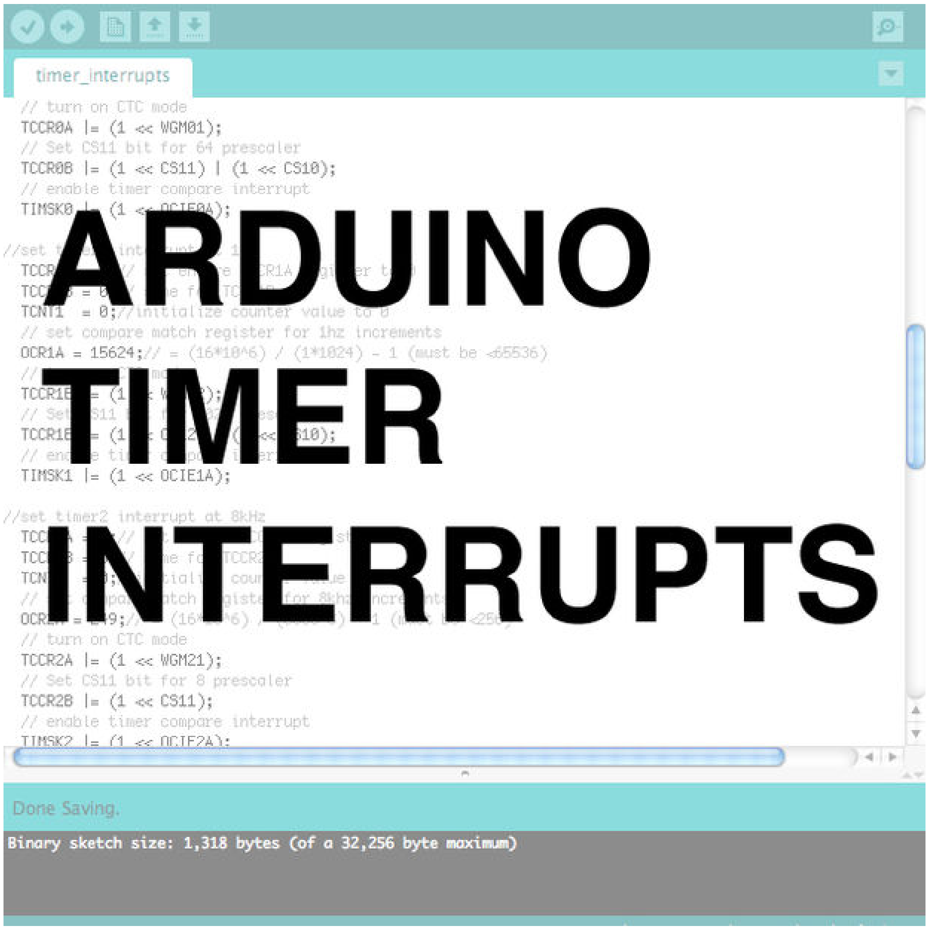
Arduino Timer Interrupts

by amandaghassaei (/member/amandaghassaei/)

Download (/id/Arduino-Timer-Interrupts/?download=pdf) (/id/Arduino-Timer-Interrupts/) 6 Steps

+ Collection I Made it! Favorite



```
  // turn on CTC mode
  TCCR0A |= (1 << WGM01);
  // Set CS11 bit for 64 prescaler
  TCCR0B |= (1 << CS11) | (1 << CS10);
  // enable timer compare interrupt
  TIMSK0 |= (1 << OCIE0A);

//set t    int     1
  TCCR    //    e  CR1A  gi  er t
  TCC  = 0     e f  TC
  TCNT1  = 0;//initialize counter value to 0
  // set compare match register for 1hz increments
  OCR1A = 15624;// = (16*10^6) / (1*1024) - 1 (must be <65536)
  //      CTC mo
  TCCR1   (1   W   );
  // Se  S11        es
  TCCR1   (1   C  2  (  << 10);
  // en  e ti    pa  i  er
  TIMSK1 |= (1 << OCIE1A);

//set timer2 interrupt at 8kHz
  TCC      //    t     CC      st
  TCC      //   me f  TCCR2
  TCN    0;    tial   coun   ue
  //    pa  atch  giste  for 8khz  ncre  nts
  OCR    = 249;//   (16   6)  (    )   1  (   <256)
  // turn on CTC mode
  TCCR2A |= (1 << WGM21);
  // Set CS11 bit for 8 prescaler
  TCCR2B |= (1 << CS11);
  // enable timer compare interrupt
  TIMSK2 |= (1 << OCIE2A);
```

```
Done Saving.
Binary sketch size: 1,318 bytes (of a 32,256 byte maximum)
```

(http://cdn.instructables.com/FYT/ZLFY/H3Z3SUFM/FYTZLFYH3Z3SUFM.LARGE.jpg)

Timer interrupts allow you to perform a task at very specifically timed intervals regardless of what else is going on in your code. In this instructable I'll explain how to setup and execute an interrupt in Clear Timer on Compare Match or CTC Mode. Jump straight to step 2 if you are looking for sample code.

Normally when you write an Arduino sketch the Arduino performs all the commands encapsulated in the loop() {} function in the order that they are written, however, it's difficult to time events in the loop(). Some commands take longer than others to execute, some depend on conditional statements (if, while...) and some Arduino library functions (like digitalWrite or analogRead) are made up of many commands. Arduino timer interrupts allow you to momentarily pause the normal sequence of events taking place in the loop() function at

About This Instructable

255,536 views

257 favorites

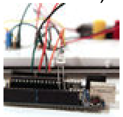
License: (cc) BY-NC-SA

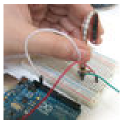
amandaghassaei (/member/amandaghassaei/)
uh-man-duh-guss-eye-dot-com (http://www.amandaghassaei.com)
(/member/amandaghassaei/) Follow 1990

Bio: I'm a developer here at Instructables, I work on the website and iOS app.

More by amandaghassaei

(/id/Sugarcube-MIDI-Controller)


(/id/Beginner-Arduino)


(/id/Arduino-Sensors-and-MIDI)

Tags:

precisely timed intervals, while you execute a separate set of commands. Once these commands are done the Arduino picks up again where it was in the loop().

Interrupts are useful for:

Measuring an incoming signal at equally spaced intervals (constant sampling frequency)
Calculating the time between two events
Sending out a signal of a specific frequency
Periodically checking for incoming serial data
much more...

There are a few ways to do interrupts, for now I'll focus on the type that I find the most useful/flexible, called Clear Timer on Compare Match or CTC Mode. Additionally, in this instructable I'll be writing specifically about the timers to the Arduino Uno (and any other Arduino with ATMEL 328/168... Lilypad, Duemilanove, Diecimila, Nano...). The main ideas presented here apply to the Mega and older boards as well, but the setup is a little different and the table below is specific to ATMEL 328/168.

