Brushless setup

The h8, h101 and stm32 based boards could be used with brushless escs with some modifications. Depending on the board, inverted flight might also be supported.

H8 boards (green)

The h8 boards need a “esc driver” replacing the normal pwm driver file. From the hardware point, the output could be before or after the FETs. If after, a pullup resistor needs to be installed, of 100 - 1000 ohms, and the capacitor, present across the motors, removed. The h8 board supports non inverted flight only.

H101 boards

The H101 setup is similar to the H8, except it also supports inverted flight, using bidirectional escs. It can also work upright only, using “normal” escs.

STM32 boards

The stm32 port has the esc driver built in, it just needs to be enabled in hardware.h. From the hardware point, the same options apply, before / after fets and pull-ups. This port supports non-inverted flight only. Additional options exist in the “drv_esc.c” file, such as PPM limits.

OneShot 125 note

The blheli escs are not supposed to work with oneshot125 and loop times under 1100. This is the case of this firmwares, in this case, to use oneshot125, the escs have to be flashed with PWM input “OFF”

Using PWM input

The blheli escs can also be flashed to support “pwm” input, and in such case the esc driver is unnecessary. The motor minimum should be changed from zero to a slightly higher value. To enforce the minimum motor limit in config.h uncomment the following:

```
// limit minimum motor output to a value (0.0 - 1.0)
#define MOTOR_MIN_ENABLE
#define MOTOR_MIN_VALUE 0.05
```

The 0.05 value (5%) may need to be adjusted slightly. Use throttle test feature to check for motor start.

The same before/after fets signal options apply. A pwm frequency of 8Khz should be used in the quad code. The pwm input only recognizes certain frequencies.

PPM limits

The PPM limits are usually found in the esc driver file, and should be set correctly. They do not necessarily correspond with the esc set limits, they may be slightly different. Especially important is the “minimum” limit, as the brushless motors should never stop in flight. The minimum limit should be set slightly higher in the code for this reason. A throttle check should be used to make sure the motors never stop above zero throttle.

Bidirectional escs

Only the h101 code currently supports inverted flight and bi-directional escs. Not all escs/motors combinations will be able to reverse direction in the short amount of time needed, I recommend damped light as a esc feature as it will improve changeover time.

BLheli notes

Blheli has option to use “pwm input”, PPM input is always enabled. Oneshot125 is also always enabled, and autodetected, but may interfere with the pwm input. Blheli will autodetect the input type after powerup. “Enable pwm input” should be set to off if not used, and “programming by tx” unchecked as well.

Always consider that the quad may malfunction, and may go full power suddenly. Do not flash the quad with the escs connected/powered. Always remove props when testing. Do not fly quads that have severe oscillation/wobbles.

Remember, this is **experimental**.

Links

H101 esc driver : (rcgroups)

H8 mini green board esc driver: (rcgroups)

BASIC BRUSHLESS ESC SETUP (NON-3D)

See in the screenshot of these esc's, all the settings are default except Startup Power. If a setting is not default, you get the little house symbol beside the slider. I would leave the Startup Power at 0.50 and all else at default and see how it goes. Programming by TX needs to be unticked also, otherwise sometimes the esc's can enter programming mode when you don't want them to.

Its a good idea to check each esc by connecting to BLHeli, hit Connect and then Read Setup, and make sure each esc has the exact same settings, and also the exact same firmware, and exact same firmware revision number. You can do it manually, or you can, if you want, do it this way: Read the first esc. If you are happy with the settings, (don't forget to untick the tx programming box), you can then click on ESC Setup/Save Setup to Ini File and save those settings in a file, give the file a name like XM20 1a or whatever so you can find it in the next step. Then Disconnect from that esc, and Connect to the next esc, go to ESC Setup/Read Setup From Ini File, find the file you just saved, and write those settings to the esc. After writing a setup, click on Read Setup to confirm the settings actually made it into the esc. Then click on Disconnect. Do that for the remaining 2 esc's too. That way, all 4 esc's will have identical settings.

Now, the following text assumes you are setting this quad up for normal flight, as opposed to 3D inverted flight. For 3D, a completely different setup is used.

Notice the values in the bottom right corner in BLHeli, PPM Min Throttle 1148 and PPM Max Throttle 1832. These need to be entered into the drv_pwm.c file like this:

```
#define ESC_MIN 1148
#define ESC_MAX 1832
```

This tells the FC processor what signals the esc is expecting for min and max throttle (so the esc and the FC are synchronised).

NOTE: For H101 board using the esc driver file, you may need to set ESC_MIN approximately 50 higher to get a reliable motor start.

#define ESC_THROTTLEOFF 960

This tells the FC processor what signal to send to the esc to indicate throttle off condition. (It has nothing to do with BLHeli). 960 works for me, possibly any value from 900 to 1000 will work. And of course, for normal flight as opposed to 3D you would set bi-directional control off like this:

```
// enable for bi-directional control
//#define ENABLE_REVERSE
```

PID's - I would start with something like this for a 150mm quad. If its too doughy/very slow response, try the 2nd set below and go from there. (These are some approximate values from similar size quads I have.) Note there is no way your quad will fly properly on these exact settings, the chances of that are incredibly small. It's just a suggested starting point.

float pidkp[PIDNUMBER] = { 4.0e-2 , 4.0e-2 , 2.0e-1 };
float pidki[PIDNUMBER] = { 2.0e-1 , 2.0e-1 , 1.0e-1 };
float pidkd[PIDNUMBER] = { 4.0e-1 , 4.0e-1 , 2.0e-1 };

float pidkp[PIDNUMBER] = { 6.0e-2 , 6.0e-2 , 3.0e-1 };
float pidki[PIDNUMBER] = { 3.0e-1 , 3.0e-1 , 2.0e-1 };
float pidkd[PIDNUMBER] = { 6.0e-1 , 6.0e-1 , 3.0e-1 };

If you end up going round in circles trying to tune it, and can't seem to find some good settings, try some different props. Props that are a low load for the motors (low pitch or small diameter for the motor) are often difficult to tune right IME.