Arduino timer – Interrupciones con el Timer2

Arduino timer – Introducción

La función Arduino timer no esta implementada en el lenguaje de forma estándar. Un Timer genera interrupciones para ejecutar funciones sin involucrar al procesador. Por lo general se utilizan librerías externas para poder implementar en Arduino timer y no existe una librería oficial. Esto se debe a las funciones: millis(), micros(), delay(), Servo y Tone que usan a los timers del microcontrolador.

Por ejemplo, el timer0 de 8 bits del microcontrolador ATMEGA328p que incluye un Arduino UNO R3, se utiliza para las funciones de tiempo anteriores.Está es la principal razón por la cual no existe una biblioteca oficial para el manejo de funciones por interrupciones.

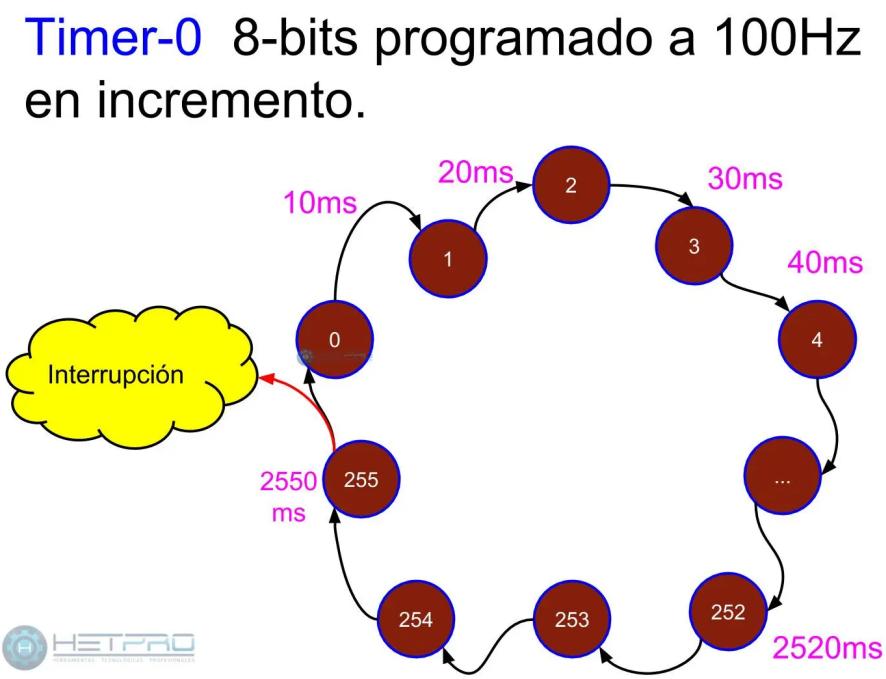
Arduino timer – qué son los timers

Un Timer es un módulo interno de un microcontrolador. Además el timer puede generar una señal periódica a una frecuencia que puede ser configurada. La función principal del timer es contar automáticamente a la velocidad de su frecuencia
ada. Por ejemplo, sus principales características son los bits que puede

El timer-0 de Arduino es de 8 bits, si se configura a una frecuencia de 100Hz, esto es un periodo de T = 1/100Hz = 10mS, le llevaría contar, automáticamente de 0 a 255 (8-bit) un tiempo de 255*10mS = 2.55s. Cuando en Arduino, el Timer termina su cuenta, esté genera una interrupción. Esto indicaría que ha la cuenta ha terminado. Dicha interrupción puede avisar o notificar al procesador para ejecutar alguna función específica. Por ejemplo,nos permite medir el tiempo transcurrido sin el uso de retardos de tiempo que llegan a detener el funcionamiento del procesador.

En Arduino así como en TODOS los microcontroladores, los timers funcionan de forma independiente al procesador. Por lo tanto podemos hacer otros procesos mientras esperamos a que el modulo nos avise que ha terminado su cuenta. Esto nos permite usar, de forma más eficiente, el tiempo del microprocesaror de Arduino.

En el Arduino UNO R3, existen 3 Timers internos. El primer timer, el timer-0 es de 8 bits, el Timer-1 es de 16 bits y el Timer-2 es de 8 bits. Es decir, que pueden contar de 0 a 255 (8-bit) o de 0 a 65535 (16 bits).