Step 1: Prescalers and the Compare Match Register

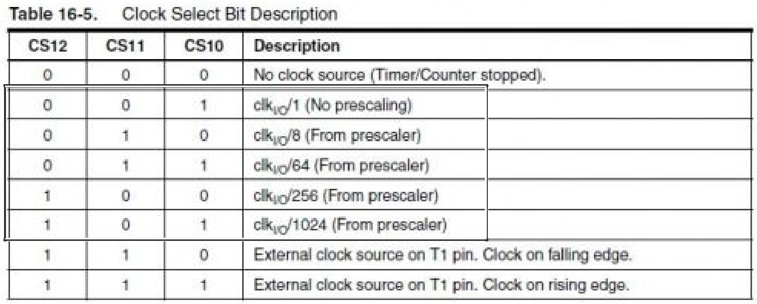
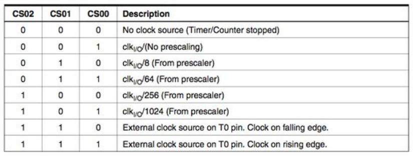


Table 16-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (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 T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

(http://cdn.instructables.com/F3T/TIKL/H3WSA4V7/F3TTIKLH3WSA4V7.LARGE.jpg)



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.

(http://cdn.instructables.com/F8G/35IH/HFD1FIKF/F8G35IHHFD1FIKF.LARGE.jpg)

arduino (/tag/type-id/category-technology/keyword-arduino/)

timer interrupt (/tag/type-id/category-technology/keyword-timer interrupt/)

timer (/tag/type-id/category-technology/keyword-timer/)

interrupt (/tag/type-id/category-technology/keyword-interrupt/)

ctc (/tag/type-id/category-technology/keyword-ctc/)

electronics (/tag/type-id/category-technology/keyword-electronics/)

Recommendations

Robotic hand for the Rubber Hand

Raspberry Pi PirateBox

Twin Raspberry Pi B+ Rack Case

Assemble the case

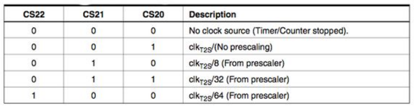
Paint the case

Mount the Raspberry Pi inside the case

Install the USB power cables

Build the LED circuit board

The Uno h d timer2. Each of the timers
has a cour the timer's clock. CTC timer
interrupts a specified value, stored in the
compare n aches this value it will clear
(reset to z , then it will continue to count
up to the c g the compare match value
and setting ts the counter, you can control
the frequer



CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)

(http://cdn.instructables.com/FY3/SNSN/HFD1FIKG/FY3SNSNHFD1FIKG.LARGE.jpg)

The first parameter I'll discuss is the speed at which the timer increments the counter. The Arduino clock runs at 16MHz, this is the fastest speed that the timers can increment their counters. At 16MHz each tick of the counter represents 1/16,000,000 of a second (~63ns), so a counter will take 10/16,000,000 seconds to reach a value of 9 (counters are 0 indexed), and 100/16,000,000 seconds to reach a value of 99.

In many situations, you will find that setting the counter speed to 16MHz is too fast. Timer0 and timer2 are 8 bit timers, meaning they can store a maximum counter value of 255. Timer1 is a 16 bit timer, meaning it can store a maximum counter value of 65535. Once a counter reaches its maximum, it will tick back to zero (this is called overflow). This means at 16MHz, even if we set the compare match register to the max counter value, interrupts will occur every 256/16,000,000 seconds (~16us) for the 8 bit counters, and every 65,536/16,000,000 (~4 ms) seconds for the 16 bit counter. Clearly, this is not very useful if you only want to interrupt once a second.

Instead you can control the speed of the timer counter incrementation by using something called a prescaler. A prescaler dictates the speed of your timer according the the following equation:

(timer speed (Hz)) = (Arduino clock speed (16MHz)) / prescaler

So a 1 prescaler will increment the counter at 16MHz, an 8 prescaler will increment it at 2MHz, a 64 prescaler = 250kHz, and so on. As indicated in the tables above, the prescaler can equal 1, 8, 64, 256, and 1024. (I'll explain the meaning of CS12, CS11, and CS10 in the next step.)

Now you can calculate the interrupt frequency with the following equation:

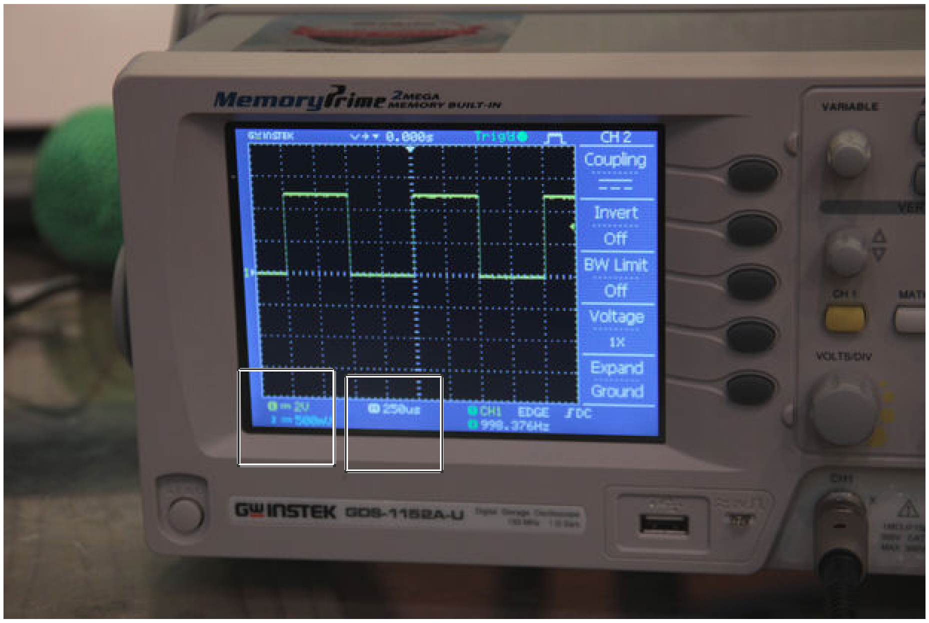
interrupt frequency (Hz) = (Arduino clock speed 16,000,000Hz) / (prescaler * (compare match register + 1))
the +1 is in there because the compare match register is zero indexed

rearranging the equation above, you can solve for the compare match register value that will give your desired interrupt frequency:

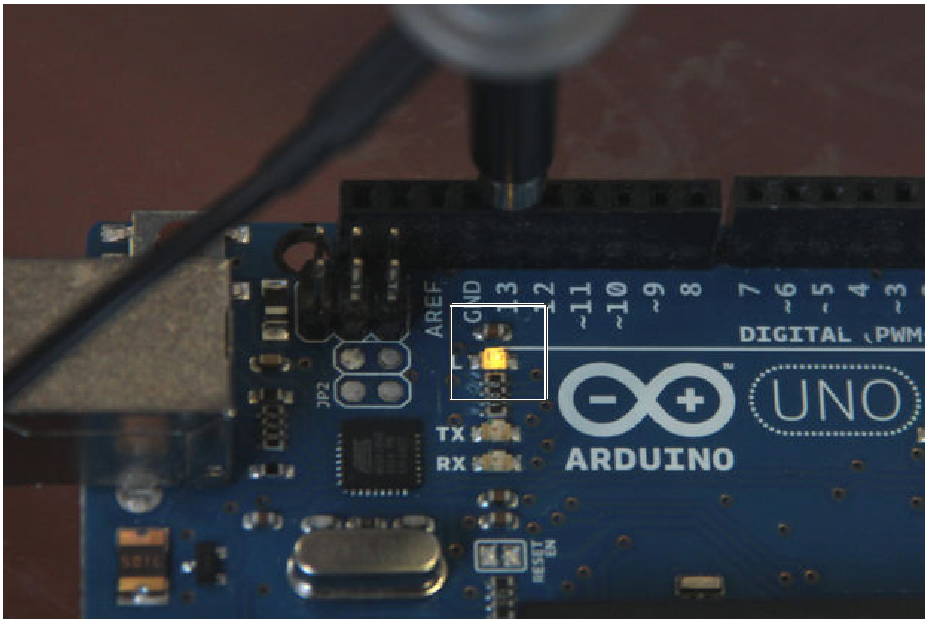
compare match register = [16,000,000Hz/ (prescaler * desired interrupt frequency)] - 1
remember that when you use timers 0 and 2 this number must be less than 256, and less than 65536 for timer1

so if you wanted an interrupt every second (frequency of 1Hz):
compare match register = [16,000,000 / (prescaler * 1)] -1
with a prescaler of 1024 you get:
compare match register = [16,000,000 / (1024 * 1)] -1
= 15,624
since 256 < 15,624 < 65,536, you must use timer1 for this interrupt.

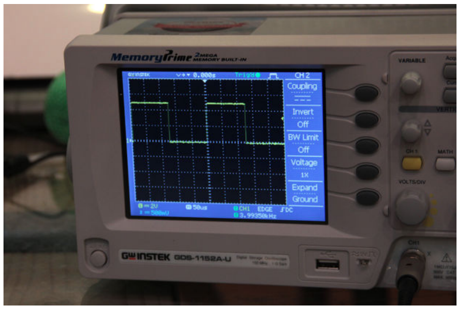
Step 2: Structuring Timer Interrupts



(http://cdn.instructables.com/FBX/1C17/H3Z3SUCT/FBX1C17H3Z3SUCT.LARGE.jpg)



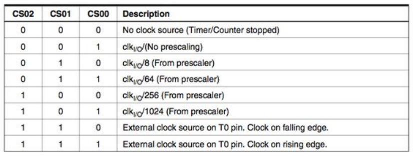
(http://cdn.instructables.com/FK9/L3PY/H3Z3C64Y/FK9L3PYH3Z3C64Y.LARGE.jpg)



(http://cdn.instructables.com/F90/L1JK/H3Z3TWLR/F90L1JKH3Z3TWLR.LARGE.jpg)

Timer setup code is done inside the setup(){} function in an Arduino sketch.

The code involved for setting up timer interrupts is a little daunting to look at, but it's actually the same main chunk of code and change ster to set the correct interrupt frequency.



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.

(http://cdn.instructables.com/FHI/YRBY/HFD189ZY/FHIYRBYHFD189ZY.LARGE.jpg)

The main s this:

```
  // set                                                ments
  OCR1A =                                               (must be <65536)
  // turr
  TCCR1B
  // Set
  TCCR1B |= (1 << CS12) | (1 << CS10);
  // enable timer compare interrupt
  TIMSK1 |= (1 << OCIE1A);

//set timer2 interrupt at 8kHz
  TCCR2A = 0;// set entire TCCR2A register to 0
  TCCR2B = 0;// same for TCCR2B
  TCNT2  = 0;//initialize counter value to 0
  // set compare match register for 8khz increments
  OCR2A = 249;// = (16*10^6) / (8000*8) - 1 (must be <256)
  // turn on CTC mode
  TCCR2A |= (1 << WGM21);
  // Set CS21 bit for 8 prescaler
  TCCR2B |= (1 << CS21);
  // enable timer compare interrupt
  TIMSK2 |= (1 << OCIE2A);


sei();//allow interrupts

}//end setup
```

Notice how the value of OCR#A (the compare match value) changes for each of these timer setups. As explained in the last step, this was calculated according to the following equation:

compare match register = [16,000,000Hz/ (prescaler * desired interrupt frequency)] - 1
remember that when you use timers 0 and 2 this number must be less than 256, and less than 65536 for timer1

Also notice how the setups between the three timers differ slightly in the line which turns on CTC mode:
TCCR0A |= (1 << WGM01);//for timer0
TCCR1B |= (1 << WGM12);//for timer1
TCCR2A |= (1 << WGM21);//for timer2
This follows directly from the datasheet of the ATMEL 328/168.

Finally, notice how the setup for the prescalers follows the tables in the last step (the table for timer 0 is repeated above),
TCCR2B |= (1 << CS22); // Set CS#2 bit for 64 prescaler for timer 2
TCCR1B |= (1 << CS11); // Set CS#1 bit for 8 prescaler for timer 1
TCCR0B |= (1 << CS02) | (1 << CS00); // Set CS#2 and CS#0 bits for 1024 prescaler for timer 0

Notice in the last step that there are different prescaling options for the different timers. For example, timer2 does not have the option of 1024 prescaler.

The commands you want to execute during these timer interrupts are located in the Arduino sketch encapsulated in the following:
ISR(TIMER0_COMPA_vect){ //change the 0 to 1 for timer1 and 2 for timer2
   //interrupt commands here
}

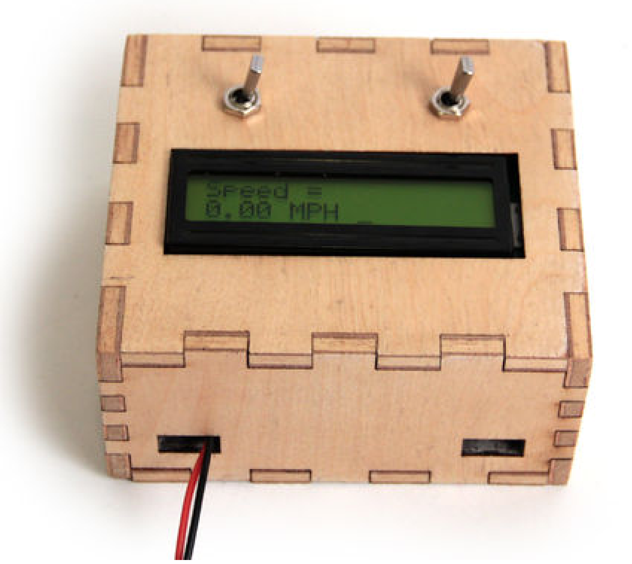
This bit of code should be located outside the setup() and loop() functions. Also, try to keep the interrupt routine as short as possible, especially if you are interrupting at a high frequency. It may even be worth addressing the ports/pins of the ATMEL chip directly instead of using the digitalWrite() and digitalRead() functions. You can find more info about that here (http://www.arduino.cc/en/Reference/PortManipulation).

