
**Universidad Autónoma de Baja California**
**FAC. DE CS. QUIM. E INGENIERIA**
**INGENIERIA EN COMPUTACION**

PRACTICA 9

# Laboratorio de: Microprocesadores y microcontradores

Equipo:

López Madrigal Leonardo

Maestro:
García López Jesús Adán

Tijuana, B. C. 1 Mayo, 2017

**Uso de Temporizadores/Contadores del uC ATmega1280**

**Objetivo:** Mediante esta práctica el alumno aprenderá la programación y uso básico del Temporizador 0 y 2 del microcontrolador ATmega1280.

**Material:**
- Computadora Personal (con AVR Studio)
- Tarjeta T-Juino.
- Programa Terminal.

**Equipo:**
- Computadora Personal con USB, AVRStudio y WinAVR

**Teoría:**
- Programación del Timer 0 del microcontrolador
- Programación del Timer 2 del microcontrolador
(Diagrama, Funcionamiento, Registros de configuración y operación)

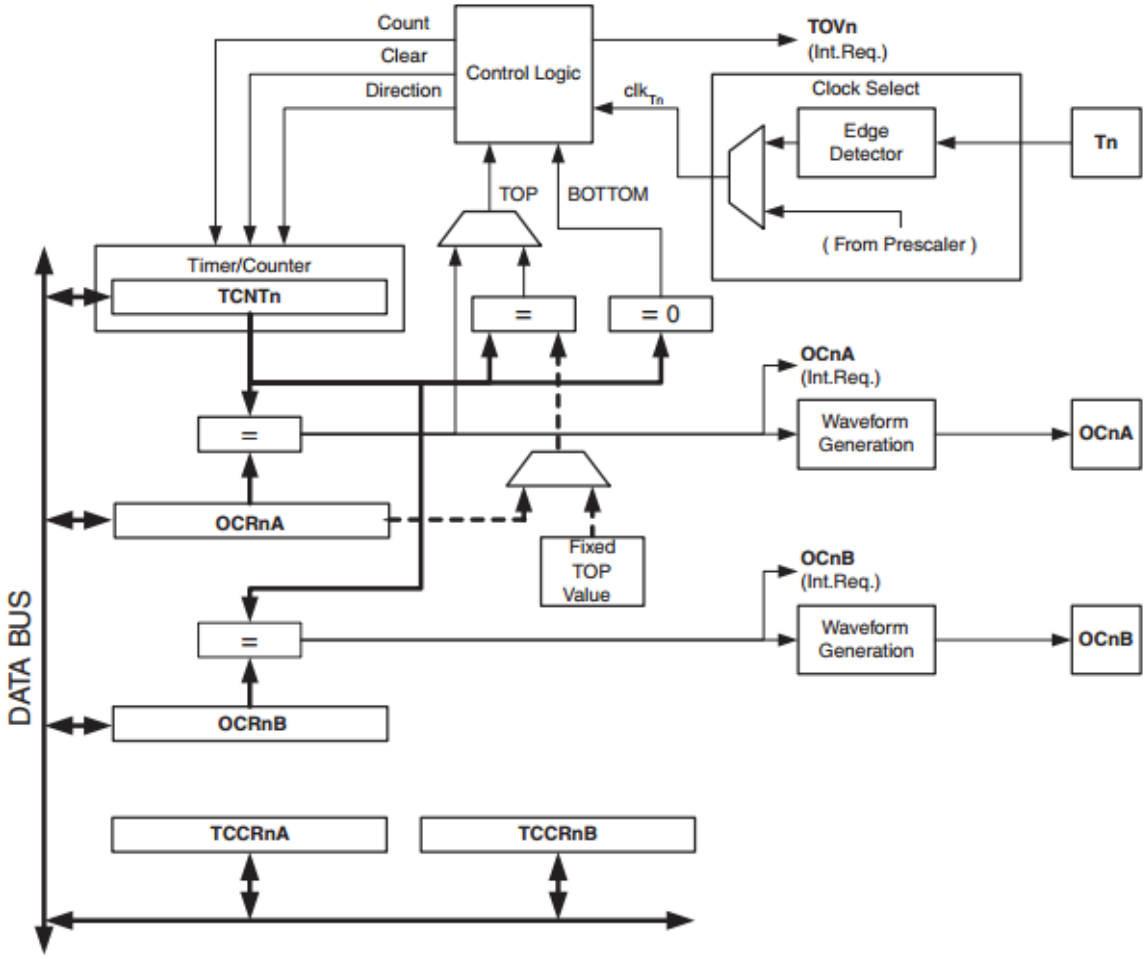
**Teoría**

**Programación del Timer 0 del microcontrolador**

El timer0 AVR es un temporizador contador de 8 bits, el registro donde se guardan los valores del timer0 AVR es el registro temporizador contador representado por TCNT0, cuando es utilizado como temporizador, sus valores aumentaran de uno en uno entre 0 y 255 con cada ciclo de reloj, por ejemplo si el oscilador con el que está funcionando el microcontrolador AVR es de 1MHz, entonces el registro TCNT0 aumentará una unidad en 1 us, si el registro TCNT0 se incrementa en 100 unidades habrán transcurrido 100us.