Como usar un Timer en Arduino

En Arduino, los timers se usan para el manejo de las funciones de tiempo, para el manejo de servos y para generar señales periódicas (PWM). Por lo tanto tenemos esta restricción si queremos usar timers. Si se usa el Timer0, estaríamos alterando a las funciones: micros(), milis() y delay(). Por otro lado si usamos el Timer-1 (16-bit), no podríamos hacer uso de las funciones usadas en el manejo de servos. Finalmente para usar un Timer-2 (8-bit) ya no podremos usar a la función tone().

Existen **dos formas** para usar un Timer en Arduino.

1. Usando cualquiera de los **Timers** antes mencionados, programando sus registros de configuración y crear nuestras propias funciones para agregar interrupciones.
2. La segunda opción es usar a la función millis() o micros() para generar un «timer virtual». Existen muchas bibliotecas que utilizan esta forma de usar un timer.
    1. Propiamente hablando, esta metodología no utilizaría las ventajas de los timers. Esto dado que solo se estaría comparando el tiempo transcurrido de forma que el procesador sigue manteniéndose ocupado.

Usar el Timer-2 con registros en Arduino

En el siguiente ejemplo, usaremos el Timer-2, el cual es de 8-bits, para activar un LED cada 1.63s sin usar retardos. Recordemos que usar un retardo significa que el procesador no puede hacer otra tarea. De hecho el uso de timers es el principio con el cual operan los sistemas operativos, asignando tiempos mediante semáforos de procesos para que estos puedan usar el tiempo del procesador.

Arduino Timer – Uso Paso #1:

Deshabilitar a las interrupciones mediante la bandera de «interrupciones globales». Para desactivar a las interrupciones globales se tiene que modificar el registro SREG. Este registro tiene 8 bits, es decir 8 banderas. La Figura-2, muestra una imagen con tales banderas.

En el programa de Arduino, en la función setup se agrega la siguiente línea:

- SREG = (SREG & 0b01111111); //Esta instrucción deshabilita a las interrupciones globales sin modificar el resto de las interrupciones.

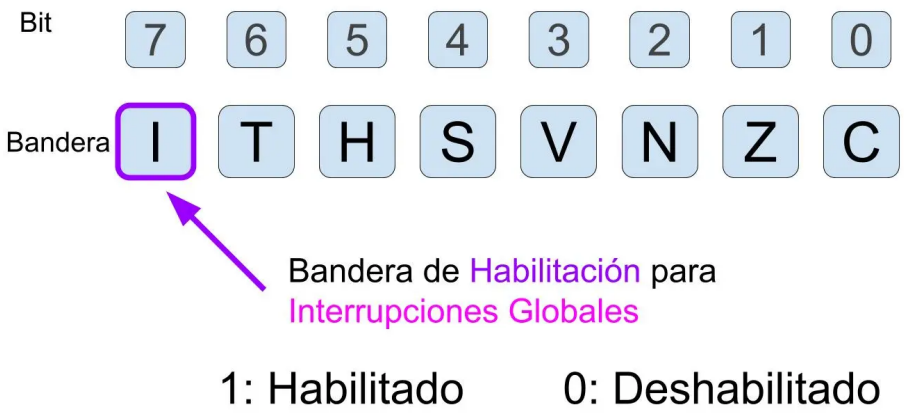


Figura-2. Banderas del registro SREG, el bit7 permite habilitar o deshabilitar el modulo.

Categorías

Elegir la categoría

Dado que el Timer actúa como una maquina de estados automática cuya función es contar a la velocidad configurada, entonces se limpia el contenido del registro que guarda dicha cuenta. Recordemos que el Timer-2 es de 8 bits, por lo tanto sólo es necesario borrar un registro. Para el Timer-1, dado que es de 16 bits, se tienen que limpiar 2 registros. En realidad este paso no seria necesario, sólo en el caso de que se quiera tener un tiempo muy exacto y por lo tanto se le tenga que pre-cargar con un valor inicial. Esto para que la cuenta no comience en cero sino en un valor distinto, lo cual desbordaría al timer más rápido.

