Search the Arduino Playground

Manuals and Curriculum
(http://playground.arduino.cc/Main/ManualsAndCurriculum)
Arduino StackExchange
(http://arduino.stackexchange.com)
Board Setup and Configuration
(http://playground.arduino.cc/Main/ArduinoCoreHardware)
Development Tools
(http://playground.arduino.cc/Main/DevelopmentTools)
Arduino on other Atmel Chips
(http://playground.arduino.cc/Main/ArduinoOnOtherAtmelChips)
Interfacing With Hardware
(http://playground.arduino.cc/Main/InterfacingWithHardware)

- Output
  (http://playground.arduino.cc/Main/InterfacingWithHardware#Output)
- Input
  (http://playground.arduino.cc/Main/InterfacingWithHardware#InputTOC)
- User Interface
  (http://playground.arduino.cc/Main/InterfacingWithHardware#ui)
- Storage
  (http://playground.arduino.cc/Main/InterfacingWithHardware#Storage)
- Communication
  (http://playground.arduino.cc/Main/InterfacingWithHardware#Communication)
- Power supplies
  (http://playground.arduino.cc/Main/IntWithHW-
  PwrSup)
- General
  (http://playground.arduino.cc/Main/InterfacingWithHardware#General)

Interfacing with Software
(http://playground.arduino.cc/Main/InterfacingWithSoftware)
User Code Library
(http://playground.arduino.cc/Main/GeneralCodeLibrary)

- Snippets and Sketches
  (http://playground.arduino.cc/Main/SketchList)
- Libraries

(http://playground.arduino.cc/Main/LibraryList)
- Tutorials
(http://playground.arduino.cc/Main/TutorialList)

Suggestions & Bugs
(http://code.google.com/p/arduino/issues/list)
Electronics Technique
(http://playground.arduino.cc/Main/ElectroInfoResources)
Sources for Electronic Parts
(http://playground.arduino.cc/Main/Resources)
Related Hardware and Initiatives
(http://playground.arduino.cc/Main/SimilarBoards)
Arduino People/Groups & Sites
(http://playground.arduino.cc/Main/People)
Exhibition
(http://playground.arduino.cc/Projects/ArduinoUsers)
Project Ideas
(http://playground.arduino.cc/Projects/Ideas)
Languages
(http://playground.arduino.cc/Main/Languages)

PARTICIPATE
(http://playground.arduino.cc/Main/Participate)
- Suggestions
(http://code.google.com/p/arduino/issues/list)
- Formatting guidelines
(http://playground.arduino.cc/Main/Participate#contribrules)
- All recent changes
(http://playground.arduino.cc/Site/AllRecentChanges)
- PmWiki
(http://playground.arduino.cc/PmWiki/PmWiki)
- WikiSandBox training
(http://playground.arduino.cc/Main/WikiSandbox)
- Basic Editing
(http://playground.arduino.cc/PmWiki/BasicEditing)
- Cookbook (addons)
(http://www.pmwiki.org/wiki/Cookbook/CookbookBasics)
- Documentation index
(http://www.pmwiki.org/wiki/PmWiki/DocumentationIndex)
- Drone with Arduino
(http://www.bartoloilliano.com/arduino-tutorial-costruire-un-drone-con-webcam-telecomandato-da-pc-tramite-csharp.html)
- Thermostat with Arduino
(http://arduinothermostat.blogspot.co.uk)

Adjusting PWM Frequencies

For further knowledge on Arduino PWM frequencies refer to the ATMega Complete Datasheet and this Arduino.cc page "Secrets of Arduino PWM" http://arduino.cc/en/Tutorial/SecretsOfArduinoPWM (http://arduino.cc/en/Tutorial/SecretsOfArduinoPWM)

Regarding Arduino Mega

This article is not fully compatible to Arduino Mega (or ATmega2560 microprocessor)

For Arduino Mega: (tested on Arduino Mega 2560)
timer 0 (controls pin 13, 4)
timer 1 (controls pin 12, 11)
timer 2 (controls pin 10, 9)
timer 3 (controls pin 5, 3, 2)
timer 4 (controls pin 8, 7, 6)

TCCRnB, where 'n' is the number for the timer.
TCCR2B for timer 2, TCCR3B for timer 3.
Eg:

```
TCCR2B = TCCR2B & 0b11111000 | 0x01;
//sets Arduino Mega's pin 10 and 9 to frequency 31250.
//code typically inserted in setup()
```

Thanks to valerio_sperati
(http://arduino.cc/forum/index.php/topic,72092.0.html
(http://arduino.cc/forum/index.php/topic,72092.0.html))

How to adjust Arduino PWM frequencies

by macegr in this forum post http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1235060559/12 (http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1235060559/12)

Pins 5 and 6: controlled by Timer 0 in fast PWM mode (cycle length = 256)

```
Setting         Divisor         Frequency
0x01            1               62500
0x02            8               7812.5
0x03            64              976.5625   <--DEFAULT
0x04            256             244.140625
0x05            1024            61.03515625

TCCR0B = TCCR0B & 0b11111000 | <setting>;
```