Example- the following sketch sets up and executes 3 timer interrupts:

```
//timer interrupts
//by Amanda Ghassaei
//June 2012
//http://www.instructables.com/id/Arduino-Timer-Interrupts/

/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3 of the License, or
 * (at your option) any later version.
 *
*/

//timer setup for timer0, timer1, and timer2.
//For arduino uno or any board with ATMEL 328/168.. diecimila, duemilanove, lilypad, nano, mini...

//this code will enable all three arduino timer interrupts.
//timer0 will interrupt at 2kHz
//timer1 will interrupt at 1Hz
//timer2 will interrupt at 8kHz

//storage variables
boolean toggle0 = 0;
boolean toggle1 = 0;
boolean toggle2 = 0;
```

The images above show the outputs from these timer interrupts. Fig 1 shows a square wave oscillating between 0 and 5V at 1kHz (timer0 interrupt), fig 2 shows the LED attached to pin 13 turning on for one second then turning off for one second (timer1 interrupt), fig 3 shows a pulse wave oscillating between 0 and 5V at a frequency of 4khz (timer2 interrupt).

Step 3: Example 1: Bike Speedometer



(http://cdn.instructables.com/FTA/IXO8/H3NN9MP6/FTAIXO8H3NN9MP6.LARGE.jpg)

(http://cdn.instructables.com/FF6/VLYP/H3XN364W/FF6VLYPH3XN364W.LARGE.jpg)

In this example I made an arduino powered bike speedometer (http://www.instructables.com/id/Arduino-Bike-Speedometer/). It works by attaching a magnet to the wheel and measuring the amount of time it takes to pass by a magnetic switch mounted on the frame- the time for one complete rotation of the wheel.

I set timer 1 to interrupt every ms (frequency of 1kHz) to measure the magnetic switch. If the magnet is passing by the switch, the signal from the switch is high and the variable "time" gets set to zero. If the magnet is not near the switch "time" gets incremented by 1. This way "time" is actually just a measurement of the amount of time in milliseconds that has passed since the magnet last passed by the magnetic switch. This info is used later in the code to calculate rpm and mph of the bike.

(http://cdn.instructables.com/FPN/OG5Q/H3O4SDBL/FPNOG5QH3O4SDBL.LARGE.jpg)

Here's the bit of code that sets up timer1 for 1kHz interrupts

```
cli();//stop interrupts
//set timer1 interrupt at 1kHz
TCCR1A = 0;// set entire TCCR1A register to 0
TCCR1B = 0;// same for TCCR1B
TCNT1  = 0;//initialize counter value to 0
// set timer count for 1khz increments
OCR1A = 1999;// = (16*10^6) / (1000*8) - 1
//had to use 16 bit timer1 for this bc 1999>255, but could switch to timers 0 or 2 with larger prescaler
// turn on CTC mode
TCCR1B |= (1 << WGM12);
// Set CS11 bit for 8 prescaler
TCCR1B |= (1 << CS11);
```

```
// enable timer compare interrupt
TIMSK1 |= (1 << OCIE1A);
sei();//allow interrupts
```

Here's the complete code if you want to take a look:

```
//bike speedometer
//by Amanda Ghassaei 2012
//http://www.instructables.com/id/Arduino-Timer-Interrupts/
//http://www.instructables.com/id/Arduino-Timer-Interrupts/

/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 3 of the License, or
 * (at your option) any later version.
 *
*/

//sample calculations
//tire radius ~ 13.5 inches
```

Step 4: Example 2: Serial Communication

(http://cdn.instructables.com/FEO/NR9H/H4AGIFI0/FEONR9HH4AGIFI0.LARGE.jpg)

(http://cdn.instructables.com/FMN/85D5/H4AGJH3W/FMN85D5H4AGJH3W.LARGE.jpg)

This project is a 4x4 backlit button pad. The project connects to my computer via usb, it sends information about the buttons to the computer and receives information about how to light up the LEDs. Here is a video:

For this project, I used timer2 interrupts to periodically check if there was any incoming serial data, read it, and store it in the matrix "ledData[]". If you take a look at the code you will see that the main loop of the sketch is what is actually responsible for using the info in ledData to light up the correct LEDs and checking on the status of the buttons (a function called "shift()"). The interrupt routine is as short as possible- just checking for incoming bytes and storing them appropriately.

Here is the setup for timer2:

```
cli();//stop interrupts
//set timer2 interrupt every 128us
TCCR2A = 0;// set entire TCCR2A register to 0
TCCR2B = 0;// same for TCCR2B
TCNT2  = 0;//initialize counter value to 0
// set compare match register for 7.8khz increments
OCR2A = 255;// = (16*10^6) / (7812.5*8) - 1 (must be <256)
// turn on CTC mode
TCCR2A |= (1 << WGM21);
// Set CS21 bit for 8 prescaler
TCCR2B |= (1 << CS21);
// enable timer compare interrupt
TIMSK2 |= (1 << OCIE2A);
sei();//allow interrupts
```

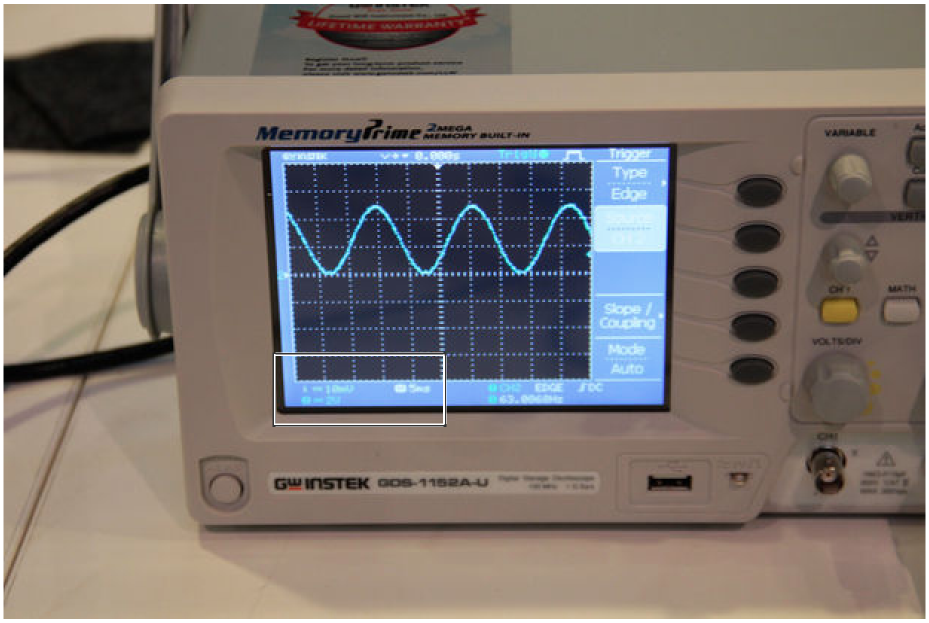
Here's the complete Arduino sketch:

```
//BUTTON TEST w/ 74HC595 and 74HC165 and serial communication
//by Amanda Ghassaei
//June 2012
//http://www.instructables.com/id/Arduino-Timer-Interrupts/

/*
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
*/

//this firmware will send data back and forth with the maxmsp patch "beat slicer"
```