- TCNT2 = 0; //Limpiar el registro que guarda la cuenta del Timer-2.

Arduino Timer – Paso # 3:

El siguiente paso será habilitar la interrupción por desbordamiento del Timer-2. Esto significa que el modulo interno, podrá generar una interrupción para el procesador cuando la cuente pase de 255 a 0. En realidad la interrupción se genera cuando el valor del registro TCNT2 es igual a cero. Cuando la interrupción se ejecuta en la función dada, no es necesario limpiar el registro, dado que se desborda automaticamente.

- TIMSK2 =TIMSK2|0b00000001; //Habilitación de la bandera 0 del registro que habilita la interrupción por sobre flujo o desbordamiento del TIMER2.

Arduino Timer – Paso # 3:

Ya que tenemos habilitado el Timer-2, inicializado el registro que guarda la cuenta, el siguiente paso es configurar la frecuencia a la que el timer comenzara a contar. Apartir de que habilitamos una frecuencia el timer comenzara a contar en forma automática sin necesidad de usar el procesador. La frecuencia que usa el timer es el equivalente a la frecuencia del oscilador dividido entre 2. Para un cristal de 16Mhz como el que trae el arduino, el oscilador para el Timer-2 seria de 8Mhz, esto lo tome de la pagina 196 de la hoja de datos [1]. Finalmente este resultado lo valide con un osciloscopio.

Para configurar la frecuencia, dado que ya conocemos que el clkT2 es de 8Mhz, procedemos a establecer una prescala. Esta prescala es controlada por las banderas: CS20, CS21 y CS22. Estas banderas se encuentran en el registro TCCR2B, y son las banderas 0, 1 y 2 respectivamente. A continuación se muestra un ejemplo de como configurarlas en Arduino:

En esta pagina encontraras tutoriales en programación y electrónica.

Tratamos temas de tecnología básica, media y avanzada. Este blog es creado y mantenido por HETPRO.

Aquí encontraras los tutoriales de temas como Arduino, Raspberry Pi, ARM, Beaglebone, PCBs, C/C++, entre otros. Si quieres un tutorial en especial déjanos un comentario, quizás nos pongamos a realizarlo.

Somos personas interesadas en la tecnología con experiencia en la docencia. Nos gusta la filosofía del Software y Hardware libre. Nuestra misión es compartir nuestra experiencia con la mayoría de las personas posibles.

Si te gustan nuestros tutoriales, puedes dejarnos un like y/o compartirlo en tus redes sociales.

- TCCR2B = 0b00000001; // ft2 = 8 Mhz
- TCCR2B = 0b00000010; // ft2 = 1Mhz
- TCCR2B = 0b00000011; //ft2 = 250Khz
- TCCR2B = 0b00000100; // ft2 = 125Khz
- TCCR2B = 0b00000101; // ft2 = 62500 Hz
- TCCR2B = 0b00000110; //ft2 = 31250 Hz
- TCCR2B = 0b00000111; // ft2 = 7812.5 Hz

CS22	CS21	CS20	Función	fT2, clk=16Mhz
0	0	0	Timer2 parado	fT2 = 0 Hz
0	x	1	clkT2 / 1	fT2 = 8Mhz
0	1	0	clkT2 / 8	fT2 = 1Mhz
0	1	1	clkT2 / 32	fT2 = 250 Khz
1	0	0	clkT2 / 64	fT2 = 125 KHz
1	0	1	clkT2 / 128	fT2 = 62500hz
1	1	0	clkT2 / 256	fT2 = 31250hz
1	1	1	clkT2 / 1024	fT2 = 7812.5hz

Tabla-1. Tabla que indica la preescala que se puede configurar para el Timer-2.

