

# Ejemplo de temporizador MCU AVR® de 8 bits



Esta página ilustra varios métodos para configurar los temporizadores en un MCU AVR ® de 8 bits . Se proporciona una descripción general de los temporizadores en un AVR antes de presentar el ejemplo paso a paso.

## Requisitos para ejecutar los ejemplos prácticos

### Requisitos de hardware

- ATmega328PB Mini tablero explicado
- Cable micro USB

Herramienta	Sobre		
 ( <a href="http://www.atmel.com/tools/MEGA328PB-XMINI.aspx">http://www.atmel.com/tools/MEGA328PB-XMINI.aspx</a> )	Mini kit de evaluación <b>ATmega328PB Xplained</b>	 ( <a href="http://www.atmel.com/tools/MEGA328PB-XMINI.aspx">http://www.atmel.com/tools/MEGA328PB-XMINI.aspx</a> )	 ( <a href="https://www.microchipdirect.co">https://www.microchipdirect.co</a> )


El kit de evaluación ATmega328PB Xplained Mini es una plataforma de hardware para evaluar el microcontrolador Atmel ATmega328PB. NO se necesita un kit de depurador externo para ejecutar estos laboratorios. El ATmega328PB tiene un depurador incorporado completamente integrado.

Este ejercicio práctico demuestra cómo configurar los diferentes temporizadores.

### Herramientas de software

		Instaladores				
Herramienta	Sobre	Windows	linux	Mac OS X	Instruccione de instalación	
Entorno de desarrollo integrado Atmel® Studio	 (/atstudio:start)	 ( <a href="http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe">http://studio.download.atmel.com/7.0.1931/as-installer-7.0.1931-full.exe</a> )			 (/install:atstudi	

### Archivos adicionales

archivos
 <b>Guía del usuario del tablero explicado</b> ( <a href="http://www.atmel.com/Images/Atmel-42287-ATmega328P-Xplained-Mini-User-Guide_UserGuide.pdf">http://www.atmel.com/Images/Atmel-42287-ATmega328P-Xplained-Mini-User-Guide_UserGuide.pdf</a> )

## Descripción general de los temporizadores en MCU AVR de 8 bits

El microcontrolador ATmega328PB tiene cinco temporizadores/contadores:

Temporizador 0	TC0	Temporizador/contador de 8 bits con PWM
Temporizador 1	TC1	Temporizador/contador de 16 bits con PWM y funcionamiento asíncrono.
Temporizador 2	TC2	Temporizador/contador de 8 bits con PWM y funcionamiento asíncrono.
Temporizador 3	TC3	Temporizador/contador de 16 bits con PWM y funcionamiento asíncrono.
Temporizador 4	TC4	Temporizador/contador de 16 bits con PWM y funcionamiento asíncrono.

Definiciones:

Nomenclatura de registro	
ABAJO	El contador llega al FONDO cuando llega a cero (0x00 para contadores de 8 bits y 0x0000 para contadores de 16 bits)
MÁX.	El contador alcanza su valor máximo cuando pasa a ser 0x0F (15 decimal) para contadores de 8 bits y 0x00FF (255 decimal) para contadores de 16 bits
PARTE SUPERIOR	El contador llega al TOP cuando su valor llega a ser igual al valor más alto posible. Al valor TOP se le puede asignar el valor fijo MAX o el valor almacenado en el registro OCRxA. Esta asignación depende del modo de funcionamiento

## Temporizador 0 - Temporizador/Contador de 8 bits con PWM

Timer/Counter0 (TC0) es un módulo de temporizador/contador de uso general de 8 bits, con dos unidades de comparación de salida independientes y compatibilidad con PWM.

### Registros TC0

El registro del temporizador/contador 0 (TCNT0) y los registros de comparación de salida TC0x (OCR0x) son registros de 8 bits. Todas las señales de solicitud de interrupción son visibles en el registro de bandera de interrupción del temporizador 0 (TIFR0). Todas las interrupciones se enmascaran individualmente con el registro de máscara de interrupción del temporizador 0 (TIMSK0).

**Name:** TCCR0B  
**Offset:** 0x45  
**Reset:** 0x00  
**Property:** When addressing as I/O Register: address offset is 0x25

Bit	7	6	5	4	3	2	1	0
	FOC0A	FOC0B			WGM02	CS02	CS01	CS00
Access	R/W	R/W			R/W	R/W	R/W	R/W
Reset	0	0			0	0	0	0

(/local--files/8avr:avrtimer/TCCR0B.PNG)

### Fuentes de reloj del temporizador/contador TC0

TC0 puede sincronizarse con una fuente de reloj interna o externa. La fuente de reloj se selecciona escribiendo en los bits de selección de reloj (CS02:0) en el registro de control de temporizador/contador (TCCR0B).

#### Bits 2:0 – CS0n: Selección de reloj [n = 0..2]

Los tres bits de selección de reloj seleccionan la fuente de reloj que utilizará el temporizador/contador.

CS02	CS01	CS00	Descripción
0	0	0	Sin fuente de reloj (temporizador detenido)
0	0	1	clkio/1 (sin preescalado)
0	1	0	clkio/8 (Del prescaler)
0	1	1	clkio/64 (Del prescaler)
1	0	0	clkio/256 (Del prescaler)
1	0	1	clkio/1012 (Del prescaler)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en flanco descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en flanco ascendente)

### Unidad de contador TC0

Dependiendo del modo de operación utilizado, T0 se borra, aumenta o disminuye en cada reloj del temporizador (clkT0). clkT0 se puede generar desde una fuente de reloj externa o interna, seleccionada por los bits de selección de reloj (CS0[2:0]).

La secuencia de conteo está determinada por la configuración de los bits WGM01 y WGM00 ubicados en el registro de control T0 A (TCCR0A) y el bit WGM02 ubicado en el registro de control de temporizador/contador B (TCCR0B).

**Bits 1:0 – WGM0n: Modo de generación de forma de onda [n = 1:0]**

Combinados con el bit WGM02 que se encuentra en el registro TCCR0B, estos bits controlan la secuencia de conteo del contador, la fuente del valor máximo del contador (TOP) y qué tipo de generación de forma de onda se utilizará. Los modos de operación admitidos por la unidad de temporizador/contador son: modo normal (contador), modo Borrar temporizador en comparación (CTC) y dos tipos de modos de modulación de ancho de pulso (PWM).

Tabla 1-2 Bit de modo de generación de forma de onda Descripción

Modo	WGM2:0	Modo de operación	PARTE SUPERIOR	Actualización OCR0x	Indicador TOV activado
0	0 0 0	Normal	0xFF	Inmediato	MÁX.
1	0 0 1	Fase PWM correcta	0xFF	PARTE SUPERIOR	ABAJO
2	0 1 0	CTC	OCRA	Inmediato	MÁX.
3	0 1 1	PWM rápido	0xFF	ABAJO	MÁX.
4	1 0 0	Reservado	~	~	~
5	1 0 1	Fase PWM correcta	OCRA	PARTE SUPERIOR	ABAJO
6	1 1 0	Reservado	~	~	~
7	1 1 1	PWM rápido	OCRA	ABAJO	MÁX.

**Nota:**

1. MAX = 0xFF
2. INFERIOR = 0x00

**Modos de operación para TC0**

El modo de operación determina el comportamiento de TC0 y los pines de comparación de salida. Se define mediante la combinación de los bits del modo de generación de forma de onda y los bits del modo de salida de comparación en los registros de control del temporizador/contador A y B (TCCR0B.WGMn2, TCCR0A.WGM01, TCCR0A.WGM00 y TCCR0A.COM0x[1:0]).

Los modos de operación disponibles para TC0 son:

- Modo normal
- Borrar temporizador en el modo de comparación de coincidencias (CTC)
- Modo PWM rápido
- Modo PWM de corrección de fase

**Borrar temporizador en el modo de comparación de coincidencias**

En el modo Clear Timer on Compare (WGM0[2:0]=0x2), el registro OCR0A se usa para manipular la resolución del contador: el contador se borra a CERO cuando el valor del contador (TCNT0) coincide con el OCR0A. El OCR0A define el valor máximo del contador y, por lo tanto, también su resolución.

El valor del contador (TCNT0) aumenta hasta que se produce una coincidencia de comparación entre TCNT0 y OCR0A, y luego se borra el contador (TCNT0). Se puede generar una interrupción cada vez que el valor del contador alcance el valor SUPERIOR configurando el indicador OCF0A. Si la interrupción está habilitada, la rutina del controlador de interrupciones se puede usar para actualizar el valor SUPERIOR.

La frecuencia de la forma de onda se define mediante la siguiente ecuación:

$$f_{\text{OCnx}} = \frac{f_{\text{clk\_I/O}}}{2 \cdot N \cdot (1 + \text{OCRnx})}$$

(/local--files/8avr:avrtimer/freqz0.PNG)

N representa el factor del preescalador (1, 8, 64, 256 o 1024).

**TC1, TC3 y TC4: temporizador/contador de 16 bits con PWM**

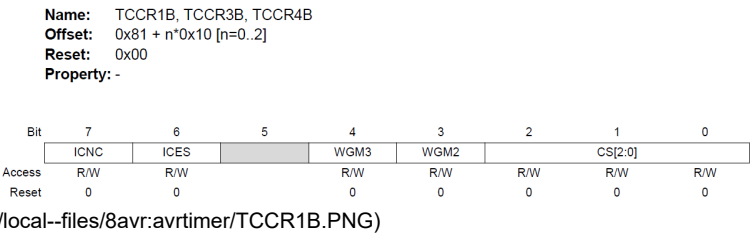
Las unidades de temporizador/contador de 16 bits permiten la sincronización precisa de la ejecución del programa (gestión de eventos), la generación de ondas y la medición de la sincronización de la señal.

**Registros (TC1, TC3, TC4)**

- El temporizador/contador (TCNTn), los registros de comparación de salida (OCRA/B) y el registro de captura de entrada (ICRn) son todos registros de 16 bits.
- Los registros de control de temporizador/contador (TCCRnA/B) son registros de 8 bits y no tienen restricciones de acceso a la CPU.
- Las señales de solicitudes de interrupción (abreviadas como Int.Req. en el diagrama de bloques) son todas visibles en el Registro de indicadores de interrupción del temporizador (TIFRn). Todas las interrupciones se enmascaran individualmente con el registro de máscara de interrupción del temporizador (TIMSKn).

Fuentes de temporizador/contador de reloj (TC1, TC3, TC4)

El temporizador/contador puede sincronizarse con una fuente de reloj interna o externa. La fuente de reloj se selecciona mediante la lógica de selección de reloj, que está controlada por los bits de selección de reloj en el registro B de control de temporizador/contador (TCCRnB.CS[2:0]).



Bits 2:0 – CS[2:0]: Selección de reloj [n = 0..2]

Los tres bits de selección de reloj seleccionan la fuente de reloj que utilizará el temporizador/contador.

CS02	CS01	CS00	Descripción
0	0	0	Sin fuente de reloj (Temporizador detenido)
0	0	1	clkio/1 (sin preescalado)
0	1	0	clkio/8 (Del prescaler)
0	1	1	clkio/64 (Del prescaler)
1	0	0	clkio/256 (Del prescaler)
1	0	1	clkio/1012 (Del prescaler)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en flanco descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en flanco ascendente)

Unidad de contador (TC1, TC3, TC4)

TC1, TC3 y TC4 son contadores bidireccionales programables de 16 bits.

Cada contador de 16 bits se asigna a dos ubicaciones de memoria de E/S de 8 bits: Contador alto (TCNTnH) que contiene los ocho bits superiores del contador y Contador bajo (TCNTnL) que contiene los ocho bits inferiores.

Dependiendo del modo de operación seleccionado, el contador se borra, incrementa o decrementa en cada reloj del temporizador (clkTn). El reloj clkTn se puede generar a partir de una fuente de reloj externa o interna, según lo seleccionado por los bits de selección de reloj en el registro de control de temporizador/contador B (TCCRnB.CS[2:0]).

La secuencia de conteo está determinada por la configuración de los bits del modo de generación de forma de onda en los registros de control de temporizador/contador A y B (TCCRnB.WGM[3:2] y TCCRnA.WGM[1:0]).

Modos de operación (TC1, TC3, TC4)

El modo de operación está determinado por la combinación de los bits del modo de generación de forma de onda (WGM[3:0]) y el modo de salida de comparación (TCCRnA.COMx[1:0]).

Los modos de funcionamiento disponibles son:

- Modo normal
- Borrar temporizador en el modo de comparación de coincidencias (CTC)
- Modo PWM rápido
- Modo PWM de corrección de fase
- Modo PWM de corrección de fase y frecuencia

Modo PWM rápido

Los modos Fast Pulse Width Modulation o Fast PWM (modos 5, 6, 7, 14 y 15, WGM[3:0]= 0x5, 0x6, 0x7, 0xE, 0xF) proporcionan una opción de generación de forma de onda PWM de alta frecuencia. El Fast PWM se diferencia de las otras opciones de PWM por su operación de pendiente única. El contador cuenta de ABAJO a ARRIBA y luego se reinicia desde ABAJO.

En el modo Fast PWM, el contador se incrementa hasta que el valor del contador coincida con uno de los valores fijos 0x00FF, 0x01FF o 0x03FF (WGM[3:0] = 0x5, 0x6 o 0x7), el valor en ICRn (WGM[3: 0]=0xE), o el valor en OCRnA (WGM[3:0]=0xF). A continuación, el contador se borra en el siguiente ciclo de reloj del temporizador.

La frecuencia PWM para la salida se puede calcular mediante la siguiente ecuación:

$$f_{OCnxPWM} = \frac{f_{clk_{I/O}}}{N \cdot (1 + TOP)}$$

(/local--files/8avr:avrtimer/freqz1.PNG)



**Nota:**

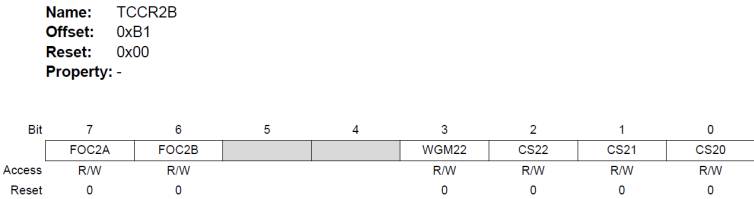
- La “n” en los nombres de bit y registro indica el número de dispositivo (n = 0 para el temporizador/contador 0) y la “x” indica la unidad de comparación de salida (A/B).
- N representa el divisor de preescala (1, 8, 64, 256 o 1024).

## TC2 - Timer/Counter2 de 8 bits con PWM y funcionamiento asíncrono

Timer/Counter2 (TC2) es un módulo de temporizador/contador de 8 bits de doble canal y propósito general.

### Registros TC2

El temporizador/contador (TCNT2) y el registro de comparación de salida (OCR2A y OCR2B) son registros de 8 bits. Todas las señales de solicitud de interrupción son visibles en el registro de indicadores de interrupción del temporizador (TIFR2). Todas las interrupciones se enmascaran individualmente con el registro de máscara de interrupción del temporizador (TIMSK2).



(/local--files/8avr:avrtimer/TCCR2B.PNG)

### Fuentes de reloj TC2

TC2 puede sincronizarse con una fuente de reloj síncrona interna o asíncrona externa:  
Los tres bits de selección de reloj (CS2:CS0) seleccionan la fuente de reloj que utilizará el temporizador/contador.

CS02	CS01	CS00	Descripción
0	0	0	Sin fuente de reloj (temporizador detenido)
0	0	1	clkio/1 (sin preescalado)
0	1	0	clkio/8 (Del prescaler)
0	1	1	clkio/64 (Del prescaler)
1	0	0	clkio/256 (Del prescaler)
1	0	1	clkio/1012 (Del prescaler)
1	1	0	Fuente de reloj externa en el pin T0 (reloj en flanco descendente)
1	1	1	Fuente de reloj externa en el pin T0 (reloj en flanco ascendente)

### Unidad de contador TC2

Dependiendo del modo de operación utilizado, el contador se borra, incrementa o decrementa en cada reloj del temporizador (clkT2). clkT2 se puede generar desde una fuente de reloj externa o interna, seleccionada por los bits de selección de reloj (CS2[2:0]).

La secuencia de conteo está determinada por la configuración de los bits WGM21 y WGM20 ubicados en el registro de control de temporizador/contador (TCCR2A) y el bit WGM22 ubicado en el registro de control de temporizador/contador B (TCCR2B).

## Modos de funcionamiento de TC2

El modo de operación, es decir, el comportamiento del temporizador/contador y los pines de comparación de salida, se define mediante la combinación del modo de generación de forma de onda (WGM2[2:0]) y el modo de salida de comparación (COM2x[1:0]) pedacitos

Los modos de funcionamiento disponibles son:

- Modo normal
- Borrar temporizador en el modo de comparación de coincidencias (CTC)
- Modo PWM rápido
- Modo PWM de corrección de fase

## Modo normal

En el modo Normal (WGM22:0 = 0) el sentido de conteo es siempre ascendente (incremental), sin que se borre el contador. El contador cambiará a 0c00 cuando pase su valor máximo de 8 bits (TOP = 0xFF).

En funcionamiento normal, el indicador de desbordamiento del temporizador/contador (TOV2) se establecerá en el mismo ciclo de reloj del temporizador cuando el TCNT2 se vuelve cero. El indicador TOV2, en este caso, se comporta como un noveno bit, excepto que solo se establece, no se borra. Sin embargo, combinado con la interrupción de desbordamiento del temporizador que borra automáticamente el indicador TOV2, el software puede aumentar la resolución del temporizador. No hay casos especiales a considerar en el modo Normal, se puede escribir un nuevo valor de contador en cualquier momento.



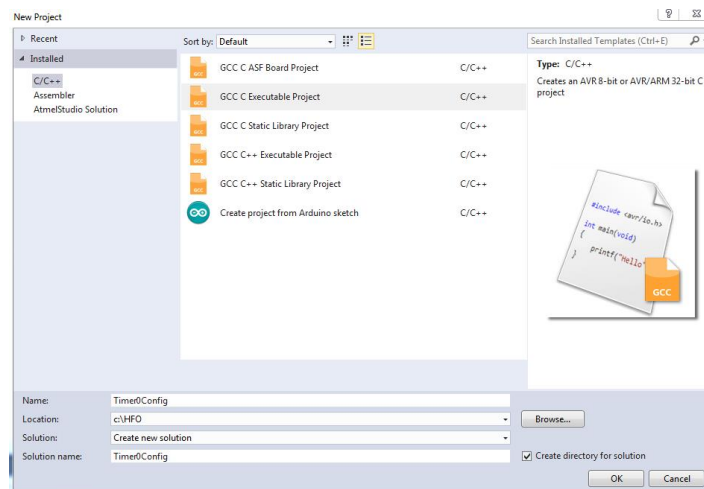
Independientemente del modo que se utilice, el programador debe recordar dos cosas:

1. El temporizador debe iniciarse seleccionando la fuente del reloj.
2. Si se utilizan interrupciones, deben estar habilitadas.

## ! Paso a paso

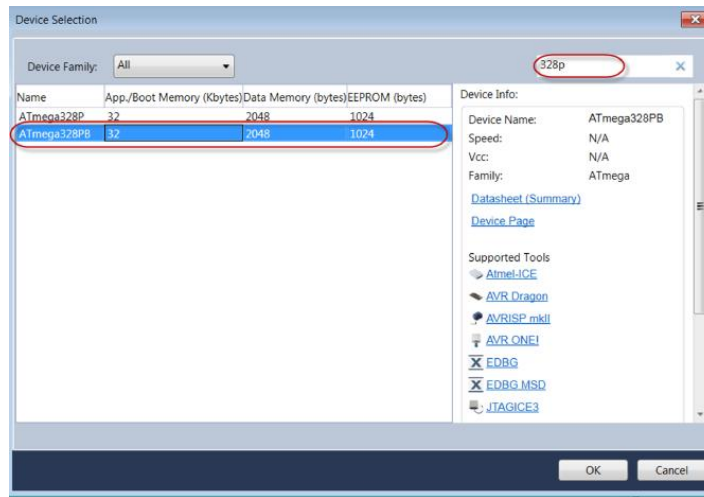
### Creación de proyectos

- 1 Abrir Atmel Studio 7
- 2 Seleccione **Archivo > Nuevo > Proyecto**
- 3 En la ventana **Nuevo proyecto**, seleccione **Proyecto ejecutable GCC C** y asigne el nombre del proyecto a "Timer0Config". Establezca la ubicación de los archivos del proyecto. (Este ejemplo usa "C:\HFO\TimersConfigurations"). Haga clic en **Aceptar**.



(/local--files/8avr:avrtimer/new-project.jpg)

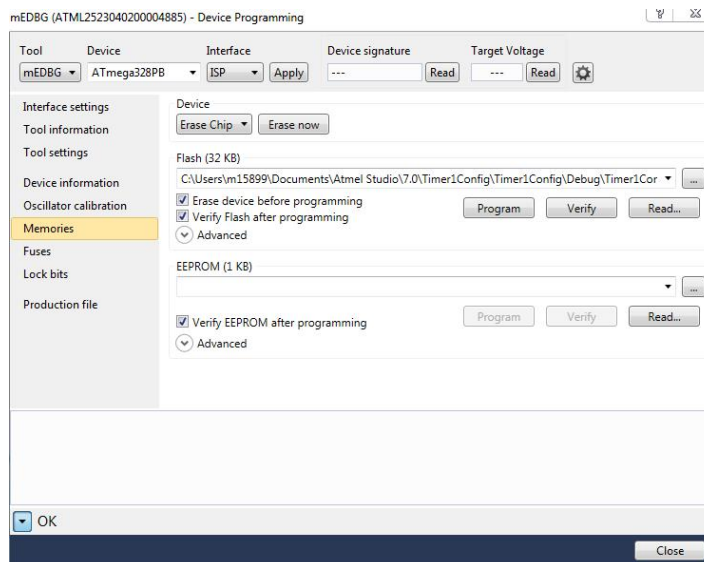
- 4 En la barra de búsqueda de la ventana **Selección de dispositivo**, escriba "328p", luego seleccione el dispositivo **Atmega328PB**, haga clic en **Aceptar**.



(/local--files/8avr:avrtimer/select-device.jpg)

## Programando la placa

- 5 Seleccione **Build > Build Solution** en el menú superior para compilar el código. Verá un mensaje "BUILD SUCCEEDED" en la ventana de salida de Studio 7.
- 6 Seleccione **Herramientas > Programación de dispositivos**, asegúrese de que la configuración sea como se muestra en la imagen a continuación y haga clic en **Aplicar**.



(/local--files/8avr:avrtimer/program.jpg)

- 7 Selecciona **Memorias > Programa** Espera hasta que aparezca el mensaje "Verificando Flash...OK".

## Modificaciones del proyecto



avr/io.h y avr/interrupt.h deben estar **#incluidos** en main.c para que este proyecto se compile y ejecute.

- io.h proporciona la definición de todos los periféricos AVR.
- interrupt.h es necesario para todas las aplicaciones que utilizan interrupciones.

## 8 Modificación 1: parpadeo de un LED usando TC0

En este ejemplo, el temporizador 0 parpadea el LED conectado a PB5 cada 32 ms.

- a Modifique la función principal agregando el siguiente código:



```

1  int main(void)      ?
2
3  {
4      //set RB5 as output
5      DDRB |= 1 << DDRB5;
6
7      //call TMR0 initial
8      init_TC0();
9
10     //enable interrupt
11     sei();
12
13     while (1)
14     {
15         //main loop
dieciséis    }
17 }

```

**b** Cree la función `init_TC0 ( )` en `main.c`. Agregue el siguiente código:

```

1  void init_TC0(void)  ?
2  {
3
4      // Set the Timer Mode to CTC
5      TCCR0A |= (1 << WGM01);
6
7      // Set the value that you want
8      OCR0A = 0xF9; //249
9
10     //Set the ISR COMPA vect
11     TIMSK0 |= (1 << OCIE0A);
12
13     // set prescaler to 1024
14     TCCR0B |= (1 << CS00) | (1 << CS02);
15 }

```

El código anterior hace lo siguiente:

- Configura el Timer 0 en modo CTC configurando el bit WGM correspondiente en el registro TCCR0A;
- Establece el valor TOP en el registro OCR0A calculado usando la ecuación 1 para hacer parpadear el LED a 30 kHz;
- Habilita la interrupción del temporizador;
- Establece el escalador previo a 1024 configurando los bits CS00 y CS02 en el registro TCCR0B;

**c** Agregue la rutina de servicio de interrupción a `main.c` :

```

1  ISR (TIMER0_COMPA_vect)  ?
2  {
3      //event to be executed every
4      counter++;
5      if (counter == MyTimerConstant)
6      {
7          counter = 0;
8          PORTB ^= 1 << PORTB5;
9      }
10 }

```



El uso de la constante *MyTimerConstant* y la variable *contador* es opcional. Si se usa, cambiar el valor de MyTimer Constant alterará el tiempo que tarda el LED en parpadear.

## 9 Modificación 2: uso de TC1 en modo PWM para atenuar el LED

Este ejemplo usa TC1 para generar una señal PWM que se alimenta a través de un pin de E/S para controlar un LED. Cambiar el ciclo de trabajo de PWM dará como resultado un cambio en el brillo del LED.

**a** Modifique `main()` agregando el siguiente código.





```

1  int main(void)
2
3  {
4      //set direction c
5      DDRB |= 1 << DDRE
6
7      init_TC1_pwm();
8
9      //enable global i
10     sei();
11
12     while (1)
13     {
14         //main loop
15     }
16 }

```

Para configurar el bit de dirección del pin PB1, escriba el bit DDB1 en 1 lógico en el registro DDRB. En la miniplaca ATmega328PB Xplained, el LED está conectado a PB5 y PWM se genera en PB1. Para ver la atenuación del LED, conecte un cable de PB5 a PB1 en la miniplaca ATmega328PB Xplained, como se muestra a continuación.

- b** Crea la función `init_TC1_pwm()` antes del bucle principal con el siguiente código:

```

1  void init_TC1_pwm(void)
2  {
3      //clear 0CnA on compare ma
4      TCCR1A = (1 << COM1A1);
5
6      //the counting sequence is
7      TCCR1A |= (1 << WGM10);
8      TCCR1B |= (1 << WGM12);
9
10     //256 prescaler clock sele
11     TCCR1B |= (0 << CS10) | (1
12
13     //enable interrupts
14     TIMSK0 |= (1 << OCIE1A);
15 }

```

`init_TC1_PWM()` realiza lo siguiente"

- Establece los bits COMA1=1 y COMA0=0 (modo no inversor de la tabla de la hoja de datos Modo de salida de comparación, PWM rápido en el registro TCCR1A;
- Configura Bits 1:0 – WGM[1:0]:Modo de generación de forma de onda en consecuencia para Fast PWM, modo de 8 bits;
- Verifica la configuración de la fuente de reloj y el preescalador que utilizará el temporizador/contador, que decide la frecuencia del PWM. La fuente de reloj interna dividida por 256 se configura escribiendo en los bits CS10 y CS12 en el registro TCCR1B;
- Habilita la interrupción del temporizador;

- C** Agregue la función ISR después del bucle principal para ejecutar el evento LED de atenuación:

```

1  ISR (TIMER1_COMPA_vect) // t2.
2  {
3      duty_cycle--;
4      if (duty_cycle == 0) {
5          duty_cycle = 0xFF;
6      }
7      OCR1A = duty_cycle;
8  }
9  }

```

El registro OC1RA decide el ciclo de trabajo de PWM. Para atenuar el LED, disminuya gradualmente el ciclo de trabajo.

## 10 Parpadeo de un LED usando TC2

Para este ejemplo, el temporizador 2 (TC2) está configurado para hacer parpadear el LED conectado a PB5 en un período de 3 segundos.

- a** Modifique la función principal agregando el siguiente código:

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```


1  int main(void)      ?
2  {
3      //set direction of
4      DDRB |= 1 << DDRB5;
5
6      /* Timer clock = I,
7      TCCR2B = (1 << CS2;
8
9      /* Clear overflow 1
10     TIFR2 = 1 << TOV2;
11
12     /* Enable Overflow
13     TIMSK2 = 1 << TOIE2;
14
15     // enable global ir
dieciséis sei();
17
18     while (1)
19     {
20         // Main loop
21     }
22 }

```

- Configura el bit de dirección del pin PB5, escribe el bit DDB5 a 1 lógico en el registro DDRB;
- Establece el preescalador en 1024 configurando los bits CS20, CS21 y CS22 en el registro TCCR2B;
- Borra el indicador de desbordamiento: TOV2 se borra escribiendo un uno lógico en el indicador.
- Habilita interrupciones globales llamando a `sei();`

**b** Habilite la interrupción del temporizador de desbordamiento;

Agregue la función ISR después del ciclo principal para ejecutar el evento LED parpadeante:



```

1  ISR (TIMER0_COMPA_vect)  ?
2  {
3      counterTimer2++;
4      if(counterTimer2 == MyTimerConstant)
5      {
6          counterTimer2 = 0;
7          PORTB ^= 1 << PORTB5;
8      }

```

Configure el registro PORTB para que parpadee el LED



MyTimer2Constant es opcional. Esta constante se puede cambiar para modificar el tiempo que tarda el LED en parpadear.