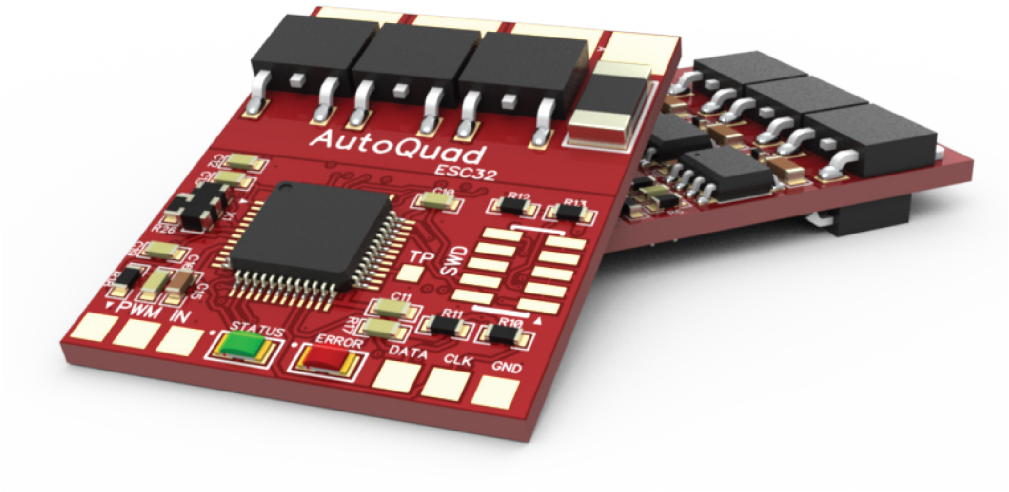
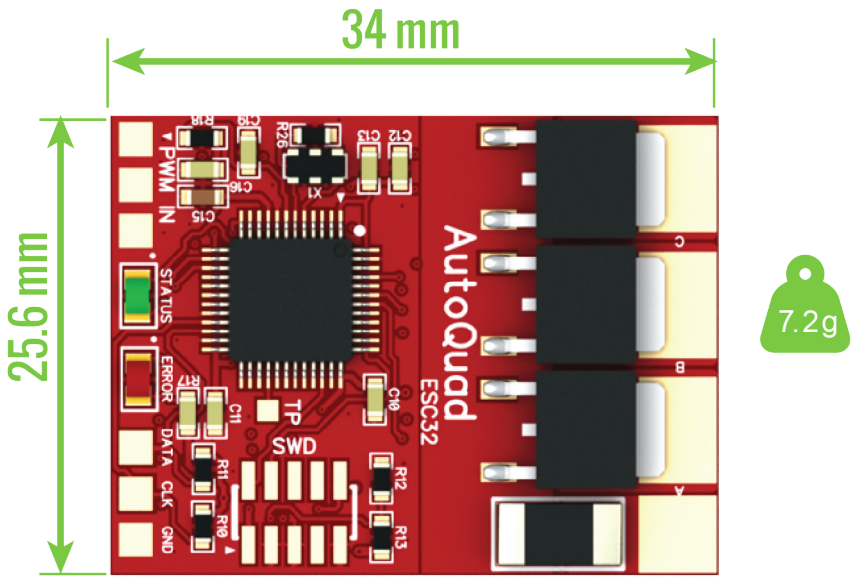




Autoquad ESC32 Features:

- STM32F103 72MHz ARM 32-bit Cortex(TM) M3 MCU
- All N-FET design with gate drivers
- 2S through 5S battery voltage
- Option to power logic side via UART or PWM IN at +5V
- CAN transceiver hardware support onboard
- Firmware written completely in C
- Cortex SWD connector pads for real-time debugging
- Communications ports: PWM IN / UART / I2C / CAN Bus
- Communications protocols: PWM IN / CLI / binary / 1-wire / I2C** / CAN**
- 4KHz to 64KHz PWM out
- Current sensing / limiting with real shunt resistor
- Virtual current limiter
- Regenerative braking (experimental)
- Closed loop control modes
- Lot of available RAM / FLASH for experimentation and development

** I2C and CAN drivers not yet released!

Important note from ESC32 creator Bill Nesbitt about current levels attainable:

"I am sure the first question will be how much current can they handle. The truth is that I don't really know. According to their data sheet, the FETs are capable of 160A continuous. As a practical matter, there is no way you could come close to dissipating the amount of heat that would be generated at those levels. With large heat sinks you could probably run them over 50A continuous, but without any heat sinks the limit will be significantly less. In the end, the limiting factor is simply cooling. If you can get them into some prop wash or rig some sort of small heat sink your results will get better. I've been flying them for for more than 6 months with no heat sink at all, but my machines typically weigh less than 3Kg."

