Brushless setup - driving esc's - firmware settings. Revised Feb, 2018.

This page follows on from the first brushless page with a little more detail about setting up a silverwared FC to control a quad with brushless motors.

It is highly recommended to use BLHeli_S esc's to make things easy and effective. The reason is that they can use Dshot or the older PPM protocols (both protocols give great results), and are easy to set up with BLHelisuite. A BLHeli_S will auto-detect either Dshot or various PPM and oneshot protocols. For Dshot, the esc drive signal must come from the FET gate and you would select dshot_driver_beta in hardware.h. If you do not plan to use the signal from the FET gate (i.e. you will use pull-up resistors after the FET's) then you must use PPM protocol (e.g. oneshot) and select esc_driver in hardware.h.

If you are familiar with Betaflight or Cleanflight, you might wonder what “PPM” is. In silverware, a PPM signal is a conventional signal that drives an esc using (for example) oneshot protocol, and a PWM signal is one that drives an FET for a brushed motor.

If you have a BLHeli flashable esc and want to connect to it to check or change the settings, you will need an interface between BLHeliSuite and the esc. Many people use an Arduino Nano, which is only a few dollars, and the Nano can be flashed by BLHeliSuite to become the interface to an individual esc, or to a 4 in 1 esc (either individually or as a group).

It is **strongly recommended** to connect to your esc's with BLHelisuite to untick the “Programming by TX” check box, and while you are there, check that all four esc's have the same firmware revision and same default settings. The esc's will almost always misbehave if the “Programming by TX” check box is left in the default (ticked) setting. They can enter the programming mode when all you want to do is fly the quad…and if that happens, you will have to connect to BLHelisuite to correct the settings.

How to set up the Dshot esc driver:

The Dshot esc driver is selected in hardware.h.

```
//#define USE_PWM_DRIVER
//#define USE_ESC_DRIVER
#define USE_DSHOT_DRIVER
```

Then, in drv_dshot.c, select either DSHOT600, DSHOT300 or DSHOT150, and adjust the IDLE_OFFSET number if you need to, in order to set the desired “idle” speed. Default is 40 and I often use 32 for high-Kv 100mm and 120mm quads.

```
#define DSHOT600
//#define DSHOT150
//#define DSHOT300

// IDLE_OFFSET is added to the throttle. Adjust its value so that the motors
// still spin at minimum throttle.
#define IDLE_OFFSET 32
```

Note for Dshot: The signal can only be taken from before the FET (FET gate) for it to work.

If you are using Dshot, you do not need to read any further, the following instructions are for PPM.

How to set up the PPM esc driver:

In **hardware.h** - the esc PPM driver is selected instead of the default PWM driver

```
//#define USE_PWM_DRIVER
#define USE_ESC_DRIVER
//#define USE_DSHOT_DRIVER
```

In **drv_esc.c** - the esc min throttle, esc max throttle and throttle_off values are set

The default values for BLHeli_S are Min Throttle 1148 and Max Throttle 1832. The values in drv_esc.c need to be set so that the FC values match what the esc is expecting, so ESC_MAX can be set to 1832 and ESC_MIN is set a bit higher than what the esc is expecting, to raise the idle speed a little, and usually a value of 1152 to 1160 works well. It can be changed/tuned to suit your particular quad and preferences.

```
#define ESC_MIN 1154
#define ESC_MAX 1832
```

ESC_THROTTLEOFF is the value sent to the esc by the FC to to indicate the motors off condition. Anything under 1148 should work but I often use 960 to be sure.

```
#define ESC_THROTTLEOFF 960
```

Finally, the PPM signal polarity needs to be set in drv_esc.c. If you are taking the signal from after the FET with pullup resistors to drive the esc, i.e. from the motor output - (neg) pad, the signal must be inverted in the firmware because the FET also inverts it, (by inverting twice, the signal ends up having the correct polarity).

```
// invert = signal after fets (may need 1k pullup resistor)
// commented = signal straight from CPU pins
#define ESC_INVERT_SIGNAL
```

Or for H101, H8 mini green:

```
// output polarity ( low - motor output with pullup resistor (500 ohms or
near) )
// enable for motor output after fets
#define INVERTED_PWM
```

If you are taking the signal from the gate of the FET (straight from the CPU pins) to drive the esc, the polarity would be set non-inverted, like either of these two boxes:

```
// invert = signal after fets (may need 1k pullup resistor)
// commented = signal straight from CPU pins
//#define ESC_INVERT_SIGNAL
```

Or for H101, H8 mini green:

```
// output polarity ( low - motor output with pullup resistor (500 ohms or
near) )
// enable for motor output after fets
//#define INVERTED_PWM
```