Husky Robotics - Electronics - Firmware

# Motor (BLDC & GPM) Boards: Firmware Specifications

by Michael Tsien, 2024

Boards: STM32 & PSoC
Firmware Engineers:

- BLDC - PSoC: Max Lan, Aadithya Manoj
- BLDC - STM32: Eun Be Cha, <NAME>
- GPM - STM32: Ayush

## Firmware Project Overview

The Brushless DC Motor (BLDC) and General Purpose Motor board are used to control parts of the rover like the base, shoulder, elbow, etc., with one board driving one motor each (for example, one board for the base, one board for the shoulder, and one board for the elbow). Functionally, the BLDC board is essentially identical to the general-purpose motor boards, which control motors through one of two modes: PWM or PID.

- In PWM mode, the motor will run for 1 second at a speed given by a CAN packet, after which, the motor will stop if no new CAN packet has sent a request to run the motor.
- In PID mode, the motor will rotate to a target position given by a CAN packet using the proportional, integral, and derivative constants also set with received CAN packets.

To use the PID mode, the boards must also be able to receive feedback to determine positional error. With BLDC motors, this feedback will be provided by Hall Effect encoders. Additionally, the BLDC motors will be driven by an ODrive driver to bypass the issue of internal BLDC motor mechanics and timing.

## Background Information

- General Purpose Motor Board (GPMB) & firmware: GPMBs are functionally identical to BLDC boards, so they can give a good idea of the BLDC logic flow
- Brushless DC Motor (BLDC): motors
- Hall Effect Encoder (<LINK>)
- ODrive: the driver used to control the BLDC motors
- HindsightCAN Wiki: details the CAN packet used and format
- PSoC BLDC Hardware Spec, STM32 BLDC Hardware Spec, STM32 GPM Hardware Spec

- STM32 Resources

## Expected Behavior

*Primary Goals*

- [ ] Upon receiving a “Mode Set” CAN packet with packet ID 0x00 - 0x02
    - [ ] If the mode is defined in the HindsightCAN Wiki, then the board will switch to the mode given by the data bytes
    - [ ] If the mode is not defined, the mode is unchanged, and an error message is printed to the serial UART
    - [ ] If there is an error changing the mode, then an error message is printed to the serial UART
    - [ ] PWM and PID actions are disabled or enabled depending on the mode
- [ ] In PWM mode
    - [ ] If PID behaviors are requested, then an error message is printed to serial UART
    - [ ] Upon receiving a “Motor PWM & Dir Set” packet with ID 0x3, and PWM behaviors are enabled
        - [ ] The data provided by the data bytes will be scaled to 11-bits signed (-1024 to 1024)
        - [ ] A duty cycle of 50% should result in no motion, and duty cycles less than 50% should rotate the motor in opposite directions from duty cycles greater than 50%
            - [ ] Currently, 50% duty cycle is 30000, and the value written to PWM is <pwm>/0.1024+30000
        - [ ] The direction pin output will be set based on the sign of the data
        - [ ] After 1 second, the motor will stop moving
- [ ] In PID mode
    - [ ] If PWM behaviors are requested, then an error message is printed to serial UART
    - [ ] Upon receiving a “PID Position Target Set” CAN packet, set the PID target if all PID parameters have been set
        - [ ] If not all parameters have been set, then an error message is printed to serial UART
    - [ ] If the PID error is nonzero, use PID control to set the motor position to the target
- [ ] The following values can be set with their respective CAN packets
    - [ ] PID proportional, integral, and derivative constants
    - [ ] Encoder PPJR (pulses per rotation) with precision defined by Husky Robotics Firmware Standards

        - [ ] Potentiometer conversion minimum and maximum
        - [ ] Maximum PWM in PID control
        - [ ] Encoder bounds
    - [ ] The following data can be transmitted when requested by a “Telemetry Pull” CAN packet
        - [ ] Angular position
        - [ ] Raw ADC value
        - [ ] Limit switch status
        - [ ] Chip type
    - [ ] The feedback, whether potentiometer or encoder values, and stored position should reflect the movement of the motor based on given conversions
    - [ ] Encoder can be initialized via CAN packet
    - [ ] If one of two limit switches is pressed
        - [ ] Stop the motor
        - [ ] Send a limit switch alert CAN packet to the Jetson, noting which limit switch is pressed
        - [ ] Set the encoder offset if the bounds have been set

*Secondary Goals*

- [ ] Drive LEDs to show the board’s mode
- [ ] LED lights when the motor is being driven
- [ ] Standard CAN LED behavior
- [ ] Standard DBG LED behavior

*Extra Goals*

- [ ] Implement velocity PID to control the angular speed of the motor
    - [ ] Constants and maximum safety velocity can be set through CAN packets
    - [ ] Determine default speed and when to use default

# Minimum Firmware Test Cases

- [ ] Valid and invalid “Mode Set” CAN packets
- [ ] “PID Position Target Set” CAN packet in PWM mode
- [ ] “Motor PWM & Dir Set” CAN packet in PID mode
- [ ] In PWM mode
    - [ ] “Motor PWM & Dir Set” packet with positive value
    - [ ] “Motor PWM & Dir Set” packet with negative value
    - [ ] “Motor PWM & Dir Set” packet with zero

- [ ] In PID mode
    - [ ] “PID Position Target Set” packet with no initialization
    - [ ] “PID Position Target Set” packet with constants, conversion, and feedback initialized
- [ ] “Telemetry Pull” CAN packet for
    - [ ] Angular position
    - [ ] Raw ADC value
    - [ ] Limit switch status
    - [ ] Chip type
    - [ ] Unsupported data

## Recommended Initial Steps

- For the PSoC BLDC board, copy BLDC Rev4_v2 into a new folder in MotorUnit
- For the STM32 boards
    - Attempt to successfully output a motor PWM
    - Attempt to create CAN mailbox to receive and transmit CAN packets