download the MaxMSP (http://cycling74.com/downloads/) patch below (it will run in Max Runtime as well).

beat slicer.zip (/files/orig/FFP/TBHG/H4AGIFKT/FFPTBHGH4AGIFKT.zip)15 KB

Step 5: Example 3: DAC



(http://cdn.instructables.com/FYY/B8JH/H4AGKE5L/FYYB8JHH4AGKE5L.LARGE.jpg)

(http://cdn.instructables.com/FPY/2TJ4/H2WEWMIS/FPY2TJ4H2WEWMIS.LARGE.jpg)

(http://cdn.instructables.com/FAL/P9EQ/H337JN9F/FALP9EQH337JN9F.LARGE.jpg)

(http://cdn.instructables.com/FLI/SAXX/H2WEQJBZ/FLISAXXH2WEQJBZ.LARGE.jpg)

In this project I used a timer interrupt to output a sine wave of a specific frequency from the Arduino. I soldered a simple 8 bit R2R DAC (http://en.wikipedia.org/wiki/Resistor_ladder) to digital pins 0-7. This DAC was constructed from 10k and 20k resistors arranged in a multi-leveled voltage divider (http://en.wikipedia.org/wiki/Voltage_divider). I'll be posting more about the construction of the DAC in another instructable, for now I've included the photo's above.

I connected the output from the DAC up to an oscilloscope. If you need help understanding how to use/read the oscilloscope check out this tutorial (http://www.instructables.com/id/Oscilloscope-How-To/). I loaded the following code onto the Arduino:

(http://cdn.instructables.com/FBB/KP59/H2WEWLS2/FBBKP59H2WEWLS2.LARGE.jpg)

```
//63Hz sine wave
//by Amanda Ghassaei 2012
//http://www.instructables.com/id/Arduino-Timer-Interrupts/

/*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation; either version 3 of the License, or
* (at your option) any later version.
*
*/

//sends 63Hz sine wave to arduino PORTD DAC
float t = 0;
```

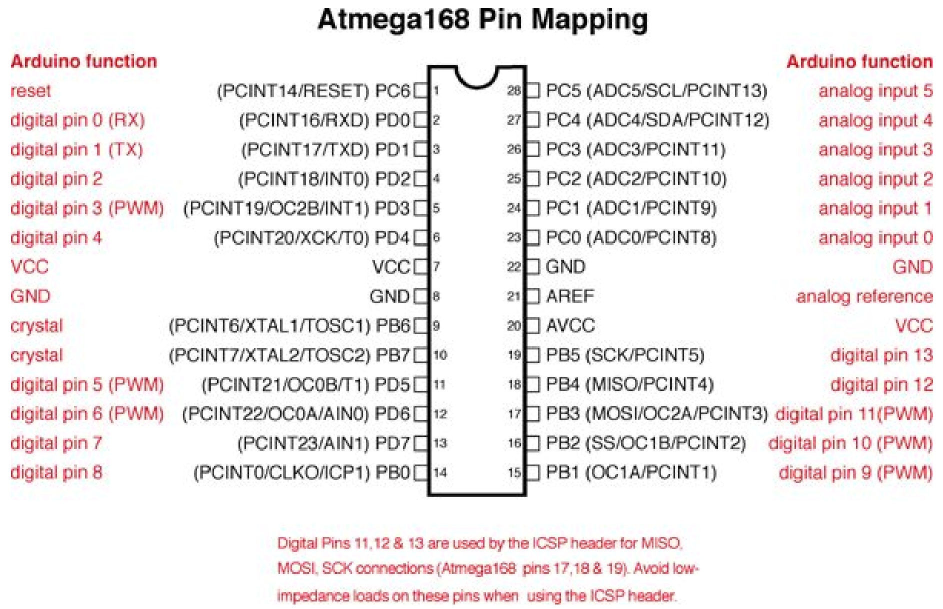
I set up a timer interrupt that increments the variable t at a frequency of 40kHz. Once t reaches 627 it resets back to zero (this happens with a frequency of 40,000/628 = 63Hz). Meanwhile, in the main loop the Arduino sends a value between 0 (00000000 in binary) and 255 (11111111 in binary) to digital pins 0 through 7 (PORTD (http://www.arduino.cc/en/Reference/PortManipulation)). It calculates this value with the following equation:

PORTD=byte(127+127*sin(t/100));

So as t increments from 0 to 627 the sine function moves through one complete cycle. The value sent to PORTD is a sine wave with frequency 63Hz and amplitude 127, oscillating around 127. When this is sent through the 8 bit resistor ladder DAC it outputs an oscillating signal around 2.5V with an amplitude of 2.5V and frequency of 63Hz.

The frequency of the sine wave can be doubled by multiplying the (t/100) term by 2, quadrupled by multiplying by 4, and so on...
Also notice that if you increase the frequency of the timer interrupt too much by decreasing the prescaler or OCR2A the sine wave will not output correctly. This is because the sin() function is computationally expensive, and at high interrupt frequencies it does not have enough time to execute. If you are using high frequency interrupts, instead of performing a computation during the interrupt routine, considering storing values in an array and simply calling these values using some kind of index. You can find an example of that in my arduino waveform generator (http://www.instructables.com/id/Arduino-Waveform-Generator/)- by storing 20,000 values of sin in an array, I was able to output sine waves with a sampling frequency of 100kHz.

Step 6: Timer and Arduino Functions



(http://cdn.instructables.com/FR8/2QNL/H4AFSO5L/FR82QNLH4AFSO5L.LARGE.jpg)

One last thing to note- certain timer setups will actually disable some of the Arduino library functions. Timer0 is used by the functions millis() (http://arduino.cc/en/Reference/millis) and delay() (http://arduino.cc/en/Reference/delay), if you manually set up timer0, these functions will not work correctly.
Additionally, all three timers underwrite the function analogWrite() (http://arduino.cc/en/Reference/analogWrite). Manually setting up a timer will stop analogWrite() from working.