El registro TCNT0 del timer0 AVR puede ser leído y escrito, puede ser prescalado para que el tiempo en su incremento de valor en una unidad sea mayor, el timer0 AVR cuando alcanza su valor máximo de 255 se reinicia, volviendo a incrementar sus valores de 0 a 255, además cuando llega a su valor máximo y se reinicia puede producir una interrupción, lo cual se decide por programa.

Diagrama.



**Timer0 AVR como *contador***

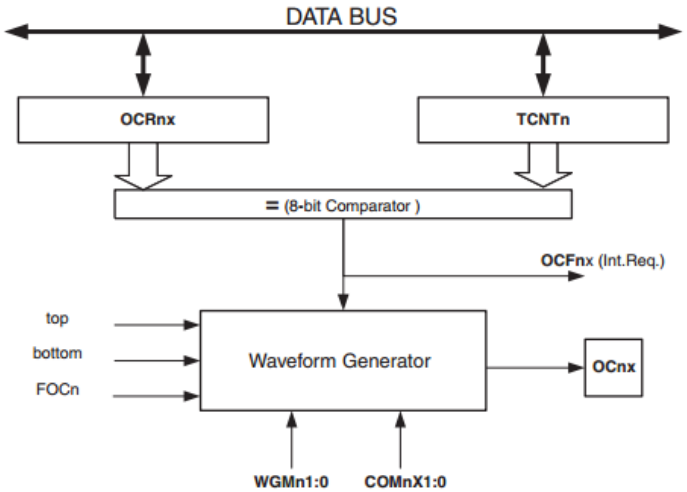
Para el ATmega88 el registro TCNT0 del timer0 AVR está formado por 8 bits por lo que se puede contar desde 0 hasta 255, al utilizar timer0 AVR como contador este no aumentará sus valores o su cuenta con cada ciclo de programa, en este caso el timer0 estará conectado al pin T0 por donde se le hará llegar una señal, la cual al cambiar de estado hará que el timer0 AVR se incremente en una unidad, estos incrementos en su cuenta pueden ser por cada flanco de subida o cada por flanco de bajada de la señal, el flanco a utilizar se elige por programa, por ejemplo su se elige el flanco de subida quiere decir que cada vez que la señal que llega al pin T0 pase de un bajo a un alto o de 0 a 1, el registro TCNT0 aumentará en una unidad, de esta manera se le puede utilizar como contador, cuando la señal pase de un alto a un bajo o de 1 a 0 el registro TCNT0 no aumentará su valor o no se incrementará.

Para programar el timer0 AVR como contador será necesario colocar todos los bits del registro **TCCR0A** a cero, esto es **TCCR0A**=0, en realidad esto no es necesario ya que el registro se inicializa automáticamente a 0, este registro será útil cuando se utilice el timer0 AVR en modo comparación y para la modulación de ancho de pulso PWM.