Usar Arduino Timer – Paso # 4:

Una vez establecida la frecuencia del timer-2, se procede a habilitar a las interrupciones globales. Es decir que el modulo, cuando pase de 255 a 0, generara una interrupción. Para el lenguaje Arduino, activar a las interrupciones globales se hace con la siguiente instrucción:

- SREG = (SREG & 0b01111111) | 0b10000000; //Habilitar interrupciones

Usar Arduino Timer – Paso # 5:

Etiquetas

7 Segmentos
ADC
Amplificador
Amplificador operacional
Analogico
analogRead
Arduino
Arduino UNO
Audio
AVR
C#
Colecciones
Contador
corriente
Cuantex
datos

tanto esta función estará abajo de la función **setup** o abajo de la función **loop**.

```
ISR(TIMER2_OVF_vect){

   //Instrucciones AQUI

}
```

Arduino-Timer-Función-ISR-desbordamiento-Timer-2.ino hosted with ❤ by GitHub view raw

Tiempo para el Timer de Arduino

Las instrucciones anteriores se codificaran dentro de la función setup. Pero que pasa con el tiempo?. El tiempo depende de la velocidad a la que el contador incremente la cuenta. Esta velocidad depende de la frecuencia. Para calcular el tiempo haremos uso del periodo T. Por ejemplo, el periodo es el inverso de la frecuencia, esto es:

$$T = 1/F$$

Por lo tanto, si configuramos a FT2 como 8Mhz, esto es: CS22 = 0, CS21 = 0 y CS20 = 1 (Ver tabla anterior). El periodo será de:

T = 1/8000000Hz = 0.125uS

Entonces, dado que el timer-2 cuenta de 0 a 255, este proceso le tomara: 255*0.125uS = 31.875uS.

Si se usará esta configuración, la función ISR(TIMER2_OVF_vect) se ejecutaría en forma autónoma cada 31.875 uS.

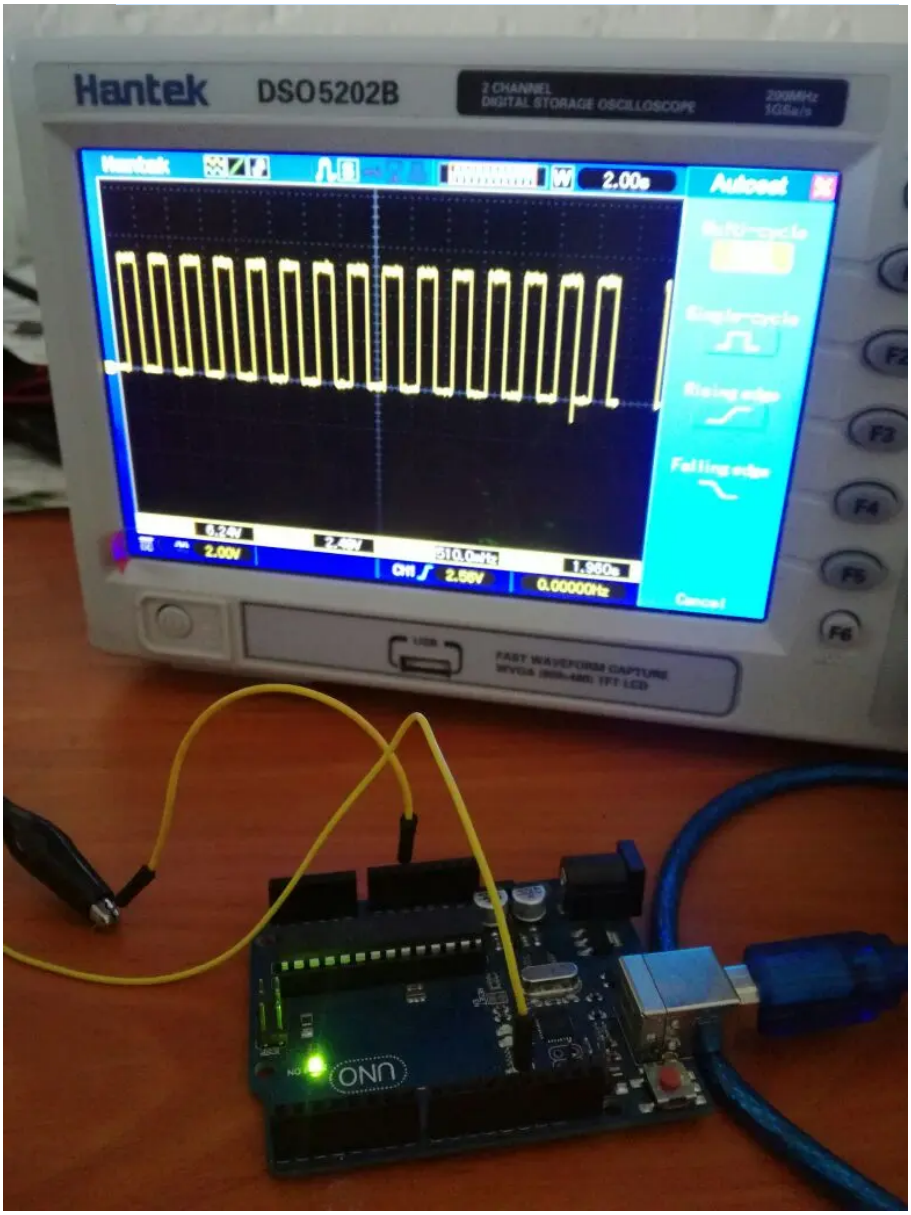
Ejemplo-1. Configurar al Timer-2 a la frecuencia más baja

