

Materia:

Microprocesadores y
microcontroladores.



Reporte #7

Uso de Puertos y retardos mediante
Software.

Alumno:

Montoya Valdivia Omar Antonio:
1252892

Profesor:

Jesús García

Retardos por software:

Los retardos por Software consisten en que el microcontrolador se quede “enciclado” durante un tiempo. Es decir, es necesario usar uno o varios contadores que deberán ser decrementados, cuando dichos contadores lleguen a 0 habrá concluido el retardo.

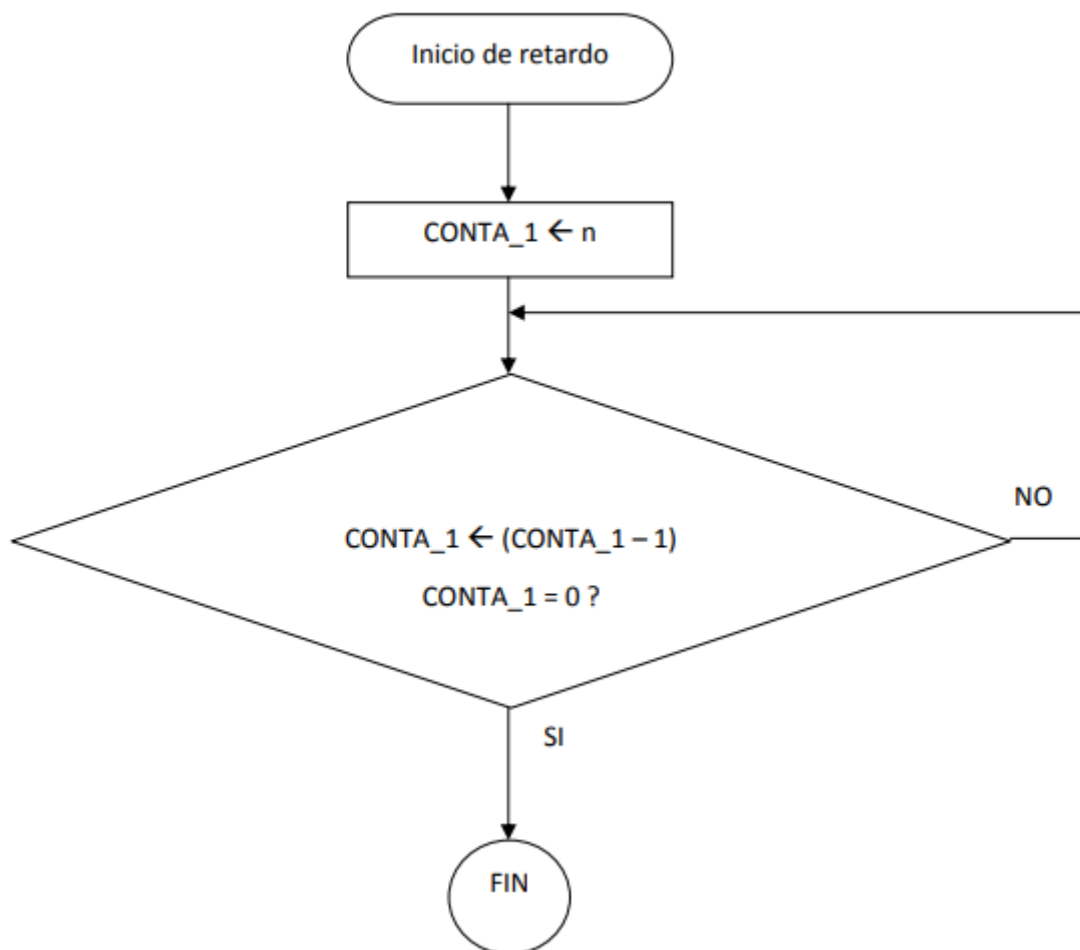


Figura 1: Ejemplo de un retardo por software

En código un retardo seria

```
for(i = 0; i < 100; i++);
```

también

```
i=100;  
while(--i);
```

En ambos casos el retardo dependería de:

- 1) las veces que se realiza el ciclo (ya sea el for o while).
- 2) del código que genera el compilador.
- 3) de la velocidad del procesador.