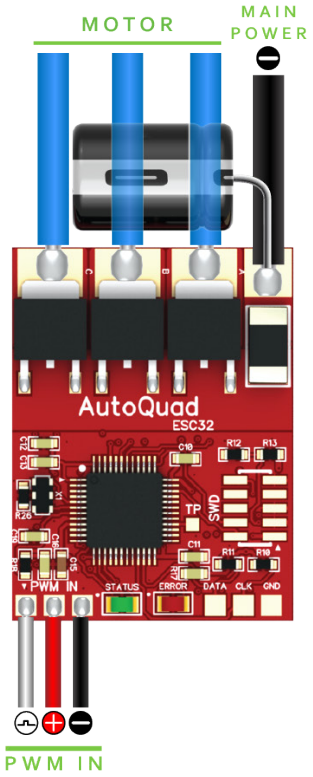
1. Assembly of ESC32:

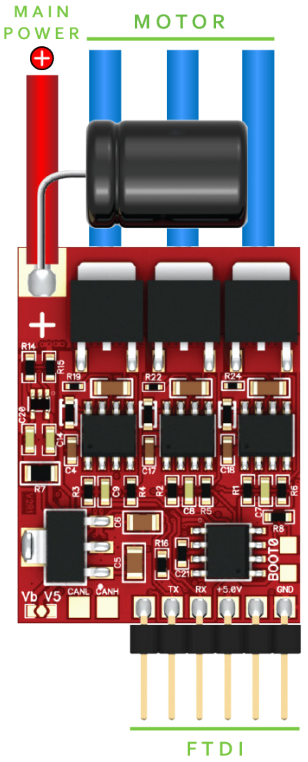
Autoquad ESC32 comes as a "SMT-preassembled" kit. This means the endusers need to add all connections by themselves. This will give you the highest flexibility in configuring the ESC32 to your exact needs.

Kit contents:

- ESC32, SMT-preassembled and factory tested board.
- 25 cm servo wire with open end.
- Input power wires
- 330uF/35V lowESR capacitor
- 1*6 pin header for UART

It´s up to you to configure the wiring so it fits your needs and frame.
A default configuration ins shown in the illustrations:





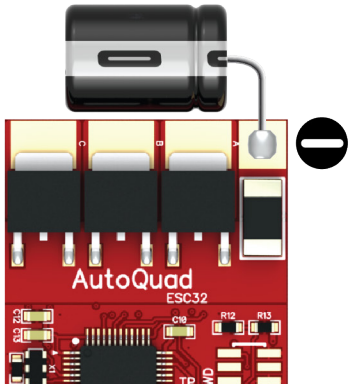
1.1 Input capacitor:

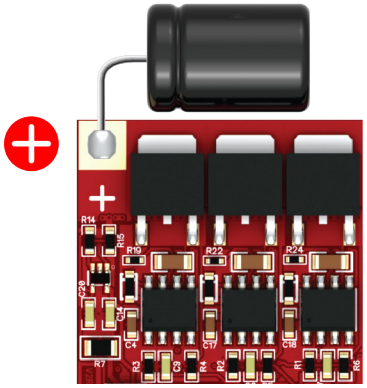
The input capactitor is a critical component for ESC32 operation. The capacitor (330uF/35V) supplied with the kit is safe for “normal” use on up to 4S with continuous currents below 10A. Increase capacitance (add another in parallel or increase the capacitor size) if:

- Running long wires from battery to ESC (more than 25cm)
- Running high continuous currents (Cooling needed. Refer to chapter 3)
- Using 5S (use 50V capacitor for 5S)

Input capacitor should be Low ESR type always!