Pins 9 and 10: controlled by timer 1 in phase-correct PWM mode (cycle length = 510)

```
Setting         Divisor         Frequency
0x01            1               31372.55
0x02            8               3921.16
0x03            64              490.20   <--DEFAULT
0x04            256             122.55
0x05            1024            30.64

TCCR1B = TCCR1B & 0b11111000 | <setting>;
```

Pins 11 and 3: controlled by timer 2 in phase-correct PWM mode (cycle length = 510)

```
Setting         Divisor         Frequency
0x01            1               31372.55
0x02            8               3921.16
0x03            32              980.39
0x04            64              490.20   <--DEFAULT
0x05            128             245.10
0x06            256             122.55
0x07            1024            30.64

TCCR2B = TCCR2B & 0b11111000 | <setting>;

All frequencies are in Hz and assume a 16000000 Hz system clock.
```

Issues from adjusting PWM frequencies and workarounds:

from koyaanisqatsi in this forum post http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1235060559/12 (http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1235060559/12)

If you change TCCR0B, it affects millis() and delay(). They will count time faster or slower than normal if you change the TCCR0B settings. Below is the adjustment factor to maintain consistent behavior of these functions:

Default: delay(1000) or 1000 millis() ~ 1 second

0x01: delay(64000) or 64000 millis() ~ 1 second
0x02: delay(8000) or 8000 millis() ~ 1 second
0x03: is the default

0x04: delay(250) or 250 millis() ~ 1 second
0x05: delay(62) or 62 millis() ~ 1 second
(Or 63 if you need to round up. The number is actually 62.5)

Also, the default settings for the other timers are:
TCCR1B: 0x03
TCCR2B: 0x04

Any thought on the Arduino Mega?

PWM frequencies on Timer 0, pins 5 and 6, Arduino Uno

```
by yanngg, 02-15-2012

Timer 0 uses a prescale factor which is set to 64 by default
To set the prescale factor use this line in the setup function

Setting                          Prescale_factor
TCCR0B = _BV(CS00);              1
TCCR0B = _BV(CS01);              8
TCCR0B = _BV(CS00) | _BV(CS01);  64
TCCR0B = _BV(CS02);              256
TCCR0B = _BV(CS00) | _BV(CS02);  1024

To use Fast PWM on Timer 0
Write this line in the setup function
TCCR0A = _BV(COM0A1) | _BV(COM0B1) | _BV(WGM01) | _BV(WGM00);
And to calculate the PWM frequency, use
Fast_PWM_frequency = (16 000 000)/(Prescale_factor*256);

To use Phase-correct PWM on Timer 0 (half the frequency of Fast PWM)
Write this line in the setup function
TCCR0A = _BV(COM0A1) | _BV(COM0B1) | _BV(WGM00);
And to calculate the PWM frequency, use
Phase_correct_PWM_frequency = (16 000 000)/(Prescale_factor*510);

Changing the prescale factor on Timer0 will affect functions
millis(), micros(), delay(),...

To adjust millis(), micros(), delay(),... accordingly,
You can modify a line in the wiring.c function in the Arduino program files
hardware\arduino\cores\arduino\wiring.c

In the beginning of wiring.c you can find:
// the prescaler is set so that timer0 ticks every 64 clock cycles, and the
// the overflow handler is called every 256 ticks.
#define MICROSECONDS_PER_TIMER0_OVERFLOW (clockCyclesToMicroseconds(64 * 256))

You need to modify the prescale factor in this function to the corresponding line

For fast PWM (default):
#define MICROSECONDS_PER_TIMER0_OVERFLOW (clockCyclesToMicroseconds(PRESCALE_FACTOR* 256))
For phase-correct PWM :
#define MICROSECONDS_PER_TIMER0_OVERFLOW (clockCyclesToMicroseconds(PRESCALE_FACTOR * 510))
```

Example: DC motor drive on the Arduino UNO, pins 5 and 6

```
For Fast PWM of 62.500 kHz (prescale factor of 1)
Use these two lines in the setup function:

TCCR0A = _BV(COM0A1) | _BV(COM0B1) | _BV(WGM01) | _BV(WGM00);
TCCR0B = _BV(CS00);

And modify the line in the wiring.c function in the Arduino program files
hardware\arduino\cores\arduino\wiring.c :
#define MICROSECONDS_PER_TIMER0_OVERFLOW (clockCyclesToMicroseconds(1 * 256))

For Phase-correct PWM of 31.250 kHz (prescale factor of 1)
Use these two lines in the setup function:

TCCR0A = _BV(COM0A1) | _BV(COM0B1) | _BV(WGM00);
TCCR0B = _BV(CS00);

And modify the line in the wiring.c function in the Arduino program files
hardware\arduino\cores\arduino\wiring.c :
#define MICROSECONDS_PER_TIMER0_OVERFLOW (clockCyclesToMicroseconds(1 * 510))
```

Share

(http://youtube.com/arduinoteam)