Como pueden ver, ustedes tienen control sobre el punto 1) dado que se define un valor de 100 en los ejemplos del for y while, pero para los otros puntos NO. En el caso de 2) depende de las opciones de optimización del compilador y en 3) de la velocidad de la computadora.

Mediante prueba y error se podría llegar a un retardo en tiempo específico, pero al llevar el código a otra computadora con otras características el retardo ya no se garantiza.

Una manera de tener control de 2) es diseñar el retardo en ensamblador, esto convirtiendo el código de alto nivel a ensamblador manualmente. Luego el código se introduce en lenguaje C de la forma inline de manera que el compilador no lo traduce y la pasa tal como fue escrito.

Por ejemplo:

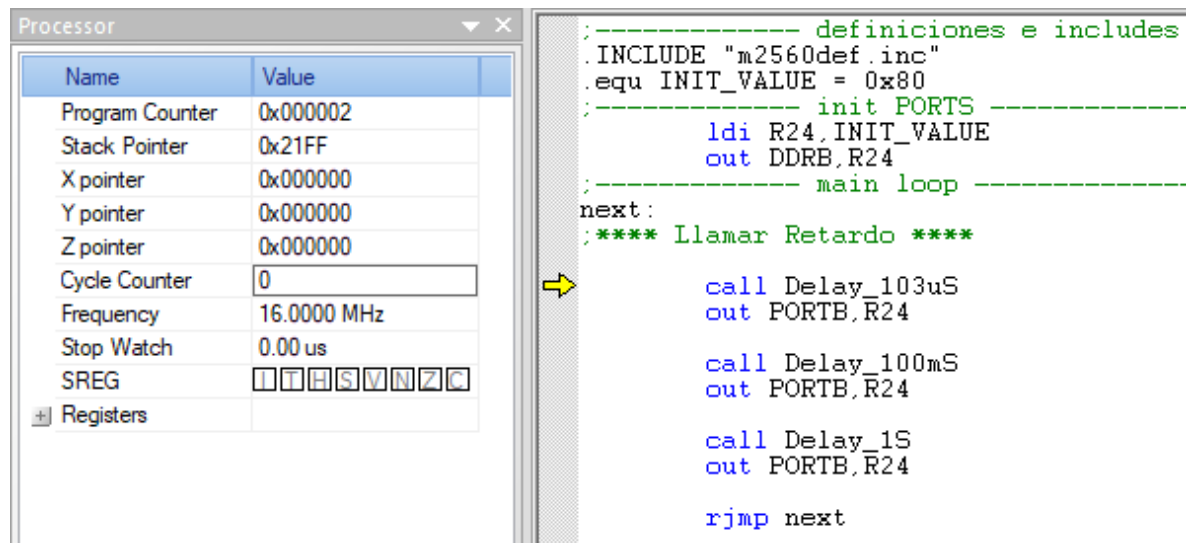
```
nxt:    ldi R24,0x05 ; [1]*(1)  
        nop ; [5]*(1)  
        dec R24 ; [5]*(1)  
        brne nxt ; [4]*(2) y [1]*(1)
```

Ahora, para conocer el tiempo total es necesario calcular el total de ciclos de reloj del código:

En total tenemos $1+5+5+(4*2)+1 = 20$ ciclos y ahora podemos determinar el tiempo total si conocemos la frecuencia con que opera el procesador. Por ejemplo si se opera a 8MHz tenemos que un ciclo de reloj tiene un período de $1/8\text{Mhz} = 125\text{nS}$, por tanto el tiempo total de la secuencia es $125\text{nS} \times 20 = 2.5 \text{ uS}$.