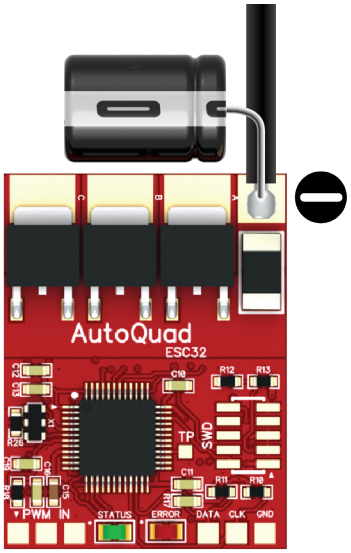
The Input capacitor(s) is soldered directly to the + and – pads on the ESC. Observe the input capacitor polarity! The MINUS (-) mark on the capacitor must be connected to the '-' terminal of the ESC32. **Failing this can blow up the capacitor and damage the ESC32!**
Make sure the capacitor legs are kept as short as possible and that they cannot be short-circuited to each other - consider using a small piece of shrink-wrap on each leg. Check for solder bridges and shorts across the capacitor legs.

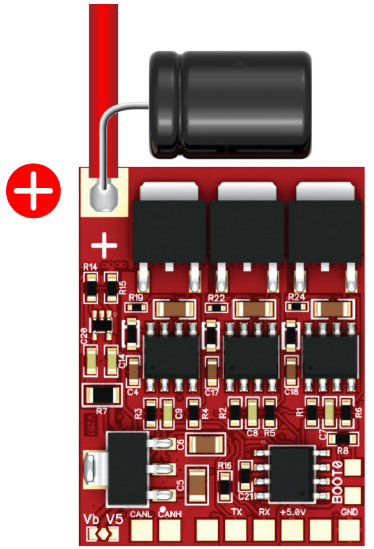




1.2 Power connection:

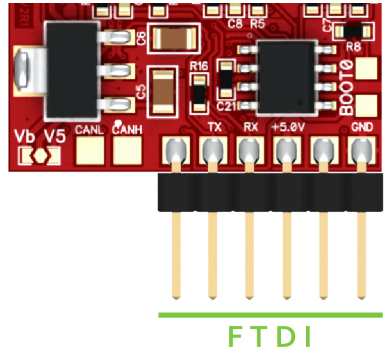
There are multiple ways the main input wires can be connected. We leave it up to you to determine what works best for you. Make sure the there is no shorts across the main power input or the motor phases!





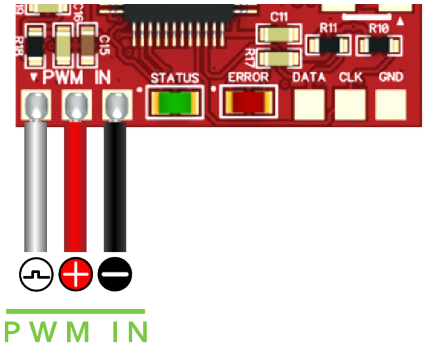
1.3 UART connection

The UART connector matches a standard FDTI style 6-pin connector. It is used for flashing firmware and connecting to GUI/CLI or running the ESC over the UART interface.



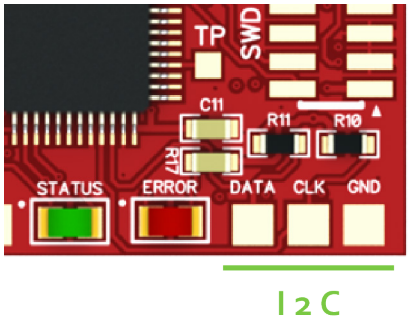
1.4 PWM connection

Solder the PWM servo wire to the PWM pads or add header pins.



1.5 I2C connection

The ESC32 offers a I2C bus connection for future use. Support for MK I2C control will be released as soon as we had a chance to test it properly - talented people are working on this - a beta release is expected soon!