**Output Compare Unit**

El comparador de 8 bits compara continuamente **TCNT0** con los registros de comparación de salida (**OCR0A** y **OCR0B**). Cuando **TCNT0** es igual a **OCR0A** u **OCR0B**, el comparador señala una coincidencia. Una coincidencia establecerá la comparación de resultados Bandera (**OCF0A** o **OCF0B**) en el siguiente ciclo del reloj del temporizador. Si la interrupción correspondiente está habilitada, la salida el indicador de comparación genera una interrupción de comparación de resultados. La bandera de comparación de resultados se borra se ejecuta la interrupción. Alternativamente, el indicador puede ser borrado por software escribiendo uno lógico en su ubicación de bit de E / S.

El Generador de Forma de Onda utiliza la señal de coincidencia para generar una salida según el modo de funcionamiento establecido por el **WGM02**: 0 bits y comparar el modo de salida (**COM0x1**: 0) bits. Las señales máximas e inferiores son utilizadas por la Generador de forma de onda para manejar los casos especiales de los valores extremos en algunos modos de operación.



**Registros**
**TCCR0A**

**TCCR0A – Timer/Counter Control Register A**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x24 (0x44) | COM0A1 | COM0A0 | COM0B1 | COM0B0 | – | – | WGM01 | WGM00 | TCCR0A |
| Read/Write | R/W | R/W | R/W | R/W | R | R | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

Este registro será útil cuando se utilice el timer0 AVR en modo comparación y para la modulación de ancho de pulso PWM.

**Bits para el modo de generación de la señal**

**Table 16-8.** Waveform Generation Mode Bit Description

| Mode | WGM2 | WGM1 | WGM0 | Timer/Counter Mode of Operation | TOP | Update of OCRx at | TOV Flag Set on[1][2] |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Normal | 0xFF | Immediate | MAX |
| 1 | 0 | 0 | 1 | PWM, Phase Correct | 0xFF | TOP | BOTTOM |
| 2 | 0 | 1 | 0 | CTC | OCRA | Immediate | MAX |
| 3 | 0 | 1 | 1 | Fast PWM | 0xFF | TOP | MAX |
| 4 | 1 | 0 | 0 | Reserved | – | – | – |
| 5 | 1 | 0 | 1 | PWM, Phase Correct | OCRA | TOP | BOTTOM |
| 6 | 1 | 1 | 0 | Reserved | – | – | – |
| 7 | 1 | 1 | 1 | Fast PWM | OCRA | BOTTOM | TOP |

Note: 1. MAX = 0xFF
2. BOTTOM = 0x00

**TCCR0B**

**TCCR0B – Timer/Counter Control Register B**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x25 (0x45) | FOC0A | FOC0B | – | – | WGM02 | CS02 | CS01 | CS00 | TCCR0B |
| Read/Write | W | W | R | R | R/W | R/W | R/W | R/W | |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

El registro TCCR0B es el que permitirá utilizar el timer0 avr como contador.

Para utilizar el timer0 AVR como contador, del registro TCCR0B hay que manipular sus bits 0, 1 y 2 (CS00, CS01 y CS02) los demás bits se pondrán a 0, se tienen dos posibilidades, para que el conteo se realice cuando la señal que llega al pin T0 pase de 0 a 1 o por flanco de subida estos 3 bits se pondrán a 1 esto es CS00=1, CS01=1 y CS02=1; y si el conteo será cuando la señal que llega al pin T0 pase de 1 a 0 o por flanco de bajada solo se pondrán a 1 los bits 1 y 2, mientras el bit0 se deja a 0 esto es CS00=0, CS01=1 y CS02=1.