If there is some portion of your code that you don't want interrupted, considering using cli() and sei() to globally disable and enable interrupts (http://arduino.cc/playground/Main/AVR).

You can read more about this on the Arduino website (http://arduino.cc/en/Tutorial/SecretsOfArduinoPWM).

Make Comment

1-40 of 53

Next » (http://www.instructables.com/id/Arduino-Timer-Interrupts/?&sort=ACTIVE&limit=40&offset=40#DISCUSS)

Unidentified Identity (/member/Unidentified+Identity/)

1 month ago

Reply (CC9QEJXHW0FTI8L)

(/member/Unidentified+Identity/)

can we generate square wave with varying pulse width with this interrupt ?

amandaghassaei (/member/amandaghassaei/) (author) Unidentified Identity

28 days ago Reply (CVNDXHSHWZ9VG4V)

yes it's possible, but probably easier to just use the analogWrite command

(/member/amandaghassaei/)

srinivas_arduino (/member/srinivas_arduino/) 1 month ago Reply (C2K8GDGHWK1MAMR)

(/member/srinivas_arduino/)

sir please send program for generating 15000 sample valuesin 10 sec using timer interrupt values adc values from accelometer sensor

srinivas_arduino (/member/srinivas_arduino/) 1 month ago Reply (CUCKA8DHWK1MAMK)

(/member/srinivas_arduino/)

sir please send program for generating 15000 sample values using timer interrupt values adc values from accelometer sensor

AbdulW87 (/member/AbdulW87/) 2 months ago Reply (C78VH5QHV0KX5EQ)

(/member/AbdulW87/)

i am confused about something in the code, the line

// turn on CTC mode

TCCR0A |= (1 << WGM01);

shouldnt it be TCCR0A |= (1<<WGM02); or am i understanding this wrong, i am a beginner at arduino so its still very confusing to me.

O-Zo (/member/O-Zo/) 5 months ago Reply (CLEBPF9HS61BFOL)

(/member/O-Zo/)

Thank you for this tutorial, I'm struggling with a project and this helped me stumble at least to the right direction.

amandaghassaei (/member/amandaghassaei/) (author) O-Zo 4 months ago Reply (CZR9JBAHSBSMB7C)

cool, glad to hear it!

(/member/amandaghassaei/)

k7thakar (/member/k7thakar/) 7 months ago Reply (CDCDUPHHOKSDAUW)

(/member/k7thakar/)

I read many doc about arduino timer and counter but this one most helpful.......thank you for making it.... :)

amandaghassaei (/member/amandaghassaei/) (author) k7thakar 4 months ago Reply (C39XFM6HSBSMB7A)

thanks!

(/member/amandaghassaei/)

cavemen (/member/cavemen/) 6 months ago Reply (CECGXQPHR0AQCTU)

(/member/cavemen/)

65,536/15,624 = 4.19457245264

So the greatest interval we can get from such timer is a little more than 4 sec. or 0.25hz?

I am new to CPL and C++. What does if (toggle1){ } stand for? toggle1 compared to what?

Thanks.

O-Zo (/member/O-Zo/) cavemen 5 months ago Reply (CNABM83HS61BETU)

(/member/O-Zo/)

Toggle1 was declared as "boolean" which means it can have values of 1 (meaning TRUE) or 0 (meaning FALSE). These values are used for "logical" operations, such as in an if statement like you noted. Normally (doing an if statement), if you're comparing variable x to some defined number A, for example A = 5, the expression x > A gives you a boolean value 1 (meaning TRUE), if x is 6 or higher. So, declaring a if expression "if(toggle1){}" is a kind of funny way of saying if toggle 1 is true (1) you're gonna do it. So now, remembering that we defined toggle1 as 0 at first, the first time we encounter if(toggle1){}, it's not gonna get executed since toggle1 is 0. Instead, since we have declared the "else" part, that's the way we're gonna take. At the end of the "else" statement, there's the line toggle1 = 1, which means that the next time we have to choose between the if and else-statements if(toggle1){} is gonna be TRUE, and since that statement ends with toggle1 = 0, we end up switching between these two expression until the very end.

thegoodhen (/member/thegoodhen/) 6 months ago Reply (CM0TK3RHPT950SJ)

(/member/thegoodhen/)

Hello... The code is very useful, however, be aware that the sketch you posted doesn't seem to ever get into the main loop!

dmeister2 (/member/dmeister2/) 9 months ago Reply (CKO98E0HN7NUFCS)

(/member/dmeister2/)

Hello Amanda!! I've been reading your post! but. What if I what to use pines to get frequencies from A0 to A5 ? I have been reading your guitar tuner post. I would like to to something similar but, reading the 6 strings signals.

What could you recommend to me? :D

greetings and congratulations for your excellent posts

amandaghassaei (/member/amandaghassaei/) (author) dmeister2 9 months ago Reply (CB2SIFPHN81YMVR)

(/member/amandaghassaei/)

it's a little tricky to read from all the analog inputs. you will definitely not be able to get ~38kHz, probably more like 7kHz (or less), is that ok? The arduino is just not fast enough for this type of thing.

huytien (/member/huytien/) 10 months ago Reply (CUN0IT1HLY7I7BQ)

(/member/huytien/)

hi everyone,
I have one problem is that. My goal is sending a PWM 100 kHz to pin 13 of arduino Atmega2560, and print data to serial port ("testing" string, for example as codes below). The problem is that, I can not receive the data (that is "tesing" string). However, when I change to send PWM 1Hz instead of 100 kHz, result is OK. Could any one help me solve this problem? Thank you. Here are codes:

```
boolean toggle1 = 0;
void setup(){
pinMode(13, OUTPUT);
cli();//stop interrupts
//set timer1 interrupt at 100kHz
TCCR1A = 0; // set entire TCCR1A register to 0
TCCR1B = 0; // same for TCCR1B
TCNT1 = 0;//initialize counter value to 0;
// set compare match register to desired timer count:
```

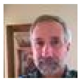
```
OCR1A = 19;
TCCR1B |= (1 << WGM12);
TCCR1B |= (1 << CS10);
//TCCR1B |= (1 << CS11);
// enable timer compare interrupt:
TIMSK1 |= (1 << OCIE1A);

sei();//allow interrupts
Serial.begin(9600);

}//end setup
ISR(TIMER1_COMPA_vect){//timer1 interrupt 10kHz toggles pin 13 (LED)
if (toggle1){
digitalWrite(13,HIGH);
toggle1 = 0;
}
else{
digitalWrite(13,LOW);
toggle1 = 1;
}
}

void loop(){
while (Serial.available() > 0) {
Serial.println("testing");
delay(1000);
}

}
```

(/member/amandaghassaei/)

amandaghassaei (/member/amandaghassaei/) (author) huytien

9 months ago Reply (C84FZVAHN81YMV4)

hmm, the higher frequency interrupt must be disrupting Serial.print. did you find a workaround? you might try using an led indicator for your "testing" message instead.