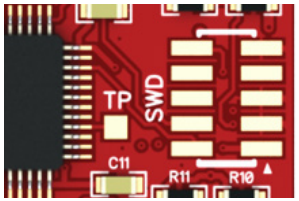


1.6 CAN-connection (Optional)

ESC32 can be fitted with an optional CAN transceiver (Texas Instruments SNHVD232) to provide CAN-bus control and bidirectional communication at up to 1Mbit/s. Currently CAN is not supported in the Firmware. It is also a work in progress, and we will release a driver for it when we had more time to implement and test CAN control.

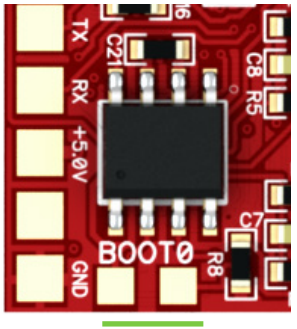
1.7 Cortex SWD connection. (J4)

A standard 10-pin SMT Cortex (TM) SWD connector (Samtec FTSH-105-01-F-DV) can be soldered to the board for realtime debugging of ESC32.



1.8 Boot jumper

The boot jumper is used to place the STM32 bootloader in flash mode for uploading new firmware. Short these two pads during power-up to activate the Bootloader and place ESC32 in flash mode (Hint: you can use a pair of tweezers, if you are bit handy - you don´t need to keep the pads shortened after the ESC has been powered up)



2 - Select power for logic side

Autoquad ESC32 is an “Opto” style ESC. It means it has no built-in BEC – so It cannot powersupply anything but the motors and itself.

The ESC32 has two options for powering the logic side:

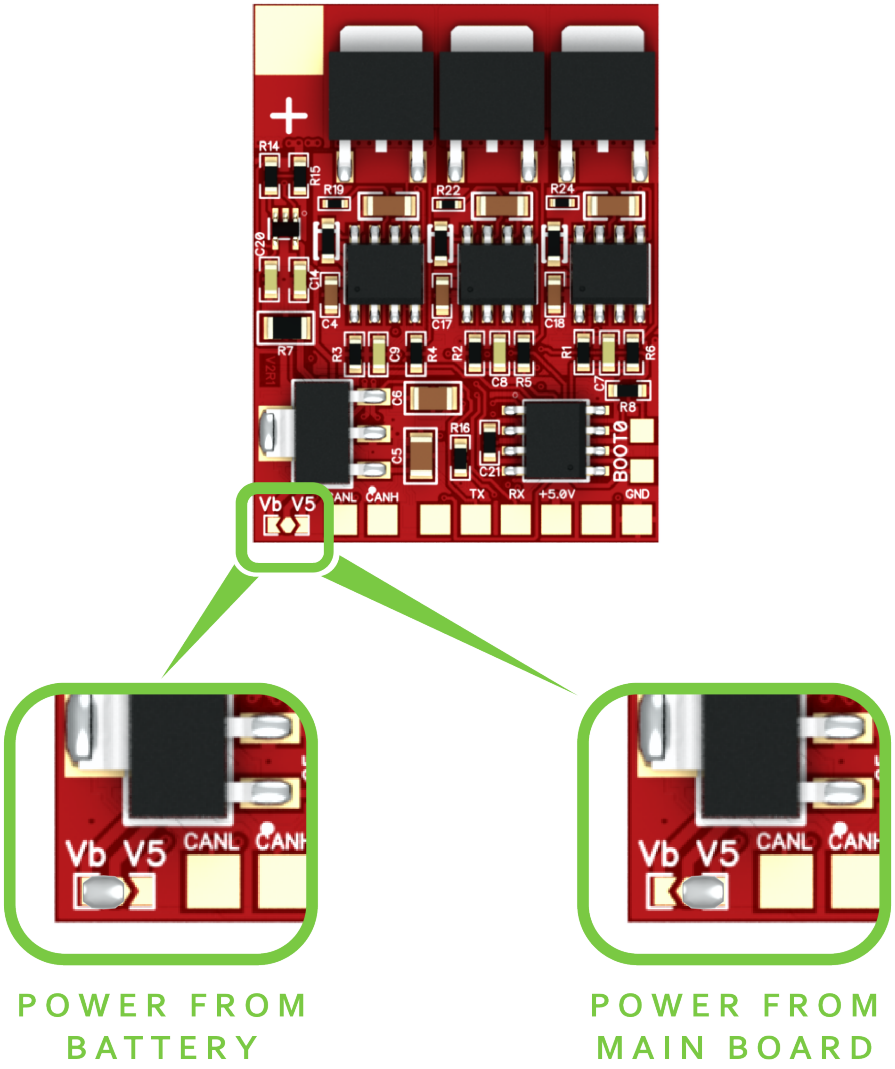
- On-board 3.3V LDO powered by main battery (Vb)
- 5V coming from UART or PWM interface (V5)

