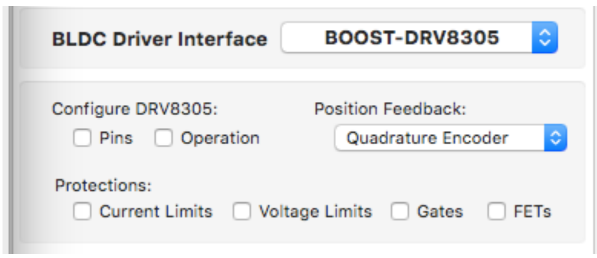
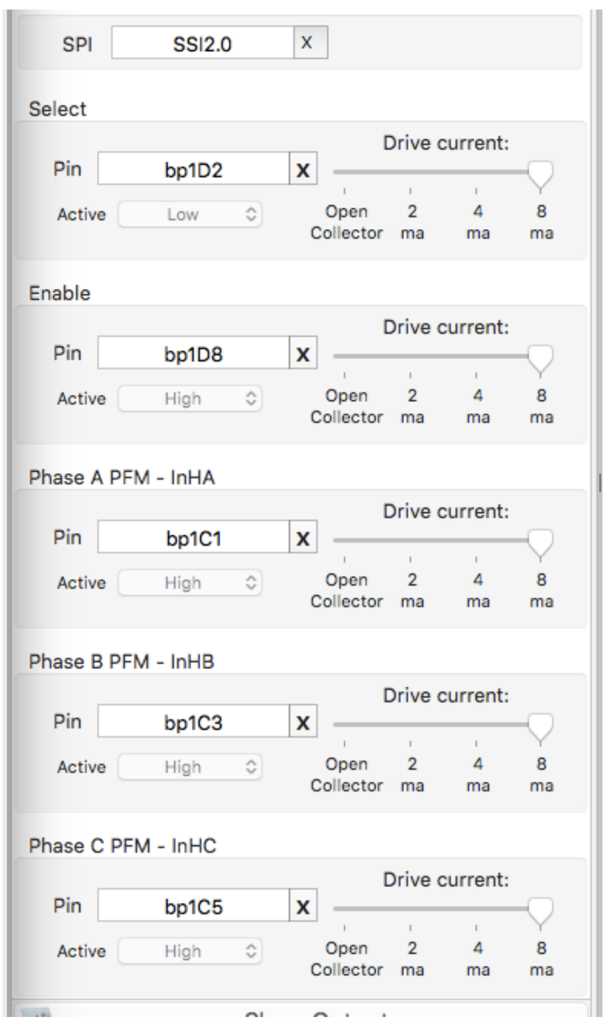
BOOST-DRV8305EVM Setup

If the selected interface type is BOOST-DRV8305EVM then a configuration menu for the high-power three-phase motor driver board will appear having checkboxes to expand in further detail.

Pins



Required pins for the BOOST-DRV8305EVM can be dragged from the pins window outline. The SPI pin selects the SPI to use, normally either 2 for BoosterPack 1 or 3 for BoosterPack 2. The daisy-chain number is not used, rather a discrete select pin is assigned for each board.

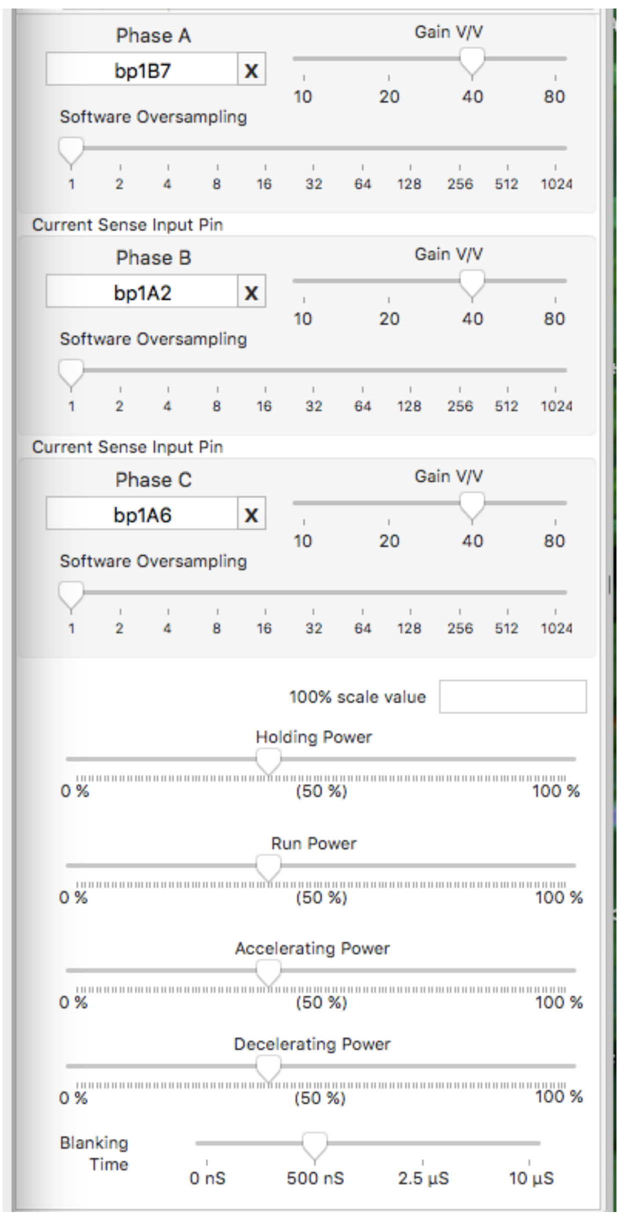


The driver is configured in the 3-PWM mode which requires a switch analog output for each motor phase connection. This mode support micro-stepping using three pins. Because Soft PFM is used as a substitute for PWM, any GPIO pins can be used.

Operation

The default operation has protections off for easier setup, but you should experiment with these settings. Turn on any protection that will still perform satisfactorily.

Current Limits

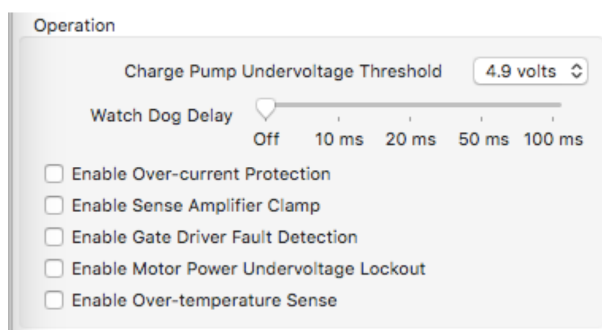


Note: Current sense input for the 8305 is still in development.

The board supports independent current sense feedback for each motor phase although they would normally be balanced with the same amplifier gain and oversampling.

A textfield enables calibration of the 100% scale position of the sliders. Maximum current can be selected independently for each of the four motor states.

Blanking time is the time that the current amplifiers are ignored following PWM output switching. It is a method the IC uses to filter out some of the switching noise generated.

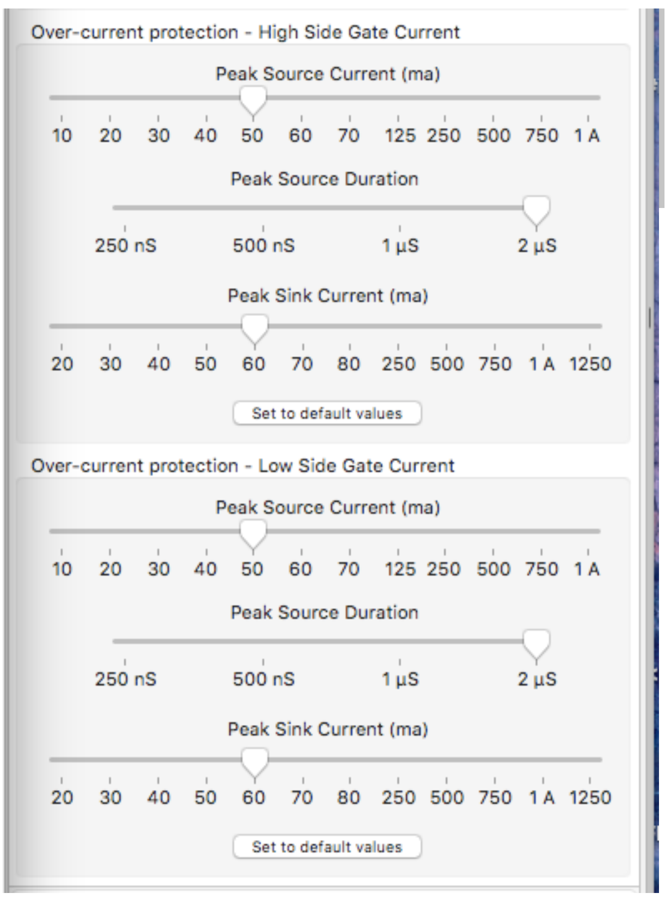


Motor Power Supply

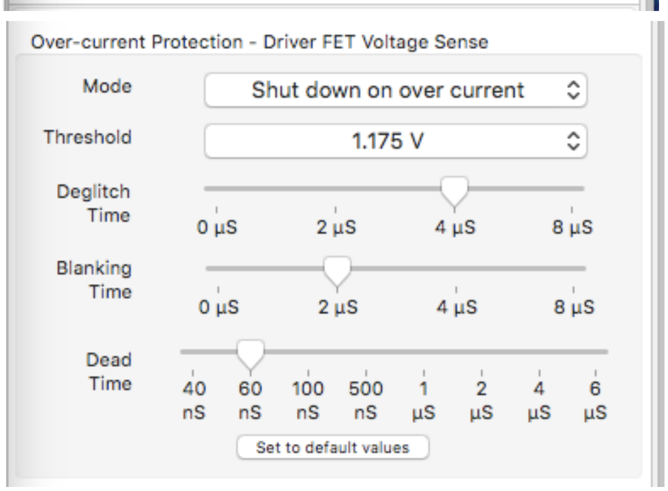
The BOOST-DRV8305EVM board has an analog output for the motor power supply voltage. RiceCNC will assign the pin only for the first driver board and use it as a common input.

If configured, the logic will sample the motor power supply to determine the maximum, i.e., no load, voltage. If the voltage falls below the threshold percentage, then the logic will current limit the motor current attempting to maintain power and prevent the power supply from shutting down. Of course, it is preferable to use a power supply with adequate reserve capacity.

Gate Protection



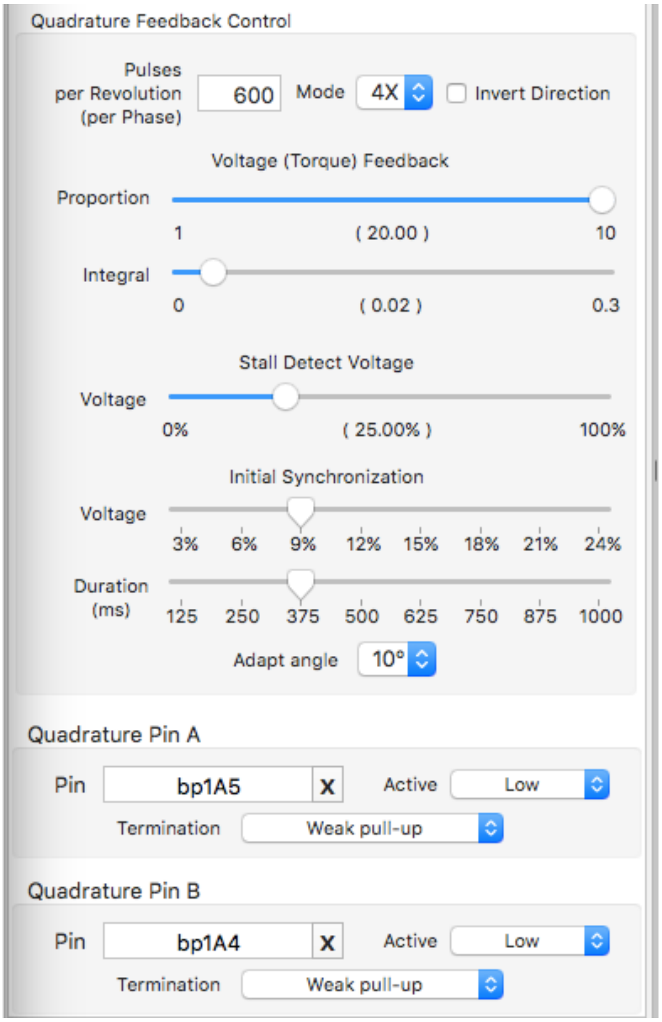
The TI 8305 gate driver IC has robust means to protect itself from overload. Consult the TI documentation before deviating from the default gate protection settings.



MOSFET Driver Protection

The TI 8305 gate driver IC monitors voltage drops across the driver FETs to provide output over-current protection. Consult the TI documentation before deviating from the default FET protection settings.

Quadrature Encoder Feedback



Quadrature encoder feedback is recommended for BLDC motors. Using an optical quadrature encoder both efficiency and position accuracy are maximized. The motor will use power only as required to maintain the desired position or speed.

Voltage is applied leading 90° plus an adjustment for motor speed. This adjustment is significant and may be up to additional 60°. Efficiency is optimal when the stator current vector leads the armature pole vector by 90°, however due to coil inductance, the current will significantly lag the applied voltage at high motor speed.

The quadrature counter can run in either 1X or 4X mode. 4X mode is more accurate but will generate more interrupts.

An error is periodically calculated between the target motor position and the actual position determined from the encoder feedback. Proportion/Integral feedback control will adjust the applied voltage and correspondingly the actual motor speed. However, if the stall detect voltage is reached then a stall condition is signaled and the motor is shut down.

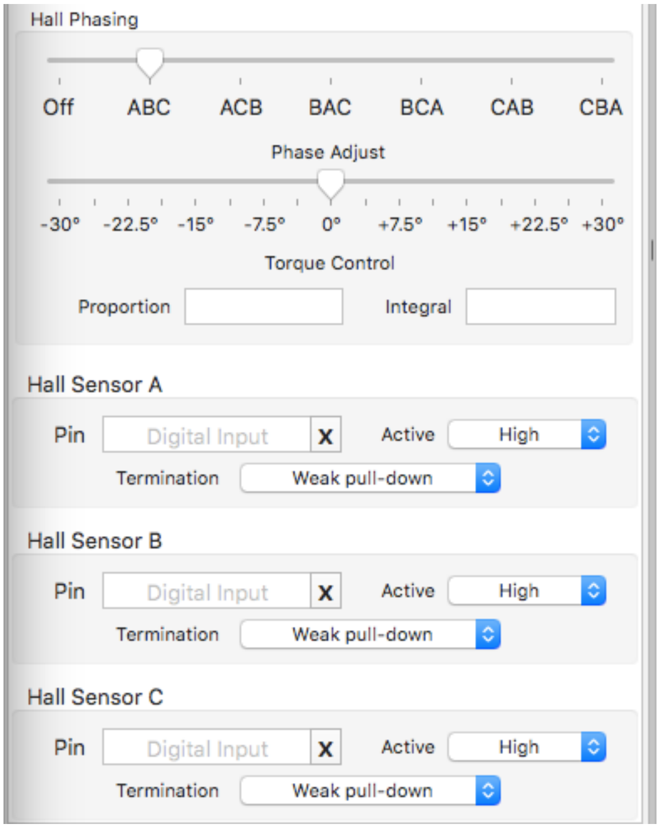
Upon startup, the motor is driven to particular stator phase and then the quadrature count is zeroed. The initialization voltage and duration must not overload the power supply or overheat the motor.

Adaptive logic can adjust for an occasional count error but cannot compensate for a noisy or incorrect counter interface. If the counter is working reliably, then disable this option.

Hall Sensor Feedback

Note: Hall sensor input for the 8305 is still in development.

Configure the hall sensors if using a hall sensored motor.



Many BLDC motors have hall sensors to use for trapezoidal commutation and position feedback. Some motors can directly replace DC motors using hall sensors for self-commutation.

RiceCNC uses micro-stepped Space-Vector Sinusoidal Commutation (SVSC). Hall sensor inputs are used to drive a Phase-Locked-Loop (PLL) from which optimal torque and stall can be determined.

Experiment to determine the hall phasing that will run the motor most efficiently. Only one of the six permutations will work well. The slider is more convenient than swapping pin assignments.

Many motors will have a mechanical adjustment for the phasing - usually factory adjusted. The phase adjust slider will have the same effect to optimize the torque vector.

A Standard Proportion-Integral-Derivative (PID) algorithm (with the derivative component always zero) is used to adjust the motor torque to correct position error between the calculated magnetic electrical angle and the actual magnetic angle reported by the hall sensors. The proportion component can be thought of as the amount of fast response and the integral component specifies the slow response. Too little response and the PLL logic will not be able to maintain synchronization. Too much response and the motor speed will resonate.

When the PLL is in synchronization and the motor is above a minimum speed, torque is applied either leading 90° or trailing 90° from the armature magnetic angle for maximum efficiency. As the motor comes to a slow speed or stop, feedback for the low resolution hall sensors becomes ineffective. Logic then switches to the open-loop mode of operation with torque applied at 0°. This provides holding torque with micro-step precision.

Note: In this release phasing and feedback control is input manually by experimentation, however, algorithms exist to self test the motor and automatically determine the optimal settings. These algorithms may be implemented in a future release.