**Table 16-9.** Clock Select Bit Description

| CS02 | CS01 | CS00 | Description |
|---|---|---|---|
| 0 | 0 | 0 | No clock source (Timer/Counter stopped) |
| 0 | 0 | 1 | $clk_{I/O}$/(No prescaling) |
| 0 | 1 | 0 | $clk_{I/O}$/8 (From prescaler) |
| 0 | 1 | 1 | $clk_{I/O}$/64 (From prescaler) |
| 1 | 0 | 0 | $clk_{I/O}$/256 (From prescaler) |
| 1 | 0 | 1 | $clk_{I/O}$/1024 (From prescaler) |
| 1 | 1 | 0 | External clock source on T0 pin. Clock on falling edge |
| 1 | 1 | 1 | External clock source on T0 pin. Clock on rising edge |

If external pin modes are used for the Timer/Counter0, transitions on the T0 pin will clock the counter even if the pin is configured as an output. This feature allows software control of the counting.

**Timer0 AVR como temporizador**

Para el ATmega1280/2560 el registro **TCNT0** del timer0 AVR está formado por 8 bits por lo que se sus valores pueden ser desde 0 hasta 255, al utilizar timer0 AVR como temporizador este aumentará sus valores o su cuenta con cada ciclo de reloj, sus valores aumentaran de uno en uno entre 0 y 255, por ejemplo si el oscilador con el que está funcionando el microcontrolador AVR es de 1MHz, entonces el registro TCNT0 aumentará una unidad en 1 us, si el registro TCNT0 se incrementa en 27 unidades habrán transcurrido 27us.

Normalmente el registro del timer0 **TCNT0** irá aumentando su conteo con cada ciclo de reloj del microcontrolador, si se usa por ejemplo una frecuencia de trabajo (lo que se conoce como **FCPU**) de 1Mhz entonces el registro TCNT0 aumentará en una unidad cada microsegundo, y como este registro es de 8 bits este aumentará desde 0, o desde algún valor que se le ponga como valor inicial al registro TCNT0, hasta un máximo de 255, por ejemplo si va desde 0 hasta 255 habrán transcurrido 255us luego volverá a 0 pero en esa vuelta a 0 transcurre 1us mas, por lo cual en ir de 0 y volver a 0 transcurren 256us; no siempre se utiliza una FCPU de 1Mhz esto puede variar, por lo que como consecuencia variará el tiempo que trascurre para que el registro TCNT0 aumente su valor.

Para programar el timer0 AVR como temporizador será necesario colocar todos los bits del registro **TCCR0A** a cero, esto es **TCCR0A=0,** en realidad esto no es necesario ya que el registro se inicializa automáticamente a 0, este registro será útil cuando se utilice el timer0 AVR en modo comparación y para la modulación de ancho de pulso PWM.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x24 (0x44) | COM0A1 | COM0A0 | COM0B1 | COM0B0 | – | – | WGM01 | WGM00 | TCCR0A |
| Read/write | R/W | R/W | R/W | R/W | R | R | R/W | R/W | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

El registro **TCCR0B** es el que permitirá utilizar el timer0 avr como temporizador:

**El timer0 AVR cuenta con lo que se conoce como el prescaler** esto hace que la frecuencia de trabajo FCPU se divida por este prescaler, con lo que se logra que el timer0 AVR tarde un poco mas en aumentar su valor en una unidad; el prescaler puede tomar el valor de 1, 8, 64, 256 o 1024; estos valores se eligen programando los bits 0, 1 y 2 del registro **TCCR0B**, los bits 7 a 3 en este caso se pondrán a 0.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| 0x25 (0x45) | FOC0A | FOC0B | – | – | WGM02 | CS02 | CS01 | CS00 | TCCR0B |
| Read/write | W | W | R | R | R/W | R/W | R/W | R/W | |
| Initial value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |

En la siguiente tabla se ve los valores que hay que dar a estos bits para obtener los diferentes valores de los prescaler para el temporizador timer0 AVR, observar que cuando estos bits son todos iguales a 0 el timer0 AVR está deshabilitado.