(/member/bstott/)

bstott (/member/bstott/) 9 months ago Reply (C4EDNEKHMMFB8JJ)

I believe the following description is all wrong ---- > This leads to the confusion of why this topic is hard. I'm trying but can't filter through the errors. Please re-edit your tutorial to be accurate. The TCCR#B is not matched to the CS#. I will not likely be able to understand and catch other errors. And if I do it will be because I'm confused from the error not knowing what is correct. Thanks for trying....

"Finally, notice how the setup for the prescalers follows the tables in the last step (the table for timer 0 is repeated above),
TCCR0B |= (1 << CS22); // Set CS#2 bit for 64 prescaler for timer 2
TCCR2B |= (1 << CS11); // Set CS#1 bit for 8 prescaler for timer 1
TCCR1B |= (1 << CS02) | (1 << CS00); // Set CS#2 and CS#0 bits for 1024 prescaler for timer 0"

(/member/amandaghassaei/)

amandaghassaei (/member/amandaghassaei/) (author) bstott

9 months ago Reply (CVIWBDHHMMFBCLO)

thanks for finding that typo!
it's fixed now

hpan (/member/hpan/) 11 months ago Reply (C3IIKSVHJKC7VNW)

hola, amanda,

(/member/hpan/)

I've been using this tutorial to set up my interrupts for a few times. It's been working great until today. I set timer1 at 2.5khz to perform the interrupt. Mainly use it as a clock and serial reading from the console. The problem is that when I use timer1, PWM on pin 9 and 10 is dead. Pin 9 is the worst one, if I try to use it as analogWrite, arduino will stall. If I use pin 10, arduino will ignore any analogWrite request. It doesn't affect pin 3 and 5. however, if I use timer0, i think it was pin 11 and 3 will be affected in the same way.

Any suggestion to get rid of the problem? or if you know the reason behind it, please do explain here. maybe PWM for PIN 3, 5, 6, 9, 10, 11 use timer0, 1, 2?

Thank you.

amandaghassaei (/member/amandaghassaei/) (author) hpan 11 months ago Reply (C6WOXFTHKCJMJNZ)

(/member/amandaghassaei/)

this is expected. if you look at the atmega 328 pin map: http://cdn3.raywenderlich.com content/uploads/2013/06/atme you will see that timer 1 involves oc1A and oc1B, which are connected to pwm pins 9 and 10. timer0 involves oc0A and oc0B, tied to pins 5 and 6, and timer 2 has the same relationship to pins 3 and 11
the reason those pins are able to do pwm is because they use the timers, so you can't have both. you just have to pick which feature you want to enable.

wlf235 (/member/wlf235/) 1 year ago Reply (CCG8EYXHI3TWJLM)

(/member/wlf235/)

Great article! I have additional question: when we modify timer interruption frequency, changing prescaller and so on, how it will affect analogWrite function call? I mean the frequency of PWM, which is "normally" 490Hz when just using arduino API and analogWrite function? In other words how to use PWM features to control external devices (like LEDs or DC engines) by analogWrite and internal interruptions together?

amandaghassaei (/member/amandaghassaei/) (author) wlf235 1 year ago Reply (CZB6R9DHI3TS4U7)

(/member/amandaghassaei/)

using interrupts, you can essentially create your own, custom analogWrite. You can set this "analogWrite" to any frequency that you want (under 16Mhz).

lnyulak (/member/lnyulak/) 1 year ago Reply (CGPCWORHI3TY6OU)

Awesome article!

(/member/lnyulak/)

Sorry for the total newbie question here, but really struggling to understand the following code:

TCCR0A = 0; ...This zeros out entire register including WGM01
TCCR0B = 0; ..This zeros out entire register including WGM02
.
.
TCCR0A |= (1 < < WGM01);
...since WGM01 was set to zero previously, doesn't this evaluate to TCCR0A = 1?

I thought we needed to set
WGM00 = 0
WGM01 = 1
WGM02 = 0

..to get into CTC mode?

In my thinking, this would set WGM00 = 1, WGM01 = 0, and WGM02, which is in TCCR0B is untouched since being reset in line 2.

So this should put us into "Mode 1 - PWM, Phase Correct Mode"

Cheers

(/member/maxx-on/)

maxx-on (/member/maxx-on/) 1 year ago Reply (CIGRF1KHET0T78Z)

Oh, I see. I can't reply to a message because there is no reCAPTCHA under the reply box, not because the reCAPTCHA isn't working. It works fine if I create a new post. Someone should fix this. I'm using Firefox 19.0.2 if that make a difference.

(/member/maxx-on/)

maxx-on (/member/maxx-on/) 1 year ago Reply (C0CFZCOHET0T777)

Yeah, I know I could do that. I could also put in a function pointer into my interrupt handler, I was just wondering if it were possible to drop that overhead and go directly to the interrupt handler of my choice.

P.S. This reCAPTCHA stuff is garbage.:(I've put in several words that I'm absolutely sure are what they should be and it keeps saying "Please type the two words as seen on image"

(/member/maxx-on/)

maxx-on (/member/maxx-on/) 1 year ago Reply (CJWSWU1HDYZISB2)

Can you only define one interrupt handler per interrupt? Can you not define multiple ones and switch it from one to another depending on circumstance?

(/member/amandaghassaei/)

amandaghassaei (/member/amandaghassaei/) (author) maxx-on 1 year ago Reply (C1RVLHEHET2TJ7I)

put and if then statement inside the interrupt

(/member/mertg/)

mertg (/member/mertg/) 1 year ago Reply (CDUWWBFHDOWANB7)

Awesome. Thanks. This helped me a lot. I was struggling to understand timers in pic. This instructable made my mind clear. Also with arduino.

(/member/amandaghassaei/)

amandaghassaei (/member/amandaghassaei/) (author) mertg 1 year ago Reply (CM7GXPQHET2TJ7G)

thanks!

FieldingBlue (/member/FieldingBlue/)

1 year ago Reply (CO6F5V0HCV99T60)

A nice overview of timer interrupts, thank you for taking the time to write the article. And thank you to those who commented with helpful feedback.

(/member/FieldingBlue/)

rsellens (/member/rsellens/) 1 year ago Reply (CYZZ83WHAQ3B296)

Nice work! One detail:

(/member/rsellens/)

timer2 uses a prescale that runs 1, 8, 32, 64, 128, 256, 1024 for the 001 through 111 bit settings. This is different from the 1, 8, 64, 256, 1024, Ext Falling, Ext Rising prescale values for timer0 and timer1 from the table you showed.