7812.5Hz. Por lo tanto, su periodo es: T = 0.000128 s , lo que significa que le tomara contar de 0 a 255 un tiempo de: 255*0.000128 = 0.03264 ó 32.64mS.

Ejemplo-2. Configurar el Timer-2 para que genere una interrupción cada segundo.

Para el ejemplo-2, necesitamos la frecuencia más baja, como la del ejemplo1. Esto nos genera una interrupción cada 32.64mS. Por lo tanto, haremos uso de un contador en la función de la interrupción para que cada que se ejecute incremente a dicho contador. Entonces, si el contador llega a la cuenta de 30, estaríamos contado 0.9792 segundos. Recordemos que este tiempo que el contador esta midiendo, es sin usar al procesador tanto. En realidad el procesador solo esta contando cada 32.64mS y durante 30 veces, en el tiempo entre este proceso podría estar ejecutando cualquier otra función.

Si en la función de interrupción hacemos que cada 30 cuentas, encienda un led y en los siguientes 30 cuentas lo apague, entonces tendríamos un led oscilando. Por lo tanto la frecuencia de este led seria de 0.5Hz es decir un ciclo completo cada 2 segundos. En la siguiente imagen se muestran los resultados así como el código del ejemplo.



Código:

```
volatile unsigned int cuenta = 0;
bool ESTADO = false;
void setup() {

 pinMode(13,OUTPUT);



 SREG = (SREG & 0b01111111); //Desabilitar interrupciones
 TIMSK2 = TIMSK2|0b00000001; //Habilita la interrupcion por desbordamiento
 TCCR2B = 0b00000111; //Configura preescala para que FT2 sea de 7812.5Hz
 SREG = (SREG & 0b01111111) | 0b10000000; //Habilitar interrupciones //Desal



}
```

```
}

ISR(TIMER2_OVF_vect){
    cuenta++;
    if(cuenta > 29) {
      digitalWrite(13,ESTADO);
      ESTADO = !ESTADO;
      cuenta=0;
    }


}
```

Arduino-Timer.ino hosted with ❤ by GitHub view raw

Autor:

Dr. Rubén E-Marmolejo.

Profesor Universidad de Guadalajara, México

Referencias:

[1] – http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

‹ Arduino analogWrite – uso y ejemplos

LM35 – El sensor de temperatura más popular ›

10 comentarios en «Arduino timer – Interrupciones con el Timer2»

vladimir

septiembre 7, 2018 a las 7:19 pm

hola Dr. Rubén E-Marmolejo. soy vladimir mella tecnico electronico de chile, si yo quisiera hacer un generador de onda cada 1us como deberia usar los timer?

enero 30, 2019 a las 3:52 pm

Hola Rubén,
Me parece excelente tu explicación sobre el uso de los Timers, solo me queda una duda, cómo mencionabas en el paso #2, tendrías que limpiar el timer TCNT2 = 0; para poder tener un control exacto y preciso para ciertas aplicaciones. Mi pregunta es, ¿ En que momento arranca a contar el timer?, es cuando invocas la instrucción ISR, o desde que recibe alimentación el Microcontrolador, arranca el timer ya que la configuración se hace al inicio del programa. Si esto es así, entonces dentro del programa debemos limpiar el TCNT2=0 para controlar el tiempo exacto?.