| CS02 | CS01 | CS00 | Description |
|---|---|---|---|
| 0 | 0 | 0 | No clock source (timer/counter stopped) |
| 0 | 0 | 1 | $clk_{I/O}$/(no prescaling) |
| 0 | 1 | 0 | $clk_{I/O}$/8 (from prescaler) |
| 0 | 1 | 1 | $clk_{I/O}$/64 (from prescaler) |
| 1 | 0 | 0 | $clk_{I/O}$/256 (from prescaler) |
| 1 | 0 | 1 | $clk_{I/O}$/1024 (from prescaler) |
| 1 | 1 | 0 | External clock source on T0 pin. Clock on falling edge. |
| 1 | 1 | 1 | External clock source on T0 pin. Clock on rising edge. |

Si el prescalador elegido es por ejemplo de 8 y la FCPU=1Mhz, entonces la frecuencia de trabajo del timer0 AVR **Ftemp** será de **Ftemp=FCPU/prescaler**, el tiempo que tardará ahora el temporizador en aumentar una unidad será la inversa de este valor lo cual se conoce como periodo **Ttemp=prescaler/FCPU** por lo que en este caso Ttemp=(8)/(1Mz) de donde Ttemp=8us, esto quiere decir que ahora el timer0 AVR aumentará en una unidad cada 8us y en este caso entonces el registro **TCNT0** para ir de 0 a 255 tardará 255*Ttemp=255*8us=2040us=2,040ms; la misma idea es para los demás prescaler.

