The playground is a publicly-editable wiki about Arduino.

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

How to adjust Arduino PWM frequencies

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

Pins 5 and 6: controlled by Timer 0

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

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

Pins 9 and 10: controlled by timer 1

Setting	Divisor	Frequency
0x01	1	31250
0x02	8	3906.25
0x03	64	488.28125
0x04	256	122.0703125
0x05	1024	30.517578125

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

Pins 11 and 3: controlled by timer 2

Setting	Divisor	Frequency
0x01	1	31250
0x02	8	3906.25
0x03	32	976.5625
0x04	64	488.28125
0x05	128	244.140625
0x06	256	122.0703125
0x07	1024	30.517578125

- Cookbook (addons)
- Documentation index
- Drone with Arduino
- Thermostat with Arduino

edit SideBar


```
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

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))
```