In most cases (especially when using LiPo batteries larger than 3S) it is recommended to power the ESC MCU with 5V from the PWM or UART interface – either from an external regulator or a UBEC hooked up to the servo rail on your FC or receiver.

Selecting 5V, also means you can power up the logic side separately when flashing firmware or altering settings over 1-wire or UART interfaces, without the need to power up the motors.

When using the 5V option to you must always DISCONNECT MAIN INPUT FIRST before disconnecting the 5V power to PWM or UART interface - disconnecting 5V while the main is still powered can damage the ESC.

To select the power source for MCU, locate the small solder jumper in the corner on the power side. It’s a simple 3-way solder jumper and works by connecting the middle pad to either the “Vb” or “V5” pads.



3 - Final checks before powering up

All ESC32 are tested in factory, before beeing released. Since it´s a delicate piece of electronics running high currents, you should Inspect your soldering carefully and make sure there are no shorts or solder splashes before powering up the first time. Measure with multimeter (if you have one) for shorts across the main power, PWM and the 3 motor phases.
First time you power up the ESC32, we recommend to have the 5V option selected and hook it up to the AutoQuad widget in Qgroundcontrol or CLI via the UART interface without main power connected.
Alternatively you can hook ESC32 to a simple servotester or receiver with 5V on the PWM interface. Don´t connect main power yet!
If the ESC32 gets 5V and a PWM pulse width of at least 750 microseconds, the green LED will light up showing you that the ESC32 is receiving a valid signal from your FC, RX or servo tester.
You can now connect a power source to main power and check if the motor runs when you increase throttle. A current limited power source is highly recommended when doing the first tests! A 9V/500mA DC adapter can be used for this purpose.This will secure nothing gets burned in case you have a short in your soldering somewhere. If you connect directly to a highcurrent source (like a LiPo battery) and there is a short, you can fry both ESC, battery and motor very quickly.
Summary: Inspect your soldering carefully and test the ESC32 on a limited supply before connecting it to a LiPo battery or other highcurrent source) ESC32 is now ready to run as a standard PWM ESC with a very high motor refresh rate and a 1000-1950us throttle range with the motor start at 1100us. As such it can now be mounted to your favorite multi and flown right away.

4 - Current and cooling considerations

How much current you can actually draw from ESC32, depends ultimately on how good they are cooled.
Our general recommendation is, that with "normal" continuous current at under 10A per motor, special cooling considerations are not needed. The gate-driven design means that the ESC´s wont warm up considerably when run at low currents.
The upper limit is probably upwards of 50A, but that will require extremely good cooling. The factory default for the current limiter in the firmware is 20A continuously. Going beyond that limit will require attention to proper cooling and temperature of the ESC.
Bear in mind that most "normal" setups under 3kg will stay well under 20A continously per motor in normal flight conditions and hover (or something is very wrong with your choice of motors and props).
Always test your individual setup and if you see signs of the ESC heating up it means you should add cooling. The better you cool your ESC´s, the more efficient they will also be so! In theory, they should continue to operate up to around 70-80 celsius, but they will age faster when run at high temps!
The best strategy keep ESC´s cool in a multirotor, is to place them in the wash from the rotors (Ideally right under the motors). In frames where that is not possible because of wiring considerations or if running high currents, heatsinks should be applied to the FETS on both sides. In general its not good practice to place ESC´s in tight places without any airflow, especially if running high currents!