**Timer2**

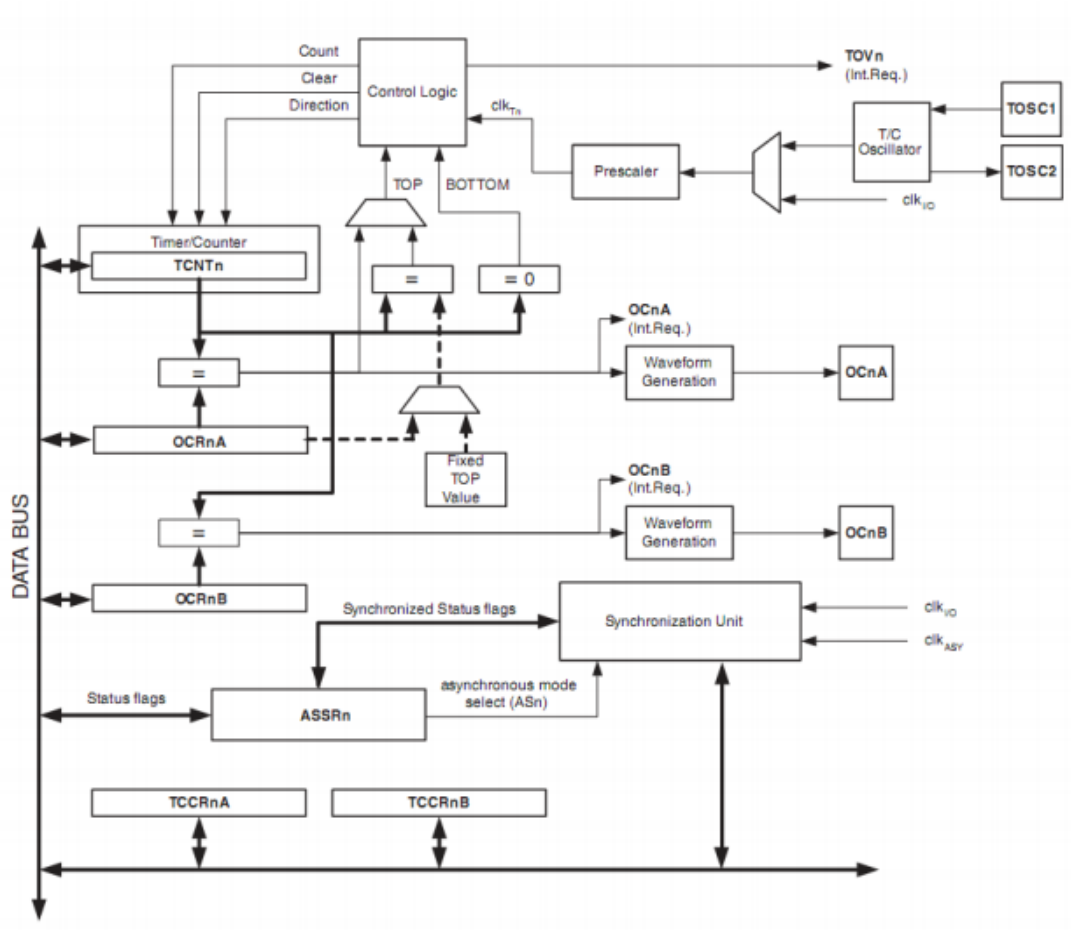
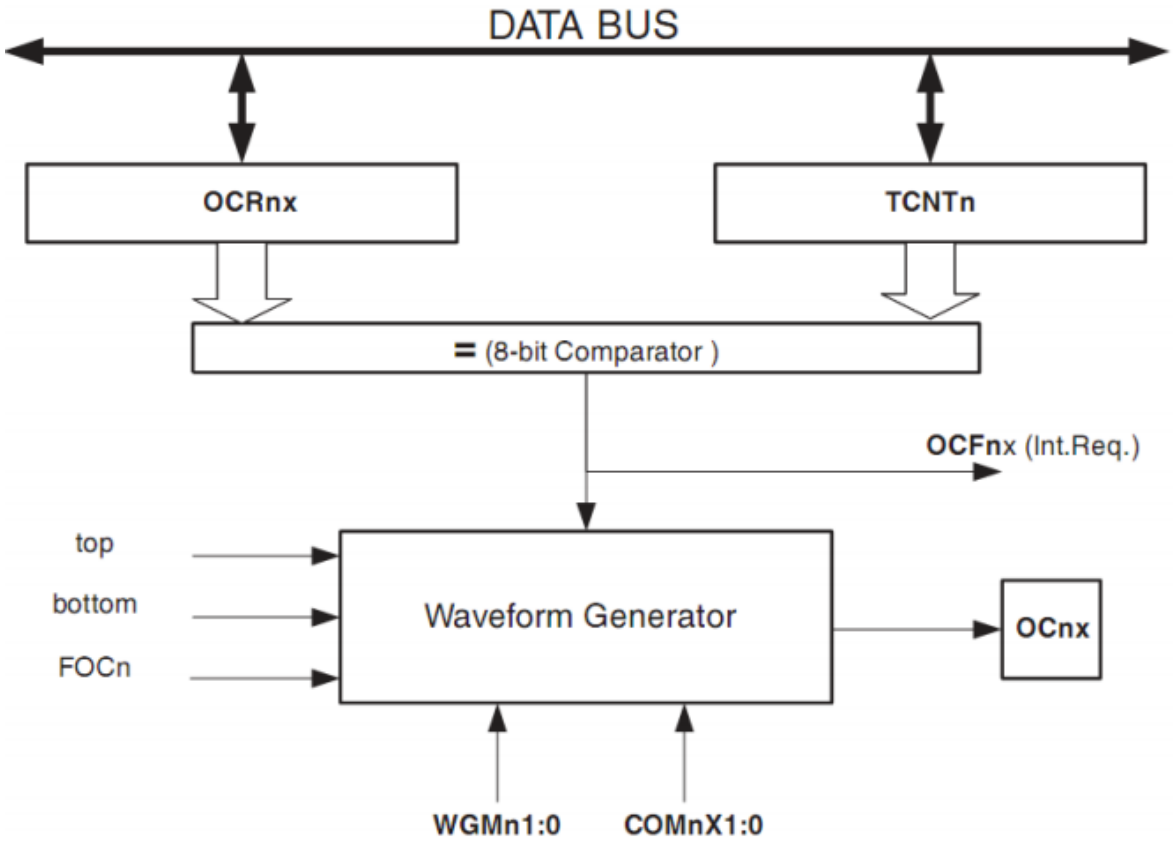