Retardo de 103us

Antes de llamar al retardo



The Processor window shows the following values:

Name	Value
Program Counter	0x000002
Stack Pointer	0x21FF
X pointer	0x000000
Y pointer	0x000000
Z pointer	0x000000
Cycle Counter	0
Frequency	16.0000 MHz
Stop Watch	0.00 us
SREG	0x00
Registers	

The assembly code is as follows:

```
;----- definiciones e includes
INCLUDE "m2560def.inc"
.equ INIT_VALUE = 0x80
;----- init PORTS -----
ldi R24, INIT_VALUE
out DDRB, R24
;----- main loop -----
next:
;**** Llamar Retardo ****
call Delay_103uS
out PORTB, R24

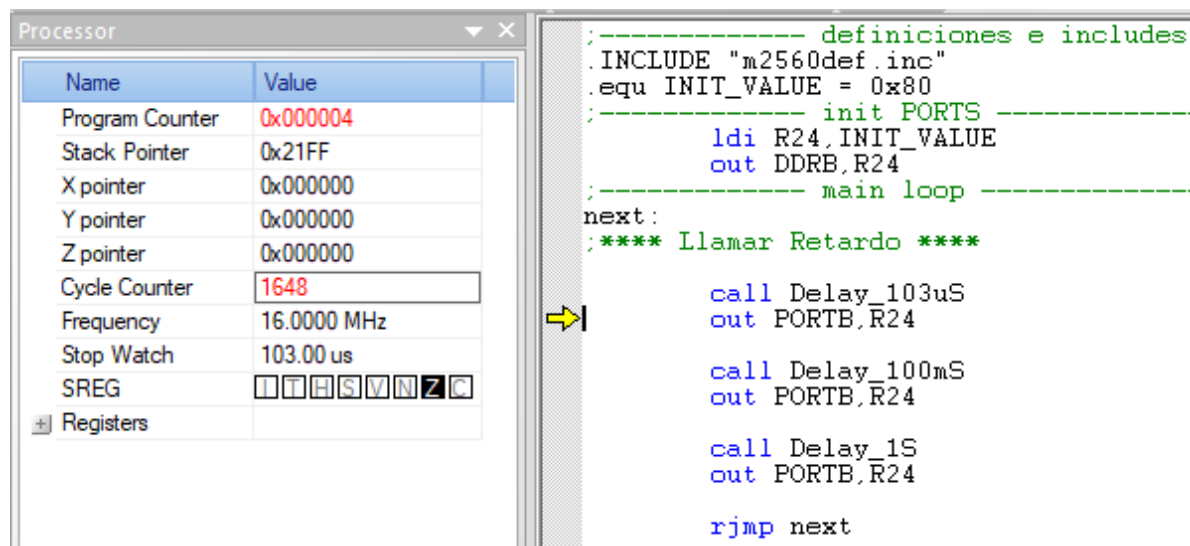
call Delay_100mS
out PORTB, R24

call Delay_1S
out PORTB, R24

rjmp next
```

Prueba 1: Contador en 0 para iniciar el retardo

Después de hacer el procedimiento del retardo



The Processor window shows the following values:

Name	Value
Program Counter	0x000004
Stack Pointer	0x21FF
X pointer	0x000000
Y pointer	0x000000
Z pointer	0x000000
Cycle Counter	1648
Frequency	16.0000 MHz
Stop Watch	103.00 us
SREG	0x00
Registers	