All three work the same for 010 giving a prescale of 8, as used in your examples, but slower timer2 interrupt frequencies will give different results.

amandaghassaei (/member/amandaghassaei/) (author) rsellens

1 year ago Reply (CYIUB1MHAQ2UJRT)

that's interesting, I didn't know that. I'll update that step. thanks for the tip!

(/member/amandaghassaei/)

kbeharee (/member/kbeharee/) 1 year ago Reply (CNH26X8H8FYOLIP)

Awesome instructable! I have one question though.. You use CS10 CS11 CS12 for timer 1 thats okay but for the other timers eg timer 2 should we use CS20 CS21 CS22 for prescalers rather than CS10 11 12 ?

(/member/kbeharee/)

Thanks.

amandaghassaei (/member/amandaghassaei/) (author) kbeharee

1 year ago Reply (CYGWI37H8FYOMSN)

good question! no it's always cs10 cs11 cs12. In the code in step 2 you can find an example of how to set up each timer interrupt, they all use cs10 cs11 and/or cs12. to decide which bits you need to set for a certain presclaer (all of them, one or two of them, etc) you use the table in step 2 (fig 4).

(/member/amandaghassaei/)

kbeharee (/member/kbeharee/) amandaghassaei

1 year ago Reply (C9JZJPLH8FOFUBF)

Thanks alot for your prompt reply!

(/member/kbeharee/)

bilbolodz (/member/bilbolodz/) 1 year ago Reply (CSJUS8EH7430DPU)

Hi,

(/member/bilbolodz/)

My problem is:
External signal (via interrupt) after 6ms delay should trigger some actions (call procedure). Each external signal during "counting delay" should reset counting to start. I've already INT0 procedure but now I've program "delay counter". I thing that best idea is to use timers but how to do it? I've Arduino Mego so I've plenty of "free timers" ;-)

{optr

amandaghassaei (/member/amandaghassaei/) (author) bilbolodz
1 year ago
Reply (CX9U7PAH7996NHY)

(/member/amandaghassaei/)

can you post your code? at least the interrupt routine and the timer setup. Are you using the delay() function? because it might act strangely depending on how you've set up your timers.

---

bilbolodz (/member/bilbolodz/) amandaghassaei
1 year ago
Reply (CSZN11TH7430JQR)

(/member/bilbolodz/)

It's quite big program so it's hard to post all code but "main function" are like this:

```
#define IRQ_1 19
volatile boolean interrupt_1=false;

void irq1_proc(void)
{
interrupt_1=true;
//Serve time critical interrupt stuff
//reset timer (hot do to it?)
}

void timer_proc()
{
//delay passed do something because
there wasn't interrupt longer then 6ms
}

void setup()
{
[..]
pinMode (IRQ_1,INPUT);
digitalWrite (IRQ_1,HIGH);
attachInterrupt(4,irq1_proc,FALLING) ;

//Timer setup (how to do it)
}

void loop(void)
{
//Do some usual stuff
if (interrupt_1 == true) {
interrupt_1 = false;
//do some staff when was interrupt but
not time critical
}
}
```

I'm not using delay() because it's real time system it has external time critical devices.

---

bilbolodz (/member/bilbolodz/) bilbolodz
1 year ago
Reply (CHT4BEMH742SND4)

(/member/bilbolodz/)

OK, If someone interested I've found solution:
delays longer than 9ms between pulses on pin 19 release timer interrupt,
----

```
#define IRQ_1 19
```

```
volatile boolean interrupt_1=false;

void irq1_proc(void)
{
interrupt_1=true;
//Serve time critical interrupt stuff

//reset timer
reset_timer();
}

ISR(TIMER5_COMPA_vect)
{
//delay passed do something because
there wasn't pulse longer then 9ms}
}

void init_timer()
{
  cli();          // disable global interrupts
  TCCR5A = 0;     // set entire
TCCR1A register to 0
  TCCR5B = 0;     // same for TCCR1B

  // set compare match register to
desired timer count:
  OCR5A = 624; //10 ms
  // turn on CTC mode:
  TCCR5B |= (1 << WGM12);
  // Set CS12 bits for 256 prescaler:
  TCCR5B |= (1 << CS12);
  // enable timer compare interrupt:
  TIMSK5 |= (1 << OCIE5A);
  // enable global interrupts:
  sei();
}
void reset_timer()
{
  cli();          // disable global interrupts
- probably not needed because we
are in IRQ1 procedure so interrupts
are already disabled????
  TCNT5=0;
  sei();        // enable global interrupts:
}

void setup() {
[..] pinMode (IRQ_1,INPUT);
digitalWrite (IRQ_1,HIGH);
attachInterrupt(4,irq1_proc,FALLING)
;
//Initialize timer5
init_timer();
}
void loop(void)
{
//Do some usual stuff
if (interrupt_1 == true)
{
interrupt_1 = false;
//do some staff when was interrupt but
not time critical
}
}
```

halamka (/member/halamka/) 2 years ago Reply (CUS8Y7TH4AGLNNQ)

(/member/halamka/)

sounds like another episode of my name is earl. can you help build a commodore computer. wait a second . well get someone who does not complicate things.

Make Comment

About Us

Who We Are (/about/)

Advertise (/advertise/)

Contact (/about/contact.jsp)

Jobs (/community/Positions-available-at-Instructables/)

Help (/community?categoryGroup=Help)

Find Us

Facebook (http://www.facebook.com/instructables)

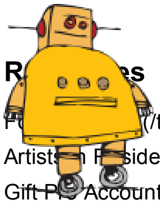
Youtube (http://www.youtube.com/user/instructablestv)

Twitter (http://www.twitter.com/instructables)

Pinterest (http://www.pinterest.com/instructables)

Google+ (https://plus.google.com/+instructables)

Tumblr (http://instructables.tumblr.com)



Resources

For Teachers (/teachers/)

Artist in Residence (/group/air/)

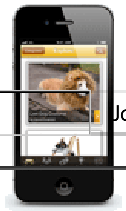
Gift Pro Account (/account/give?sourcea=footer)

Forums (/community/)

Answers (/tag/type-question/?sort=RECENT)

Sitemap (/sitemap/)

Join our newsletter: enter email Join!



Mobile

Download our new apps for iOS, Android and Windows 8!

Android (https://play.google.com/store/apps/details?id=com.adsk.instructables)

iOS (https://itunes.apple.com/app/instructables/id586765571)

Windows (http://apps.microsoft.com/windows/en-us/app/7afc8194-c771-441a-9590-54250d6a8300)

English

Terms of Service (http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=21959721) |

Privacy Statement (http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=21292079) |

Legal Notices & Trademarks (http://usa.autodesk.com/legal-notices-trademarks/) | Mobile Site (http://m.instructables.com)

(http://usa.autodesk.com/adsk/servlet/pc/index?id=20781545&siteID=123112)