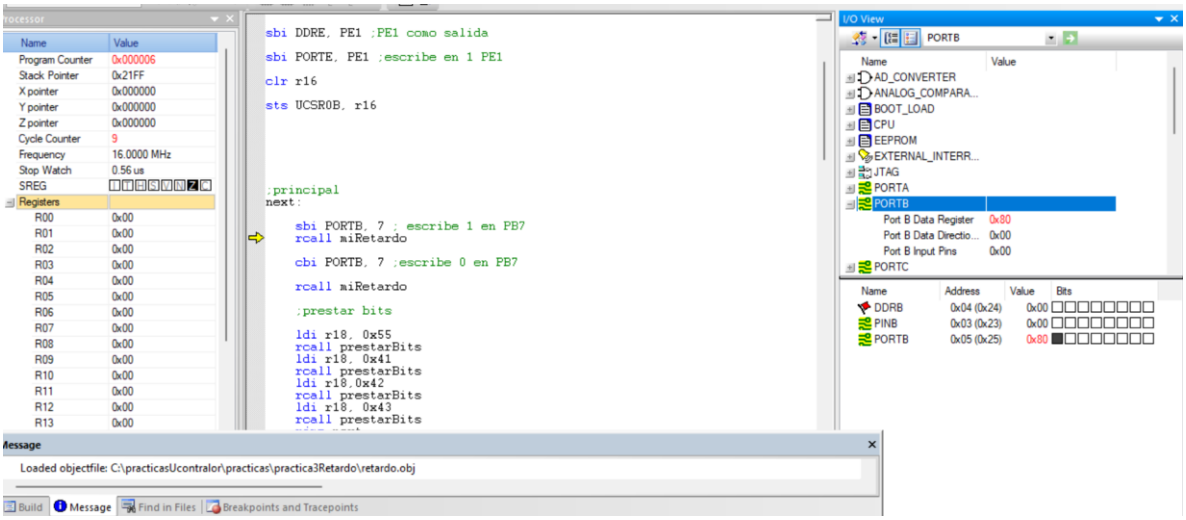
```
    brne ciclo

    nop
    sbi PORTE,PE1

    rcall miRetardo


    ret


;rutinas de apoyo
carrySet:


    sbi PORTE,PE1

    jmp segundoPaso


carryClear:

    cbi PORTE,PE1

    jmp segundoPaso
```

-programa en ejecución:

El programa inicia estableciendo el pin 1 del puerto E como salida:

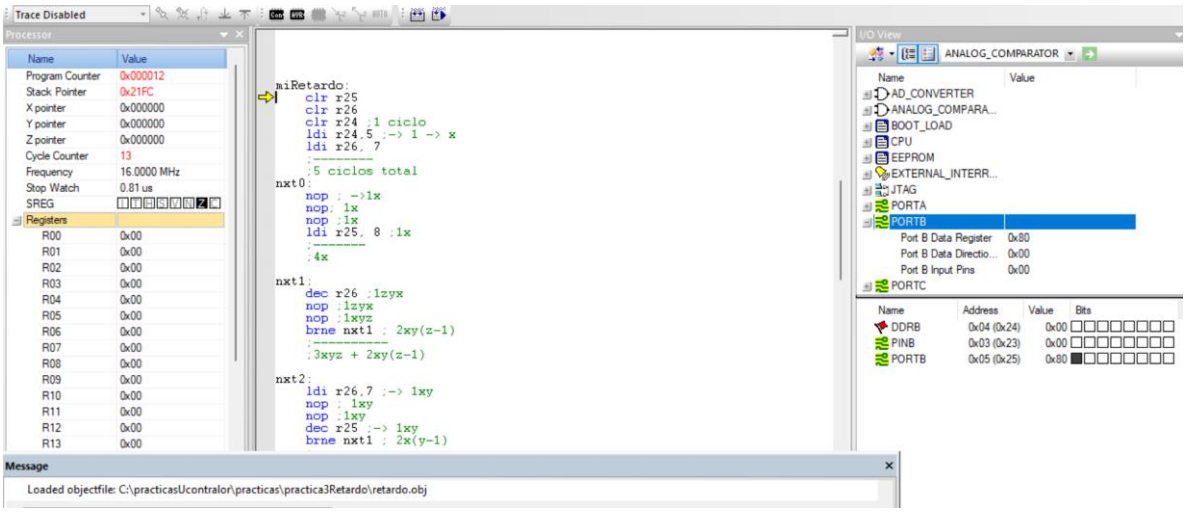


Escribe 1 en el pin 7 del puerto B y después llama a la función mi retardo, que tiene que dardar 103 uSegundos.

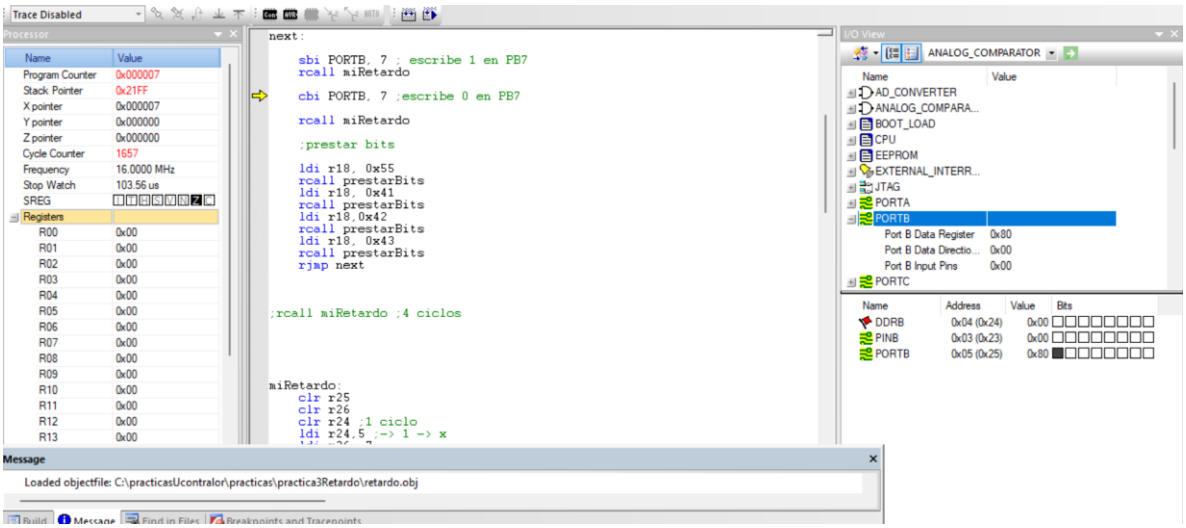


Ahora mismo hay 9 ciclos y 0.56 uS en ejecución, por lo que al ir y volver de mi retardo deberían se ser, 1657 ciclos y 103.56 uSeg.

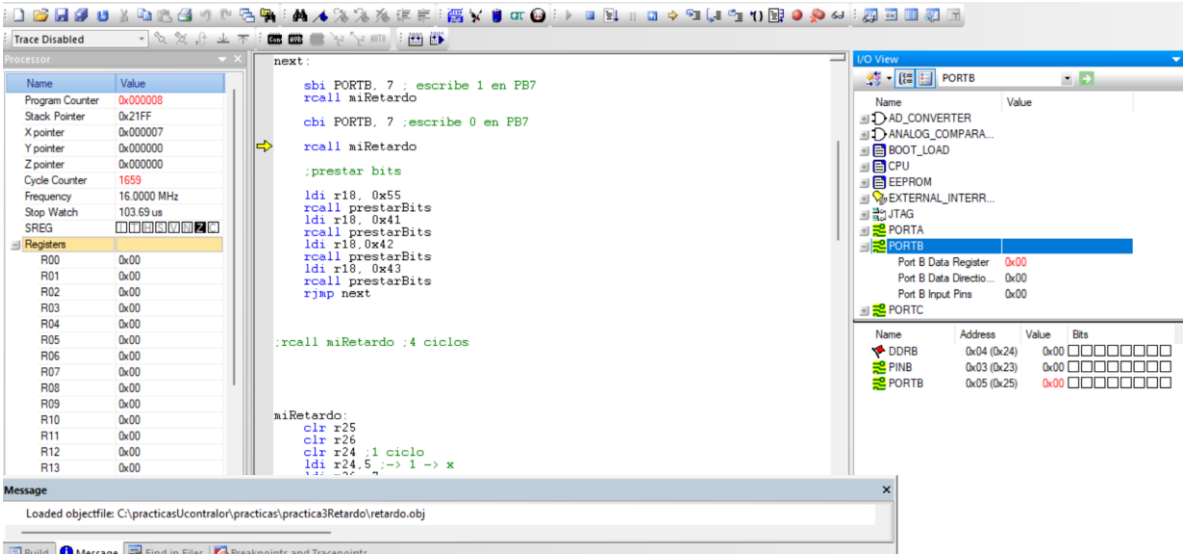
Inicia el retardo de 103 uSegundos



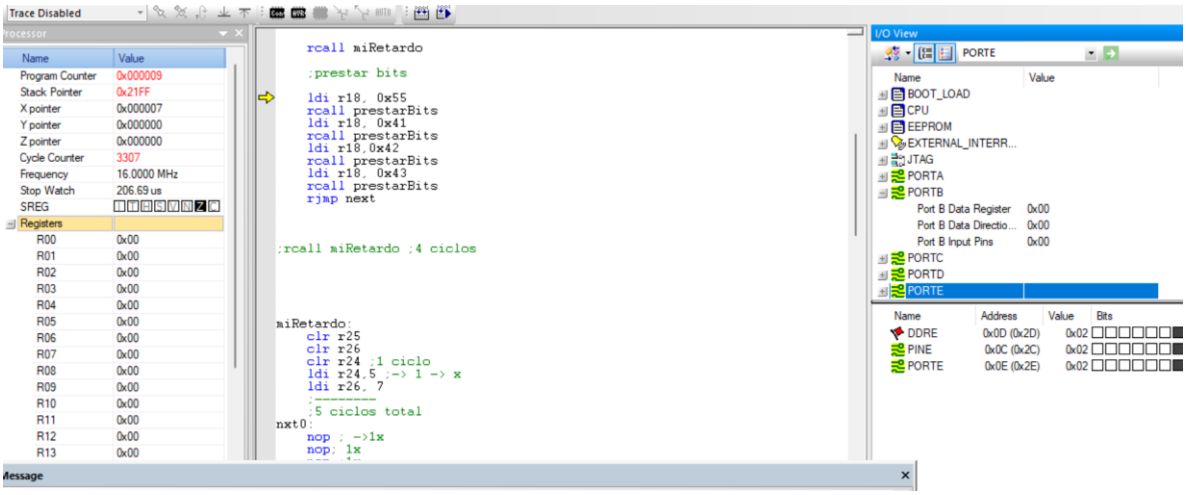
Al volver en efecto tenemos esa cantidad de ciclos y de tiempo:

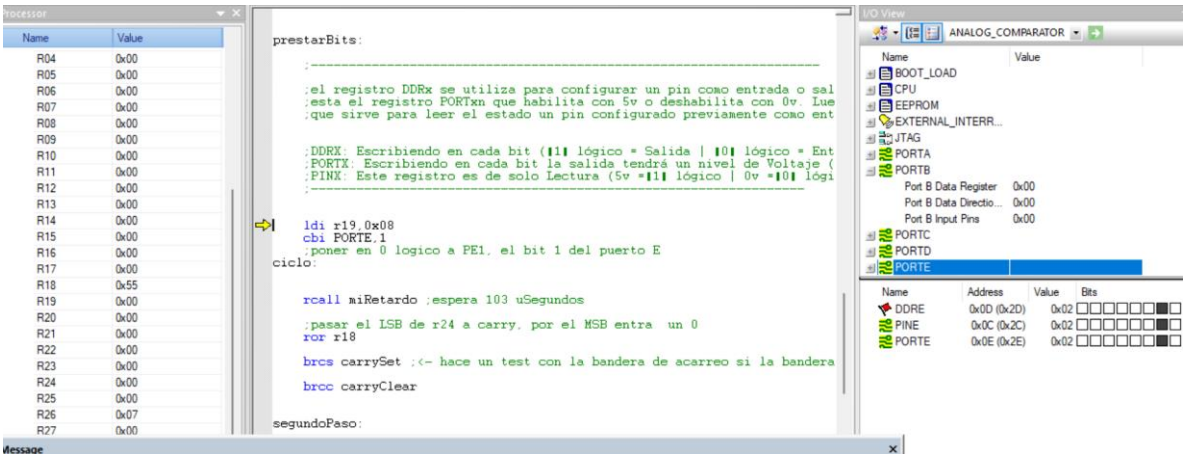


Apaga ese bit y después se vuelve a llamar a mi retardo:



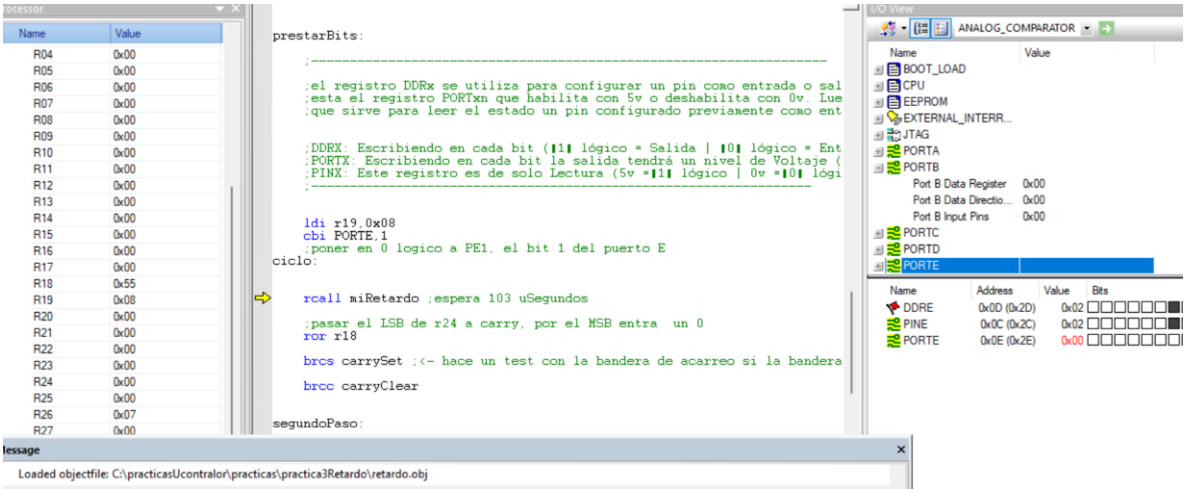
En la practica se menciona que el registro que funcionara como el prestador al pin 1 del puerto E sería el registro R24, pero en mi rutina, miRetardo uso R24 como mi variable X, por lo que no lo quise cambiar y opte por usar otro registro, en este caso uso a R18. Que se le cargaran 4 valores, pero uno a la vez, en donde se mandara a llamar a la rutina “prestarBit”



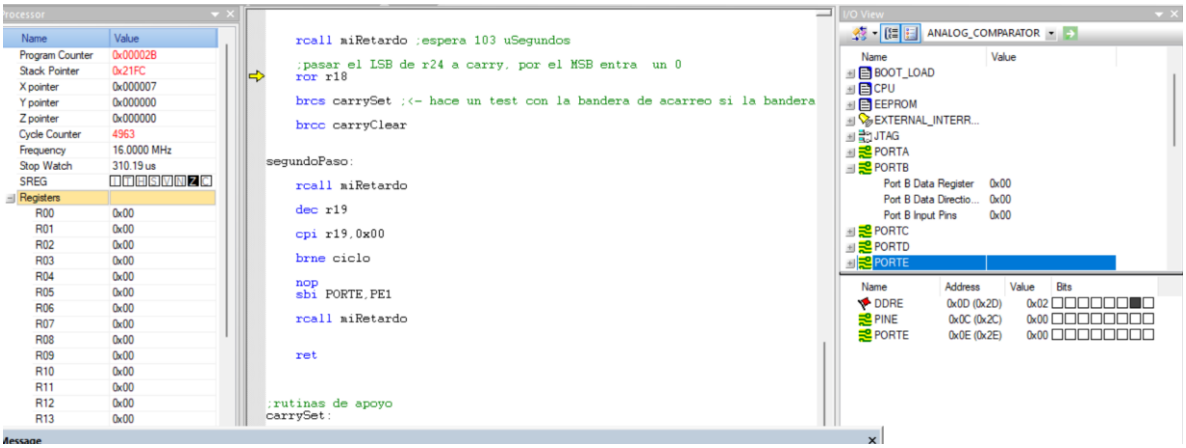


Uso el contador para que saber cunado estoy en el MSB.

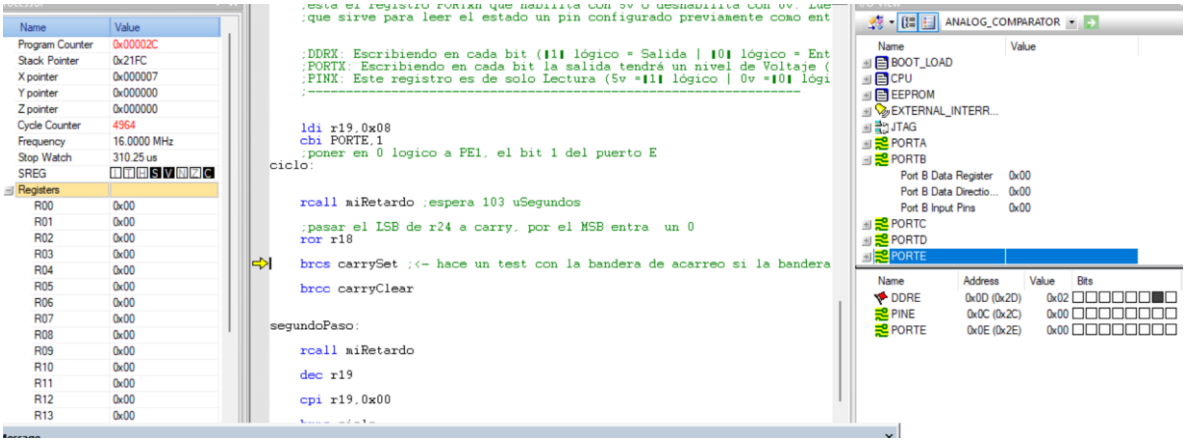
Se apaga el pin 1 del puerto E:



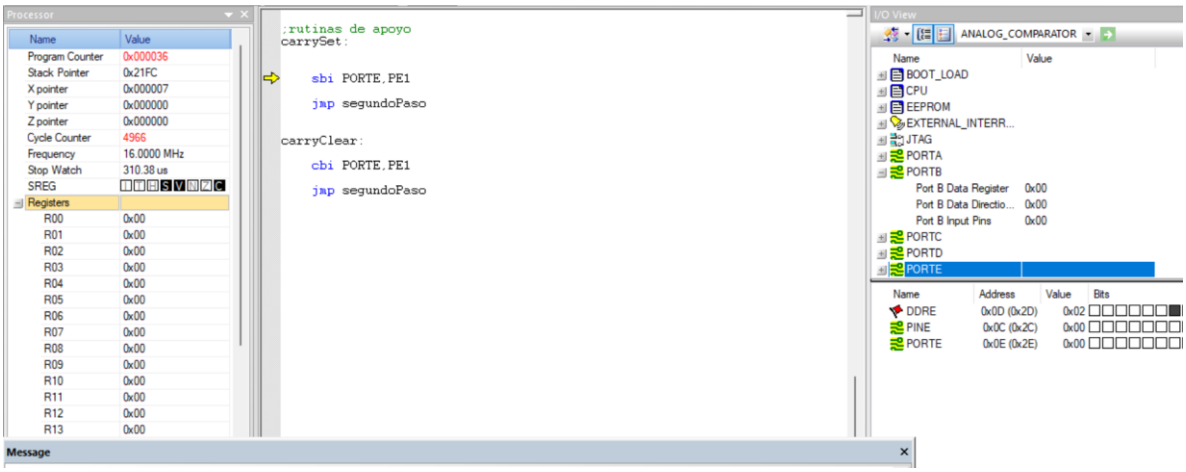
Se llamo a la rutina “miRetardo”:

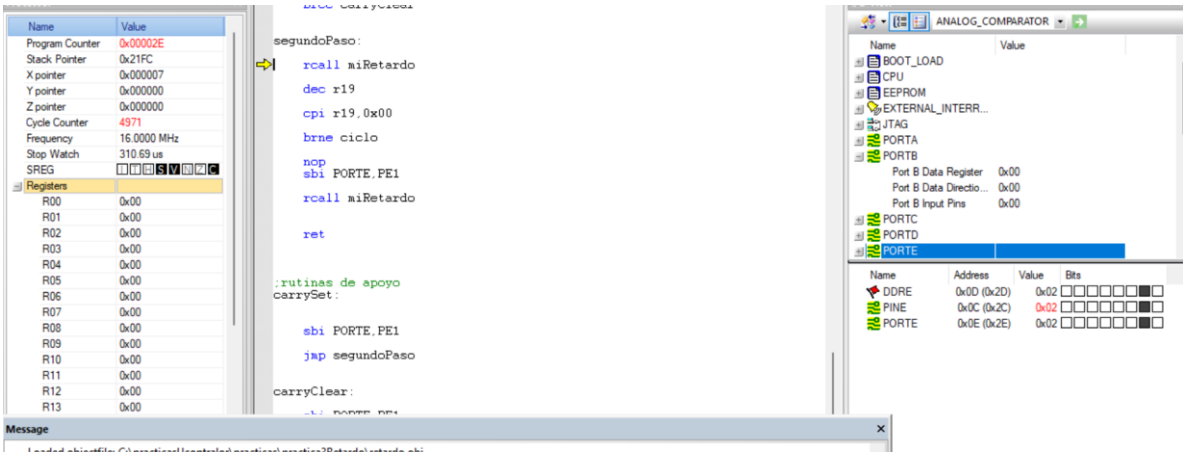


Como R18 tiene 0x55 en binario este seria R18 -> 0101 0101 por lo que el primero bit que sale es 1, carry se debe de establecer en 1.



Con la ayuda de los Branch verifica que carry está establecido o se borró, el primero brincara a “carrySet” si carry esta establecido, no es así, quiere decir que esta borrado, como esta en 1, brincara a “carrySet” en donde se establece en 1 el pin 1 del puerto E después este brinca a la etiqueta segundoPaso:





Dificultades y conclusión:

La parte en donde tuve muchos problemas a la hora de implementar sin duda fue la rutina de “miRetardo”, entendía lo que se pedía hice una ecuación de esqueleto, para saber por dónde ir, pero al programarlo era algo totalmente diferente, era porque no estaba adaptándolo de la manera correcta, por lo que el plasmar lo que decía las ecuaciones con el código fue algo en donde batalle poco, después empecé a hacer el código y ahora fue la ecuación que no me estaba dando el resultado correcto, me da algo cercano, pero no exactamente los 1648 ciclos, de ahí en fuera la rutina de prestar bits no fue difícil, solo que no sabía cómo pasar los bits al pin, pero se me ocurrió esa manera con la ayuda de los Branch.