The assembly code is the same as in the previous screenshot:

```
;----- definiciones e includes
INCLUDE "m2560def.inc"
.equ INIT_VALUE = 0x80
;----- init PORTS -----
ldi R24, INIT_VALUE
out DDRB, R24
;----- main loop -----
next:
;**** Llamar Retardo ****
call Delay_103uS
out PORTB, R24

call Delay_100mS
out PORTB, R24

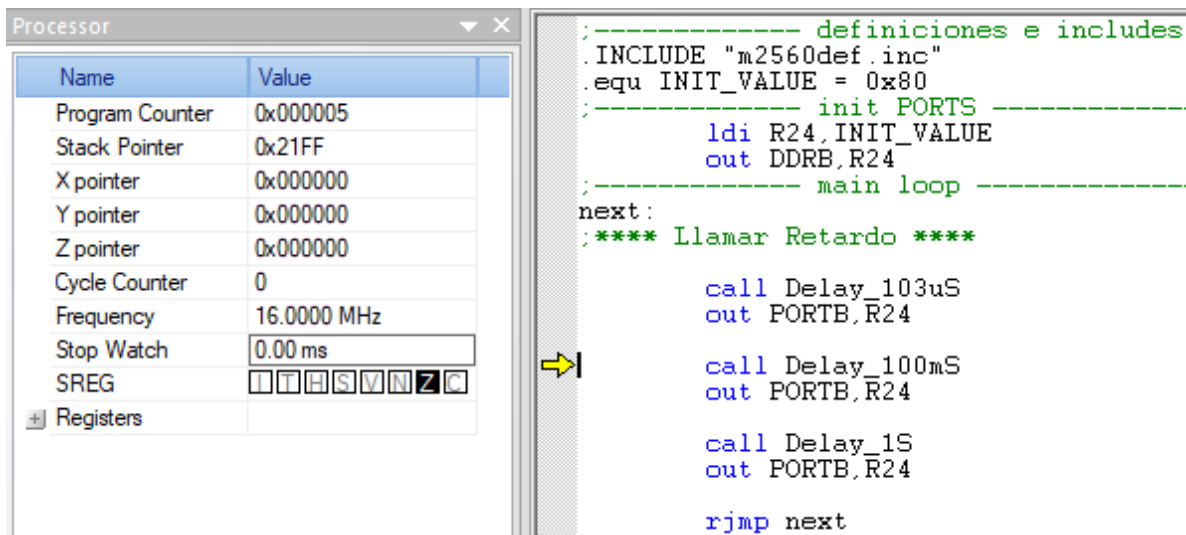
call Delay_1S
out PORTB, R24

rjmp next
```

Prueba 2: Correcta implementación del retardo de 103us

Retardo de 100ms

Contador puesto en 0 antes de llamar al procedimiento



The Processor window shows the following values:

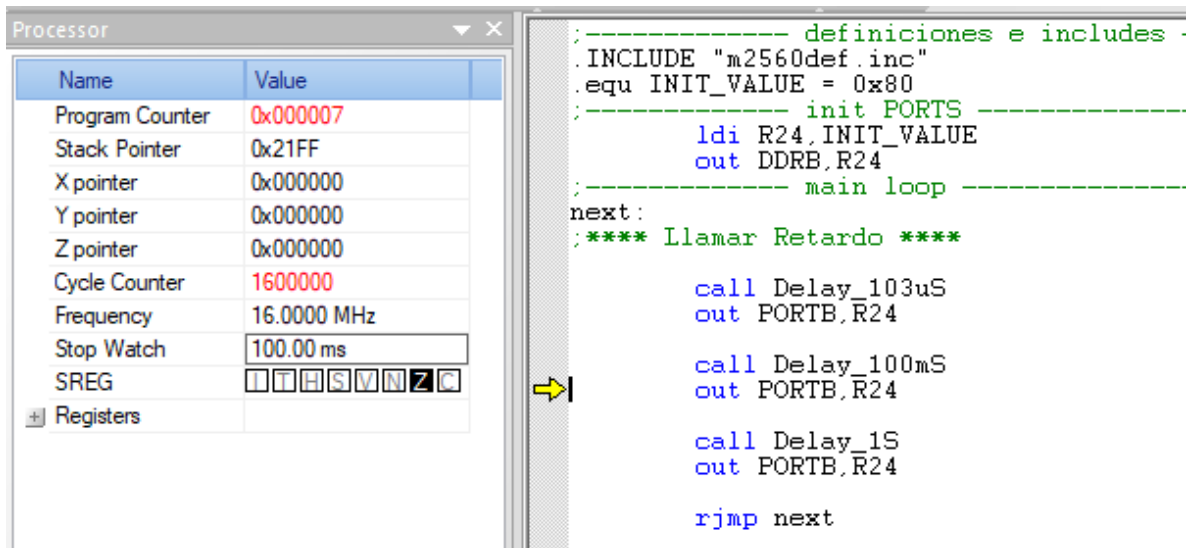
Name	Value
Program Counter	0x000005
Stack Pointer	0x21FF
X pointer	0x000000
Y pointer	0x000000
Z pointer	0x000000
Cycle Counter	0
Frequency	16.0000 MHz
Stop Watch	0.00 ms
SREG	0x00
Registers	