Responder

Ruben Estrada
enero 30, 2019 a las 5:46 pm

Hola, a partir de que se habilita la prescala y la interrupción comienza a funcionar el timer, alternativamente puedes desactivar o activar el modulo con la bandera: PRTIM0, si vale 0 esta activo el timer, si vale 1 el timer de desactiva, puedes consultar la información de icho funcionamiento en la pagina 144 de la hoja de datos, saludos.
https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf

Responder

Gerardo Pozo
febrero 2, 2019 a las 1:39 am

Muchas Gracias por contestar a mi pregunta, esta muy claro. Gracias por la información. Saludos!

Responder

Jose luis
marzo 2, 2020 a las 8:56 am

Buenos días.
desearía me aconsejaran sobre el siguiente tema:
tengo asignada a la salida 3 de arduino un atenuador que su control es mediante PWM, dicho atenuador comanda una resistencia de 300W con tensión de 220v.
El problema es el siguiente: cuando aplico valor al pin 3, el coseno de fi va retrocediendo acercándose a cero a medida que voy aumentando los pulsos.
Me ha recomendado que varíe la frecuencia del arduino UNO, pero no se en que sentido.
¿puede valer el ejemplo que tu propuesto arriba ?
si modifico lo valores y estos no son válidos ¿ puede volver luego al estado inicial arduino?
Jose Luis
Jerez de la Frontera

Dr Hector Torres

marzo 3, 2020 a las 4:47 pm

«el coseno de fi va retrocediendo acercándose a cero a medida que voy aumentando los pulsos.» No me queda claro esta frase. Que retrocede? la fase o disminuye la amplitud?

Responder

Enrique

agosto 15, 2020 a las 5:19 am

Buenas, tengo un problema con mi codigo, segui as instrucciones pero cuando pongo instrucciones en el void loop no me realiza las interrupciones y cuando quito el contenido del void loop me genera las instrucciones de la interrupcion pero no me respeta los tiempos.
Debo de hacer que tres leds prendan en difernetes intervalos, eso es lo de menos, lo impportante es la interrupcion interna ya que cada medio segundo se debe de mostrar la lectura de los datos analogicos en el monitor serial de arduino.

Responder

Bryan_nava

abril 11, 2021 a las 5:53 am

Lo que pasa que ahi en ese caso tienes que sincronizar la frecuencia del arduino con las oscilaciones de la red electrica, como hacemos esto?, con un sensor claro, los circuitos de cruce por cero emiten un pulso electrico que puede ser leido por el arduino dependiendo de tus necesidades en este caso te recomiendo que por el risingedge leas los pulsos, una ves esto solucionado puedes sincronizar tus interrupciones para poder emitir un pwm, que dispare un SCR o un Triac, y que permita hacerlo solo durante el tiempo que tu necesitas, obviamente para evitar que el disparo por pwm se haga en una zona al azar de la onda sinusoidal de la red electrica

Responder

Fernando

abril 20, 2021 a las 11:48 pm

En el paso 3 hay una afirmación que está errada:

La frecuencia con la que operan todos los timer no es Fosc/2, sino que simplemente Fosc, la referencia que haces a la pagina 196 del datasheet esta mal interpretada, ya que esa es la frecuencia de conmutación de la onda de salida en el comparador, pero no es la frecuencia del timer.

Es equivalente a decir que en el sketch usaste un toggle pin y por tanto la frecuencia

subida y bajada, pero el timer en realidad esta haciendo el overflow durante los cantos.

La funcion que define la frecuencia de conmutacion es f=16M/(prescaler*timer_counter+1)

donde timer_counter < 256

Saludos

Fernando
Ing. Civil Electronico

Responder

Vicente Danvila Fraile
mayo 16, 2021 a las 7:35 pm

Muchas gracias por la explicación!

Responder

Deja una respuesta

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario *

Nombre *

Correo electrónico *

Publicar el comentario