Count
Clear
Direction
Control Logic
TOVn
(Int.Req.)
clk_Tn
TOSC1
T/C
Oscillator
TOSC2
Prescaler
clk_I/O
TOP
BOTTOM
Timer/Counter
TCNTn
=
= 0
OCnA
(Int.Req.)
Waveform
Generation
OCnA
OCRnA
Fixed
TOP
Value
OCnB
(Int.Req.)
Waveform
Generation
OCnB
OCRnB
DATA BUS
Synchronized Status flags
Synchronization Unit
clk_I/O
clk_ASY
Status flags
ASSRn
asynchronous mode
select (ASn)
TCCRnA
TCCRnB

**Diagrama de Bloques de la comparación**



**Registros**

**Waveform Generator Clock**

| Mode | WGM2 | WGM1 | WGM0 | Timer/Counter Mode of Operation | TOP | Update of OCRx at | TOV Flag Set on[1][2] |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Normal | 0xFF | Immediate | MAX |
| 1 | 0 | 0 | 1 | PWM, Phase Correct | 0xFF | TOP | BOTTOM |
| 2 | 0 | 1 | 0 | CTC | OCRA | Immediate | MAX |
| 3 | 0 | 1 | 1 | Fast PWM | 0xFF | BOTTOM | MAX |
| 4 | 1 | 0 | 0 | Reserved | – | – | – |
| 5 | 1 | 0 | 1 | PWM, Phase Correct | OCRA | TOP | BOTTOM |
| 6 | 1 | 1 | 0 | Reserved | – | – | – |
| 7 | 1 | 1 | 1 | Fast PWM | OCRA | BOTTOM | TOP |

Notes: 1. MAX= 0xFF
2. BOTTOM= 0x00

1. Disable the Timer/Counter2 interrupts by clearing OCIE2 and TOIE2.
2. Select clock source by setting AS2 as appropriate.
3. Write new values to TCNT2, OCR2, and TCCR2.
4. To switch to asynchronous operation: Wait for TCN2UB, OCR2UB, and TCR2UB.
5. Clear the Timer/Counter2 Interrupt Flags.
6. Enable interrupts, if needed.

## Registros

### TCNT2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (0xB2) | TCNT2[7:0] | | | | | | | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### OCR2A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (0xB3) | OCR2A[7:0] | | | | | | | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

### OCR2B

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (0xB4) | OCR2B[7:0] | | | | | | | |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## ASSR

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (0xB6) | – | EXCLK | AS2 | TCN2UB | OCR2AUB | OCR2BUB | TCR2AUB | TCR2BUB |
| Read/Write | R | R/W | R/W | R | R | R | R | R |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- **Bit 6 – EXCLK: Enable External Clock Input**
- **Bit 5 – AS2: Asynchronous Timer/Counter2**
- **Bit 4 – TCN2UB: Timer/Counter2 Update Busy**
- **Bit 3 – OCR2AUB: Output Compare Register2 Update Busy**
- **Bit 2 – OCR2BUB: Output Compare Register2 Update Busy**
- **Bit 1 – TCR2AUB: Timer/Counter Control Register2 Update Busy**
- **Bit 0 – TCR2BUB: Timer/Counter Control Register2 Update Busy**

## TCCR2A

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (0xB0) | COM2A1 | COM2A0 | COM2B1 | COM2B0 | – | – | WGM21 | WGM20 |
| Read/Write | R/W | R/W | R/W | R/W | R | R | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- **Bits 7:6 – COM2A1:0: Compare Match Output A Mode**
- **Bits 5:4 – COM2B1:0: Compare Match Output B Mode**
- **Bits 3:2 – Res: Reserved Bits**
- **Bits 1:0 – WGM21:0: Waveform Generation Mode**

## TCCR2B

| (0xB1) | FOC2A | FOC2B | – | – | WGM22 | CS22 | CS21 | CS20 |
|---|---|---|---|---|---|---|---|---|
| Read/Write | W | W | R | R | R/W | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- **Bit 7 – FOC2A: Force Output Compare A**
- **Bit 6 – FOC2B: Force Output Compare B**
- **Bits 5:4 – Res: Reserved Bits**
- **Bit 3 – WGM22: Waveform Generation Mode**
- **Bit 2:0 – CS22:0: Clock Select / Prescaler**