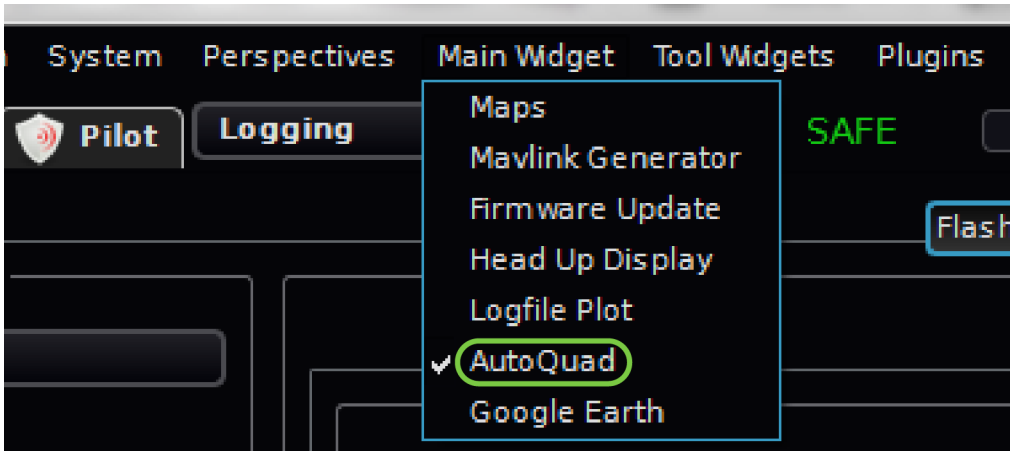
Connect ESC32 to autoquad widget in QGroundControl

The Autoquad version of QGroundControl (QGC) has a widget included to alter settings of ESC32

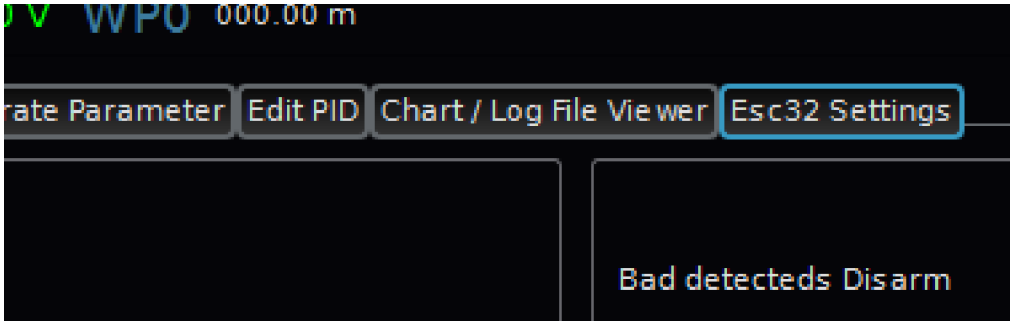
Download Qgroundcontrol for autoquad from:

http://code.google.com/p/autoquad/downloads/list

Open QGC, and choose “Autoquad” from the main widget menu:



Then go to the tab "ESC32 settings"

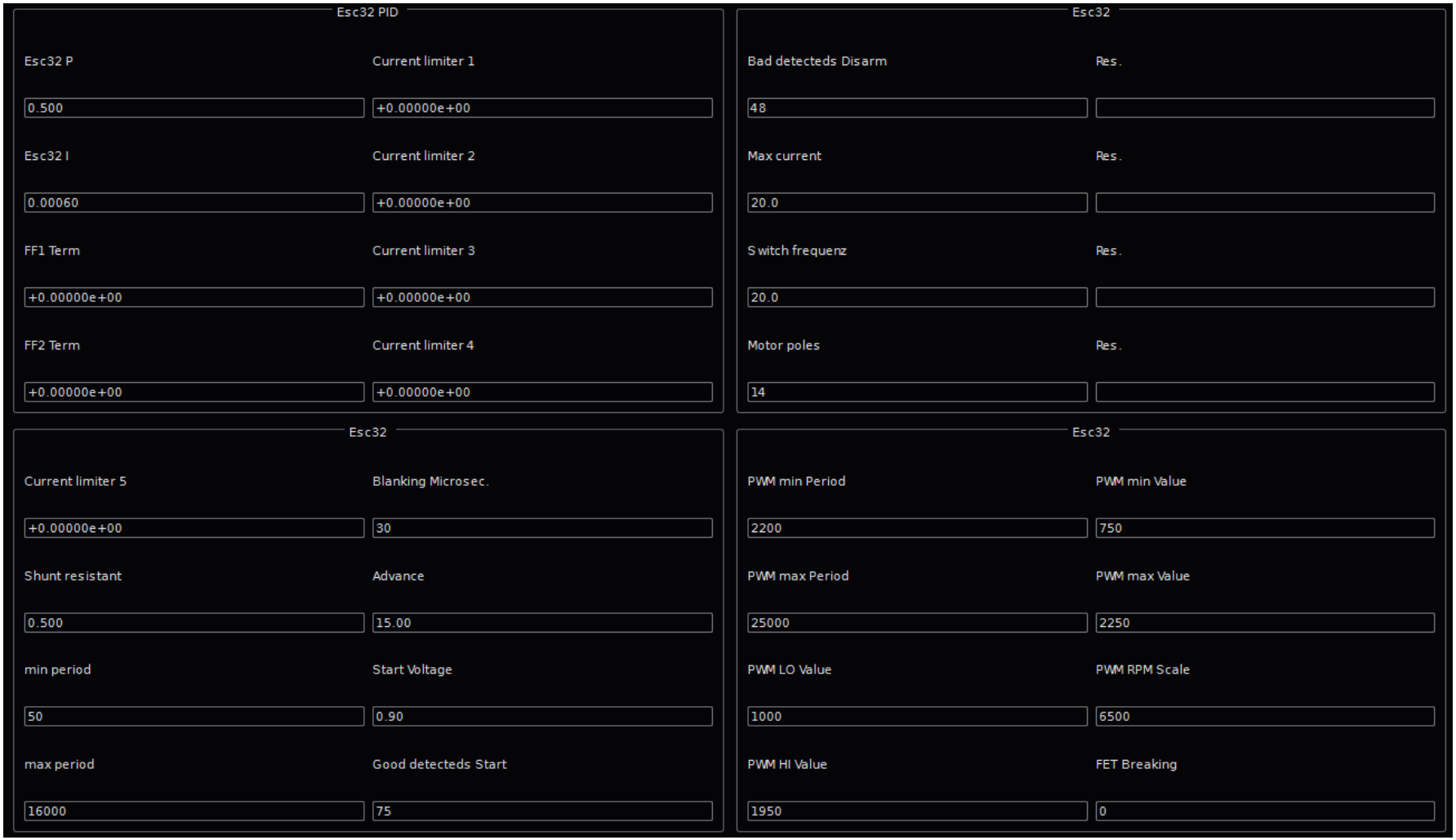


On the left side of the screen you can see the "Link ESC32" Tab:

Connect UART adapter to ESC32 and computer and choose the UART adapter you are using and click connect.
When you connect the ESC to UART, it will go into disarmed state (Red LED blinking). Try and click “arm” - the ESC green LED should light up. If there is a main power source and motor hooked to the ESC you can run the motor by clicking “Start” and the motor will run at low duty (Remove props always when working with your multirotor)

Now you can alter the parameters in ESC32, but make sure you dont change parameters unless you are know how the different parameters affects operation. Certain parameter combinations can be fatal for the ESC and motor!

Refer to the User configurable parameters section for more information on what each parameter controls.



When parameters has been altered, you can upload them to ESC32 by clicking "write config"

User configurable parameters:
Generally, no parameters will need to be changed for normal out of the box PWM operation. However, there are various parameters that can be set . Each parameter is a single numerical value. Note that some combinations of parameters can cause damage to your motor or ESC. It is recommended to leave most values at their default values unless you are sure of how they impact function and operation.

Parameter	Factory Default	Description
STATUP_MODE	0	This is the run mode that the ESC defaults to when first powered on. 0 == normal open loop operation, 1 == closed loop RPM mode, 2 == closed loop thrust mode (not yet implemented)
BAUD_RATE	230400	The UART baud rate. Allowable range 9600 to 921600
PTERM	0.5	The P term for the RPM PI controller
ITERM	0.0006	The I term for the RPM PI controller
FF1TERM	0.0	Feed forward terms used for the RPM controller. Closed loop RPM mode will not function until FF1 & FF2 terms have been set.
FF2TERM	0.0	These values should be calculated using the esc32Cal program with the --r2v option
CL1TERM	0.0	CL1TERM through CL5TERM are used by the virtual current limiter which will not function until they are set.
CL2TERM	0.0	If CL1TERM through CL5TERM is not set, a PI controller will be used for current limiting instead.
CL3TERM	0.0	CL1TERM through CL5TERM can be calculated using the esc32Cal program with the --cl option.
CL4TERM	0.0	
CL5TERM	0.0	
SHUNT_RESISTANCE	0.5	This parameter should **NOT be changed** from the factory default!!! (Value in milliohms)

MIN_PERIOD	50	The minimum commutation period allowed in microseconds
MAX_PERIOD	12000	The maximum commutation period allowed in microseconds
BLANKING_MICROS	30	The number of microseconds to ignore back EMF after a commutation.
ADVANCE	15	The amount of timing advance in electrical degrees. There are 60 electrical degrees in a commutation cycle. This value can be set from 0 to 30 degrees.
START_VOLTAGE	1.1	The amount of voltage presented to the motor during startup. Allowable range is 0.1v to 3.0v
GOOD_DETECTS_START	75	Once started, the number of good, in order zero crossings needed to be detected before the motor is considered to be in the running state.
BAD_DETECTS_DISARM	48	The number of missed zero crossing detects allowed before the ESC considers the motor not to be running at which point will go into the disarmed state.
MAX_CURRENT	20	The maximum amount of current in amps that the ESC will allow. Current is dynamically regulated. Always set this value low and only increase it if you know what you are doing.
SWITCH_FREQ	20	The output PWM pulse frequency used to power the motor windings in KHz. Valid range is from 4KHz to 64KHz.
MOTOR_POLS	14	The number of magnetic poles used in the motor's construction. This value only needs to be set correctly if you want to use the RPM closed loop mode.
PWM_MIN_PERIOD	2200	The minimum period in microseconds that the ESC will consider an input PWM waveform to be valid. Default value represents approx 450Hz
PWM_MAX_PERIOD	25000	The maximum period in microseconds that the ESC will consider an input PWM waveform to be valid. Default value represents approx 40Hz
PWM_MIN_VALUE	750	The minimum input PWM pulse length in microseconds which the ESC will consider to be valid.
PWM_LO_VALUE	1000	The input PWM pulse length in microseconds for the lowest throttle setting.
PWM_HI_VALUE	1950	The input PWM pulse length in microseconds for the highest throttle setting.

PWM_MAX_VALUE	2250	The maximum input PWM pulse length in microseconds which the ESC will consider to be valid.
PWM_MIN_START	1100	The input PWM pulse length in microseconds at which the motor will be started. Once running, the throttle can be brought as low as PWM_MIN_VALUE as long as the motor does not stall.
PWM_RPM_SCALE	6500	The scale of the input PWM pulse length. In closed loop RPM run mode, PWM_LO_VALUE will indicate 0 RPM and PWM_HI_VALUE will indicate this RPM. Closed loop modes only!
FET_BRAKING	0	Turns on regenerative braking in closed loop modes (experimental), 0 == off, 1 == on. Closed Loop modes only!!

Closed loop calibration

The ESC32 offers a closed loop RPM mode and a Closed loop thrust mode (Experimental). It requires calibration of the specific motor and prop you will be using. The calibration sequence is not yet supported in QGC, but can be run under Linux and OSX from a terminal.
Closed loop calibration is not covered by this manual yet, because it can be a bit hairy to run under Linux - we are working hard to get it included into QGC, so it can be done with a single click.
Refer to forum.autoquad.org for more details on how to perform closed loop calibration of ESC32 under linux or OSX.

Environmental
ESC32 is produced in Germany and is fully ROHS-compliant, meaning that lead and other poisonous substances are minimized as much as possible. Our manufactorer lives up to German and EU standards for environmental protection and worker safety!
Electronic products should be recycled properly when disposed of!

Autoquad ESC32 manual. Rev 0.0 preliminary. 25/6-2012, JH
Illustrations by Max Levine