The assembly code in the main window is as follows:

```
;----- definiciones e includes -----  
.INCLUDE "m2560def.inc"  
.equ INIT_VALUE = 0x80  
;----- init PORTS -----  
    ldi R24, INIT_VALUE  
    out DDRB, R24  
;----- main loop -----  
next:  
;**** Llamar Retardo ****  
    call Delay_103uS  
    out PORTB, R24  
    call Delay_100mS  
    out PORTB, R24  
    call Delay_1S  
    out PORTB, R24  
    rjmp next
```

Prueba 3: contador puesto en 0 para el retardo de 100ms

Después de ejecutar el procedimiento del retardo



The Processor window shows the following values after execution:

Name	Value
Program Counter	0x000007
Stack Pointer	0x21FF
X pointer	0x000000
Y pointer	0x000000
Z pointer	0x000000
Cycle Counter	1600000
Frequency	16.0000 MHz
Stop Watch	100.00 ms
SREG	0x00
Registers	

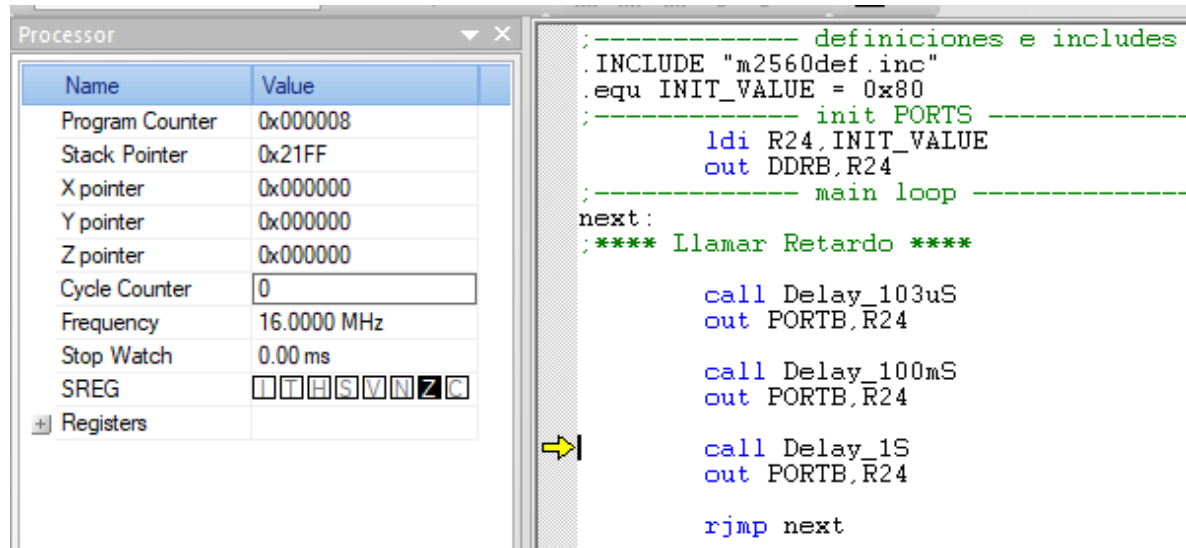
The assembly code in the main window is the same as in the previous screenshot:

```
;----- definiciones e includes -----  
.INCLUDE "m2560def.inc"  
.equ INIT_VALUE = 0x80  
;----- init PORTS -----  
    ldi R24, INIT_VALUE  
    out DDRB, R24  
;----- main loop -----  
next:  
;**** Llamar Retardo ****  
    call Delay_103uS  
    out PORTB, R24  
    call Delay_100mS  
    out PORTB, R24  
    call Delay_1S  
    out PORTB, R24  
    rjmp next
```

Prueba 4: Correcta implementación del retardo de 100ms

Retardo de 1s

Se coloca en 0 el contador de ciclos para llamar al retardo de 1s



The Processor window shows the following values:

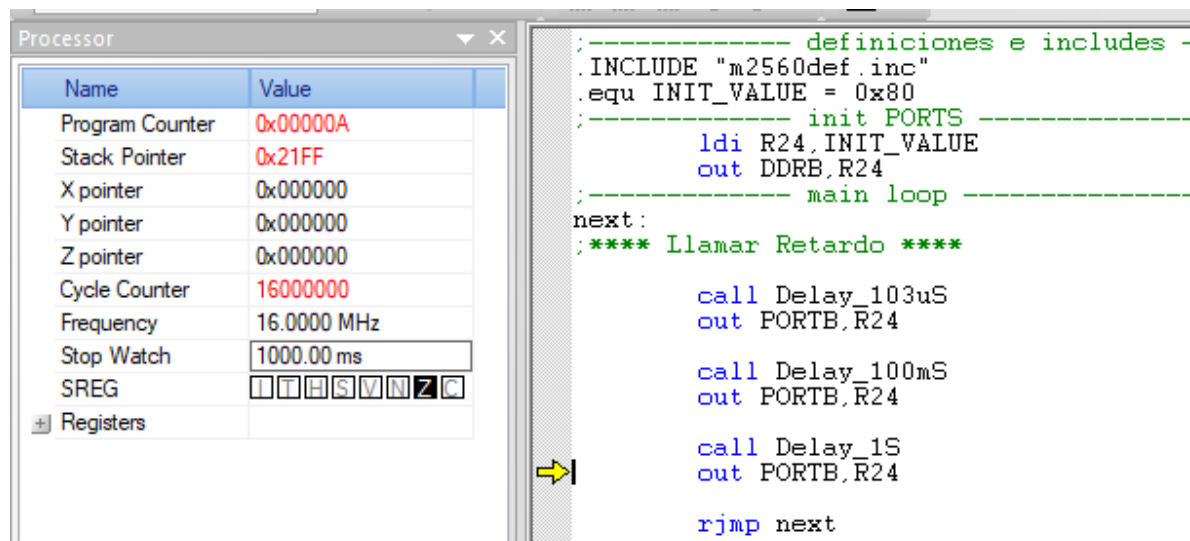
Name	Value
Program Counter	0x000008
Stack Pointer	0x21FF
X pointer	0x000000
Y pointer	0x000000
Z pointer	0x000000
Cycle Counter	0
Frequency	16.0000 MHz
Stop Watch	0.00 ms
SREG	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Registers	

The assembly code is as follows:

```
;----- definiciones e includes -----  
.INCLUDE "m2560def.inc"  
.equ INIT_VALUE = 0x80  
;----- init PORTS -----  
    ldi R24, INIT_VALUE  
    out DDRB, R24  
;----- main loop -----  
next:  
;**** Llamar Retardo ****  
    call Delay_103uS  
    out PORTB, R24  
  
    call Delay_100mS  
    out PORTB, R24  
  
    call Delay_1S  
    out PORTB, R24  
  
    rjmp next
```

Prueba 5: Retardo de 1s

Después de realizar el procedimiento con el retardo correspondiente



The Processor window shows the following values:

Name	Value
Program Counter	0x00000A
Stack Pointer	0x21FF
X pointer	0x000000
Y pointer	0x000000
Z pointer	0x000000
Cycle Counter	16000000
Frequency	16.0000 MHz
Stop Watch	1000.00 ms
SREG	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
Registers	

The assembly code is the same as in the previous screenshot:

```
;----- definiciones e includes -----  
.INCLUDE "m2560def.inc"  
.equ INIT_VALUE = 0x80  
;----- init PORTS -----  
    ldi R24, INIT_VALUE  
    out DDRB, R24  
;----- main loop -----  
next:  
;**** Llamar Retardo ****  
    call Delay_103uS  
    out PORTB, R24  
  
    call Delay_100mS  
    out PORTB, R24  
  
    call Delay_1S  
    out PORTB, R24  
  
    rjmp next
```

Prueba 6: Retardo de 1s correctamente implementado

Conclusión:

Al realizar la práctica, aprendí a realizar correctamente los retardos que se pedían mediante software realizando ciclos anidados. Dependiendo el tiempo que se requiera es la cantidad de ciclos que debes anidar, si son tiempos de retardo muy pequeños se pueden implementar con un sencillo loop, pero si son retardos de segundos ya se requieren más ciclos anidados.

Los retardos se implementan de acuerdo a la aplicación, hay algunas aplicaciones que requieran ciertos retardos para que otro dispositivo que se esté usando responda de manera correcta.