## TIFR2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0x17 (0x37) | – | – | – | – | – | OCF2B | OCF2A | TOV2 |
| Read/Write | R | R | R | R | R | R/W | R/W | R/W |
| Initial Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- **Bit 2 – OCF2B: Output Compare Flag 2 B**
- **Bit 1 – OCF2A: Output Compare Flag 2 A**
- **Bit 0 – TOV2: Timer/Counter2 Overflow Flag**

## TIMSK2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (0x70) | – | – | – | – | – | OCIE2B | OCIE2A | TOIE2 |
| Read/Write | R | R | R | R | R | R/W | R/W | R/W |

- **Bit 2 – OCIE2B: Timer/Counter2 Output Compare Match B Interrupt Enable**
- **Bit 1 – OCIE2A: Timer/Counter2 Output Compare Match A Interrupt Enable**
- **Bit 0 – TOIE2: Timer/Counter2 Overflow Interrupt Enable**

### Como programar el Timer2

When Timer/Counter2 operates asynchronously, some considerations must be taken.

- Warning: When switching between asynchronous and synchronous clocking of Timer/Counter2, the Timer Registers TCNT2, OCR2x, and TCCR2x might be corrupted. A safe procedure for switching clock source is:
  a. Disable the Timer/Counter2 interrupts by clearing OCIE2x and TOIE2.
  b. Select clock source by setting AS2 as appropriate.
  c. Write new values to TCNT2, OCR2x, and TCCR2x.
  d. To switch to asynchronous operation: Wait for TCN2UB, OCR2xUB, and TCR2xUB.
  e. Clear the Timer/Counter2 Interrupt Flags.
  f. Enable interrupts, if needed.

```
void Ini_FiveSecTimer( void ){
                    /*  Inicializar Timer2                                  */
  TIMSK2 ? <-- 0x??;  /*Desactivar interrupciones del Timer2               */
    ASSR ? <-- 0x??;  /*Seleccionar modo asincrono para  Xtal 32.768KHz */
  TCCR2A ? <-- 0x??;  /*Inicializa Timer en modo 2  (CTC-mode)            */
  TCCR2B ? <-- 0x??;  /*Seleccionar Prescaler a: 1024                     */

   TCNT2 ? <-- 0x??;  /*Borrar Timer2                                     */
   OCR2A ? <-- 0x??;  /*valor para generar sobreflujo cada 5 segundos     */
    ASSR ? --> 0x??;  /*Esperar por TCN2UB, OCR2AUB y TCR2AUB             */

   TIFR2 ? <-- 0x??;  /*Borrar banderas de sobreflujo y comparacion       */
   TIMSK2? <-- 0x??;  /*Habilitar interrupcion por Output Compare A       */
    sei();           /*Habilitar interrupciones (global)                  */
}
```

**Conclusiones**

Esta práctica esta complicada debido a que tuve que realizar 3 timers uno fue con el Timer0 y los otros dos con el Timer2, el primero era hacer un reloj el cual estaba programado en el modo CTC donde se hace la comparación entre OCR0A y TCNT0, y al hacer match se iba a la interrupción, OCR0A está definido por una ecuación y su limitante es que solo le cabe un dato de un byte de 0 a 255, para lograr a ese rango nos ayuda un pres-calador el cual esta en múltiplos de 8 hasta 1024, y se logro visualizar el contador en el MTTY, para la segunda parte se hizo prácticamente lo mismo solo que con el Timer2 el cual usa una fuente de oscilación especial de 32768Hz y con esa frecuencia se hizo lo mismo solo cambiando la configuración y otros factores, para la tercera parte la única modificación fue que solo en el Timer2_Ini() contenía un parámetro llamado baseT que es la base de tiempo a la que se hiba a generar el desface de tiempo en el MTTY, también se realizo un autobaudRate el cual genera un UBBR0 al presionar una tecla del teclado que este en el rango de 8,000 a 20,000 bauds.

**Bibliografía**

Atmega1280 Datasheet

**http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf**