Sigmar gear motor drive instruction manual

Contents

Version record

Version number	Date	Revisions/Notes
0.1	2023.12.5	Support for the first version of odrivetool
0.2	2023.12.15	Added instruction list with instruction classification
0.3	2023.12.19	Added Python, Arduino, ROS SDK
0.4	2023.12.23	Added switching description for USB and CAN compatibility
0.5	2023.12.29	Add actual cases of PC commissioning
0.6	2024.1.2	Added practical cases of CAN protocol and Python commissioning
0.7	2024.1.8	Added CAN interface 120R Match resistor switch option
0.8	2024.2.15	Added the second encoder related content and user zero setting
0.9	2024.2.23	Added CAN instructions to modify parameters and call interface functions
0.91	2024.3.12	Added driver download address for National download software
0.92	2024.3.22	Added a description of the default baud rate of CAN, and a description of the initial position range of the rotor under the action of the second encoder.
1.0	2024.3.26	Added motor temperature protection instructions
1.1	2024.7.2	Add error code table
1.2	2024.8.18	Added instructions for changing ids

Precautions

1. Please use according to the working parameters of this manual, otherwise it will cause irreversible damage to the product!
2. During the operation of the motor, please do a good job of the power supply overvoltage protection measures, so as not to damage the drive.
3. Before use, please check the parts in good condition. If the parts are missing or damaged, please contact technical support in time.
4. The drive has no anti-reverse connection capability, please refer to section 2.4.1 before connecting the power supply to ensure that the power supply is correct.
5. Do not touch the exposed part of the drive with your hands to avoid electrostatic damage!

Legal Notice

Before using this product, be sure to read this manual carefully and operate the product according to the contents. If the user violates the contents of the manual to use this product, resulting in any property damage, personal injury accident, the company does not assume any responsibility. Because this product is composed of many parts, do not allow children to touch this product, in order to avoid accidents. In order to prolong the service life of this product, do not use this product in high temperature and high pressure environment. This manual has been printed to the extent possible to include the function of the description and instructions. However, due to the continuous improvement of product functions, design changes, etc., there may still be discrepancies with the products purchased by users.

This manual may differ from the actual product in terms of color, appearance, etc. Please refer to the actual product. The Company may, at any time without notice, make necessary improvements and changes to typographical errors, inaccurate and up-to-date information in this manual, or make improvements to procedures and/or equipment. Such changes will be uploaded to a new version of this manual, which should be obtained by contacting Technical Support. All images are for function description reference only, please refer to physical objects.

After-sales policy

The after-sales service of this product is strictly in accordance with the Law of the People's Republic of China on the Protection of Consumer Rights and Interests and the Product Quality Law of the People's Republic of China. The after-sales service is as follows:

1. Warranty period and content

   1) Users who place orders on online channels to purchase this product can enjoy the return service without reason within 7 days from the day after signing. When returning the product, the user must present a valid proof of purchase and return the invoice. The user must ensure that the returned goods maintain the original quality and function, the appearance is intact, and the trademarks and various logos of the goods themselves and accessories are complete. If there are any gifts, they should be returned together. If the goods appear artificial damage, artificial disassembly, missing packing cases, missing parts, will not be returned. The logistics cost incurred during the return shall be borne by the user. If the user does not settle the logistics cost, it will be deducted from the refund amount according to the actual amount incurred. Refund the amount paid to the user within seven days from the date of receipt of the returned goods. The refund method is the same as the payment method. The specific arrival date may be affected by factors such as bank, payment institution, etc.

   2) Within 7 days after the user signs for the next day, non-human damage performance failure occurs, through the company's after-sales service center test and confirmation, for the user to handle the return business, the user must present a valid purchase voucher, and return the invoice. If there are gifts, they should be returned.

   3) 7 to 15 days from the next day after the user signed, non-human damage performance failure occurs, after the company's after-sales service center test and confirmation, for the user to replace the whole set of goods. After the replacement, the three guarantee period of the goods themselves is recalculated.

   4) 15 days to 365 days from the day after the user signed, after the company's after-sales service center test and confirmation, belongs to the quality of the product itself, can provide free maintenance services. The replacement of the faulty product belongs to the company. The faulty products will be returned as is. This product has been strictly tested after the factory, if there is not the quality of the product itself, we will have the right to refuse the user's return demand.

If the after-sales policy of this manual is inconsistent with the after-sales policy of the store, the after-sales policy of the store shall prevail.

2. Non-warranty Policy

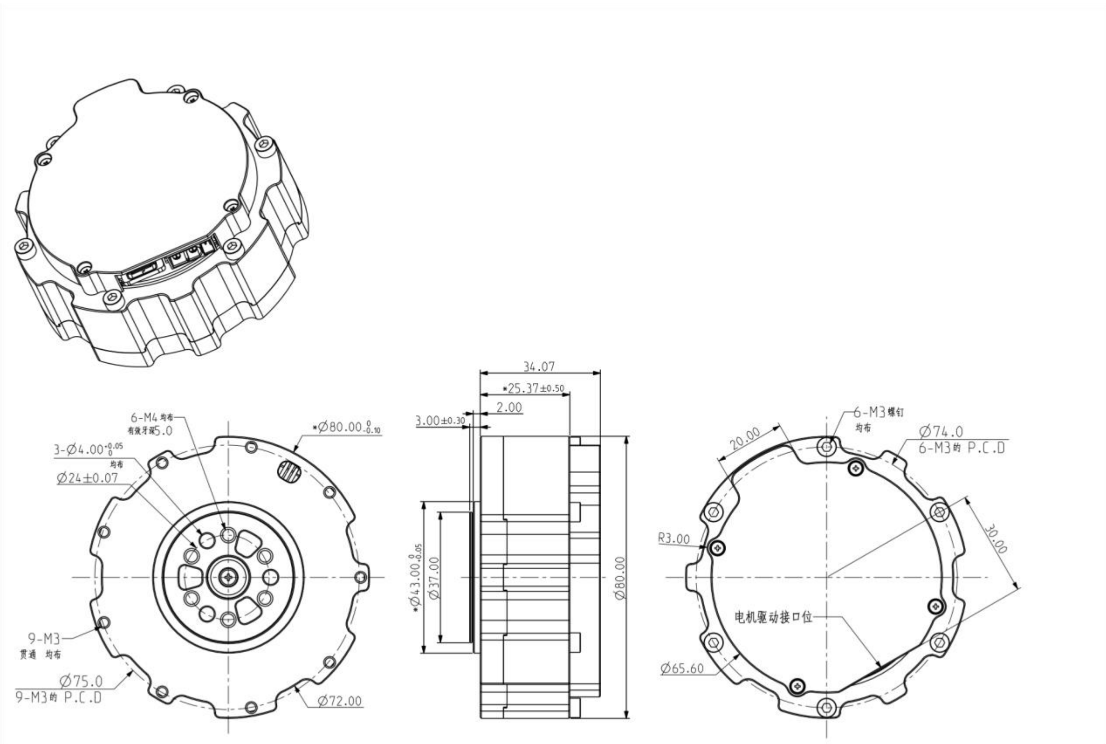
The following conditions are not covered by the warranty:

1) Exceeding the warranty period specified in the warranty terms.
2) Failure to follow instructions, damage caused by misuse of the product.
3) Damage caused by improper operation, maintenance, installation, modification, testing and other improper use.
4) Normal mechanical wear and tear caused by non-quality failure.
5) Damage caused by abnormal working conditions, including but not limited to falling, impact, liquid immersion, violent impact, etc.
6) Damage caused by natural disasters (such as flood, fire, lightning strike, earthquake, etc.) or incapacitated forces.
7) Damage caused by use of more than peak torque.
8) Not the company's original genuine products or can not provide legal proof of purchase.
9) Failure or damage caused by design, technology, manufacturing, quality and other problems of other non-products.
10) Damage caused by unauthorized disassembly of this product.

If the above situation occurs, the user shall pay the cost by himself.

1 Motor specifications

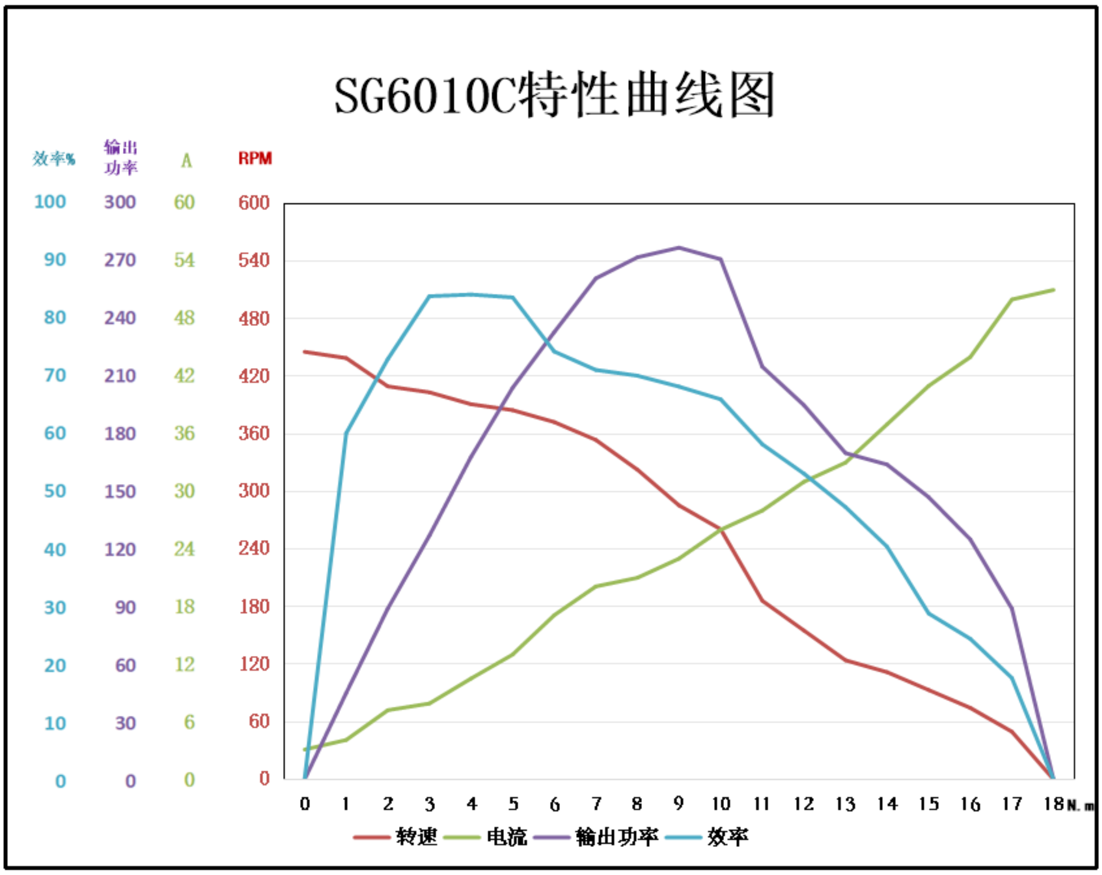
1.1 Drawings and dimensions



1.2 Electrical characteristics

Rated speed	170rpm±10%
Maximum RPM	490rpm±10%
Rated torque	6.5 N.m
Gridlock torque	18N.m
Rated current	16.5 A
Locked-rotor current	50A
No-load current	0.05 A
Interphase resistance	0.158 Ω
Phase to phase inductance	107 mu H
RPM constant	104rpm/v
Torque constant	0.042 N.M/A

The characteristic curve is as follows:

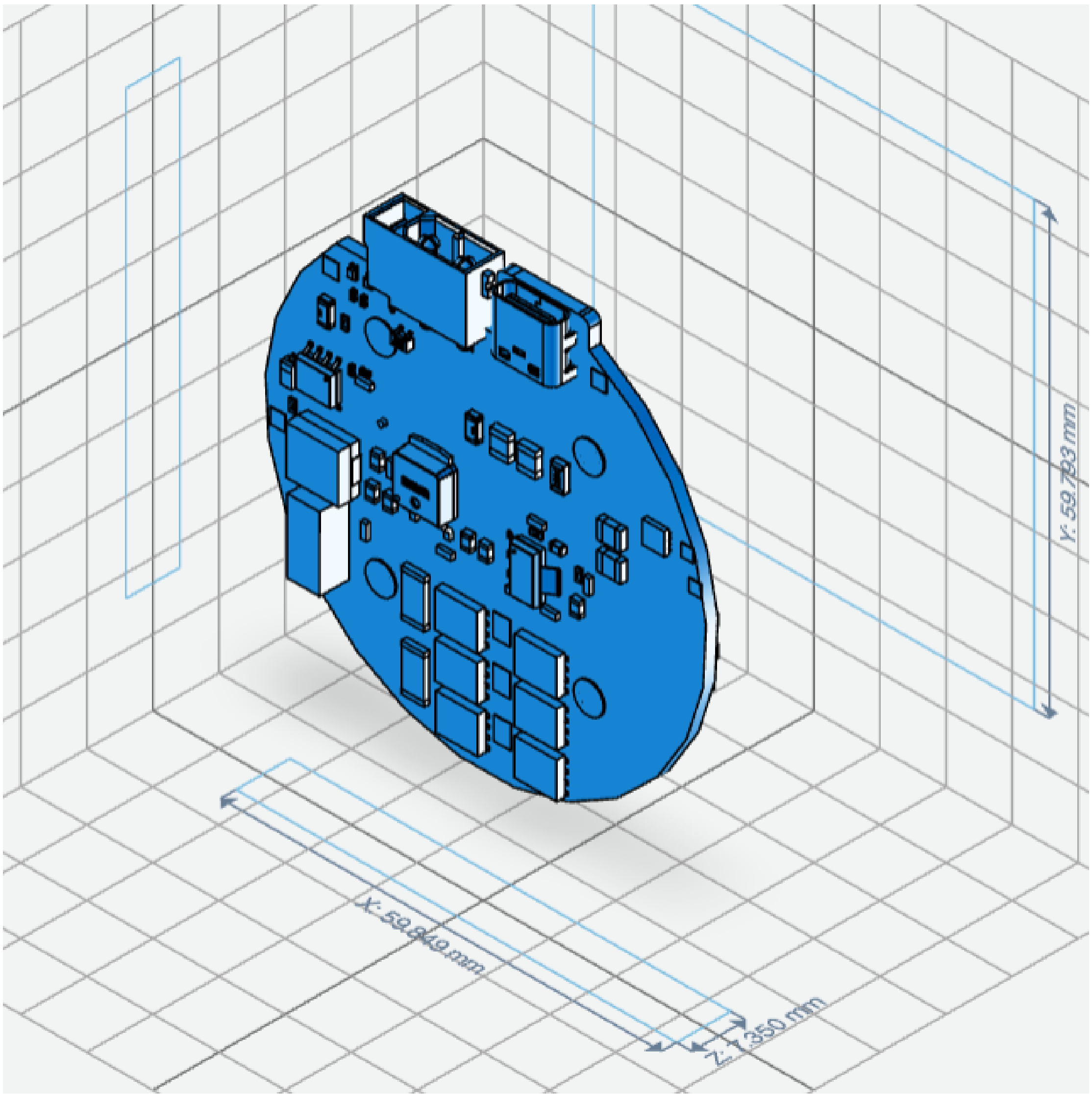


1.3 Mechanical characteristics

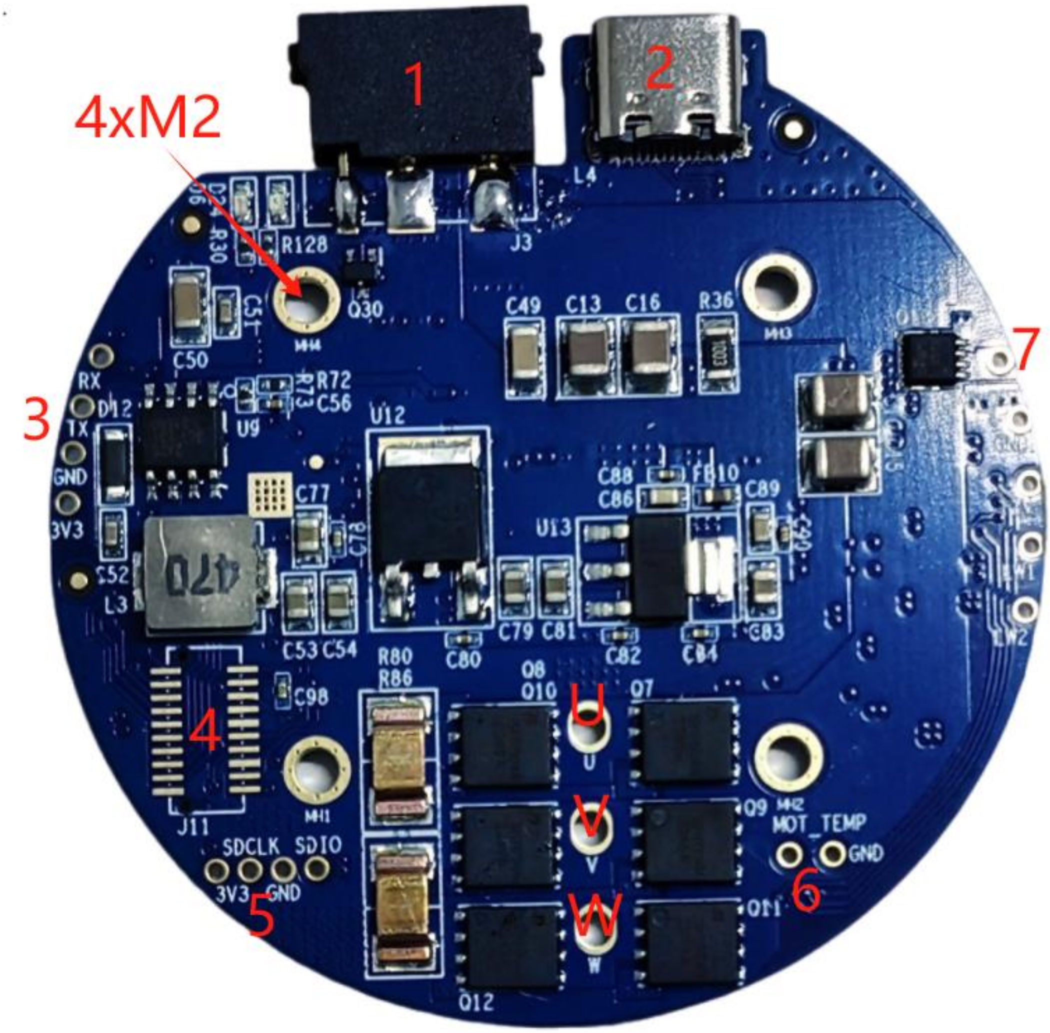
Weight	382g±3
Number of poles	10 pairs
Phase number	3 phases
Drive mode	FOC
Reduction ratio	9.67:1.

2 Drive Info

2.1 Appearance and 3D dimensions



2.2 Interface Overview



Interface serial number	definition
1	15~60V power supply and CAN communication integrated terminals
2	Type-C debugging interface and communication interface of upper computer
3	Second encoder interface (supports I2C and UART)
4	Interface expansion slot (expandable RS485, EtherCAT, airplane model, pulse direction, throttle control and other interfaces/protocols)
5	SWD debug and download interface
6	Motor Temperature Interface (NTC)
7	Hold brake/brake resistance interface, 12V power supply, min/Max limit switch interface

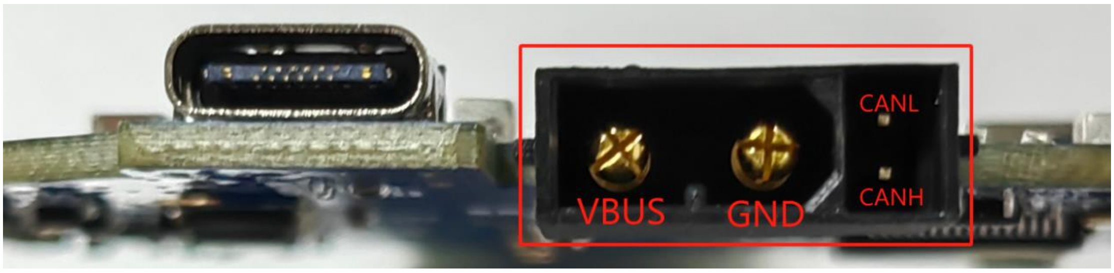
U/V/W	Weld holes for three-phase winding
4xM2	Mounting hole

2.3 Specifications

Rated voltage	15~48V DC
Min/Max voltage	12/72V DC
Rated current	6A
Maximum line current	30A
Maximum phase current	90A
Standby power	<10mA
Maximum baud rate of the CAN bus	1Mbps
Type-C rate	10Mbps
Encoder resolution	16bit (absolute value per turn)
Operating ambient temperature	-20℃ to 70℃
Alarm motor temperature	90 ° C (adjustable)
Alarm driver board temperature	90 ° C (adjustable)

2.4 Interface detailed definition

2.4.1 Power supply and CAN communication terminal



On-board terminal model XT30PB(2+2)-M, wire end model XT30(2+2)-F, brand manufacturer AMASS.

2.4.2 Type-C debugging interface

Type-C adopts standard data cable specifications, and is compatible with common PC or mobile phone Type-C data cables.

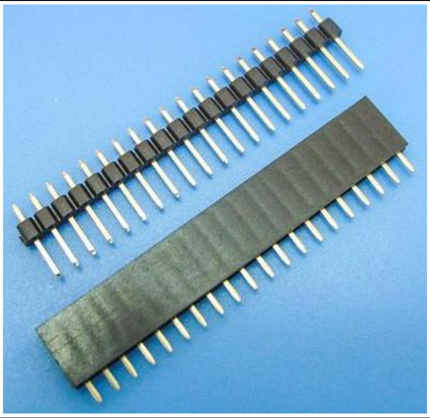
2.4.3 Second encoder interface

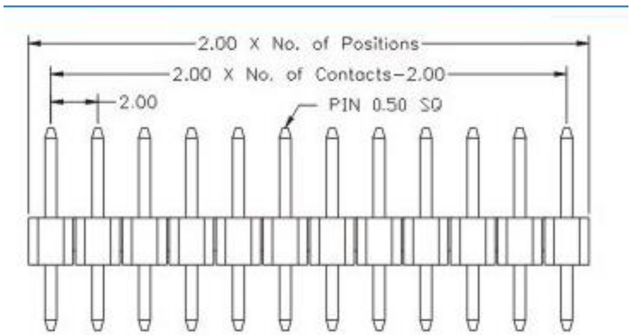
Pin holes spaced 2mm apart, users can weld 2mm straight single row pins, see 2.4.4.

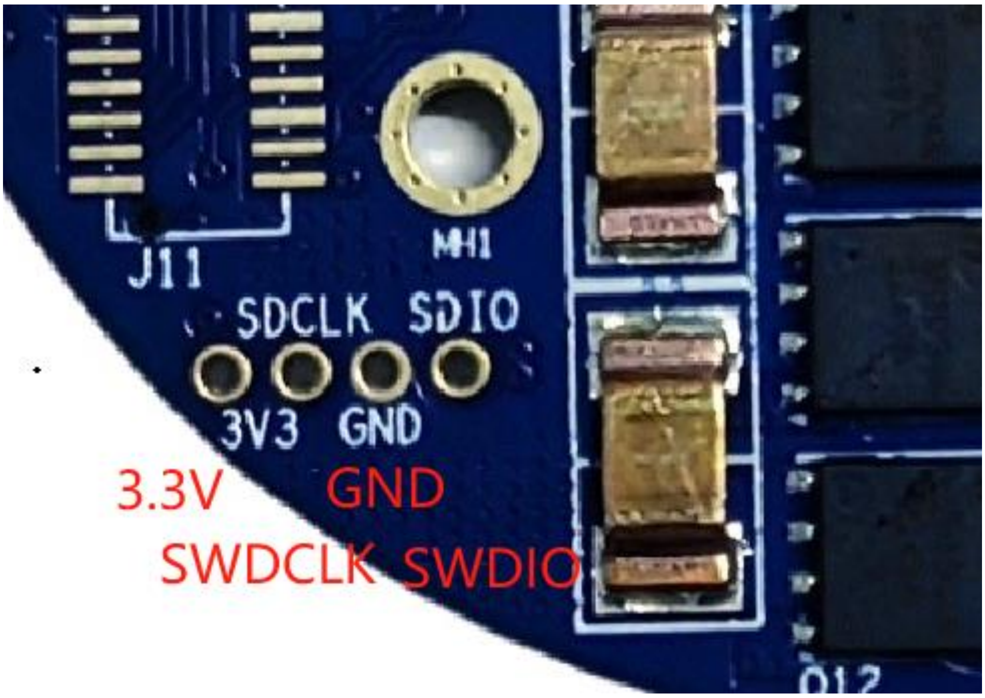
This interface can communicate with the second encoder via USART (TX/RX) or I2C (SCL/SDA).

2.4.4 SWD debug interface

Insert pin holes spaced 2mm apart, users can weld 2mm straight insert single row pins, as shown below:

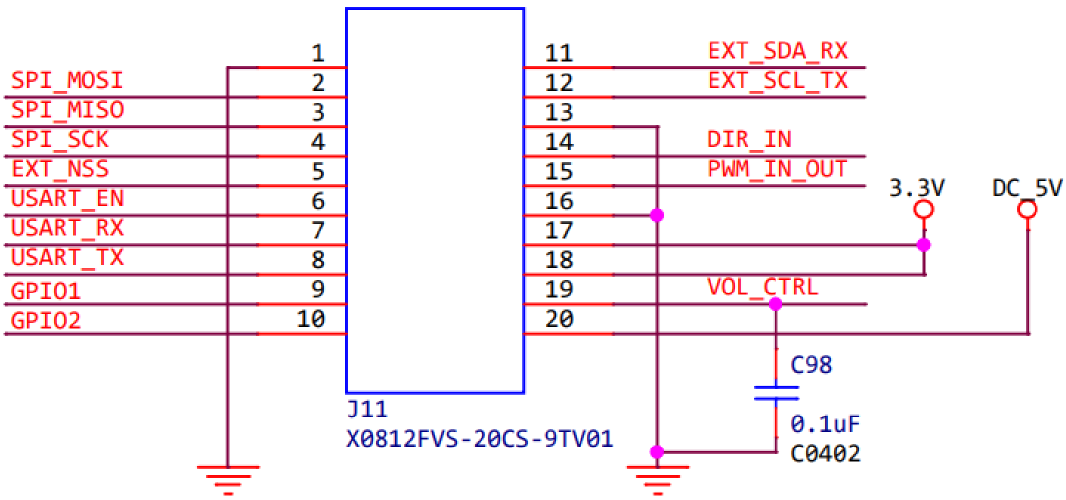






2.4.5 Interface expansion slot

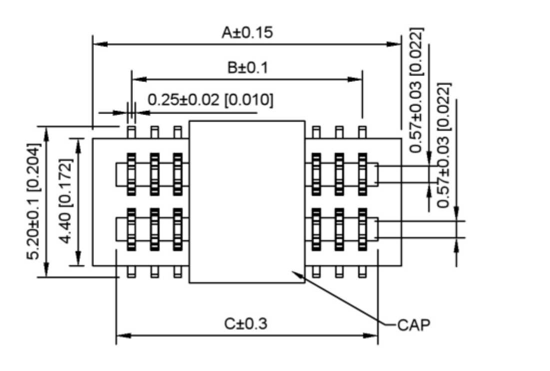
This slot is designed in the following ways to provide a wealth of inter-board expansion interfaces, and can be developed by a third party any expansion board:

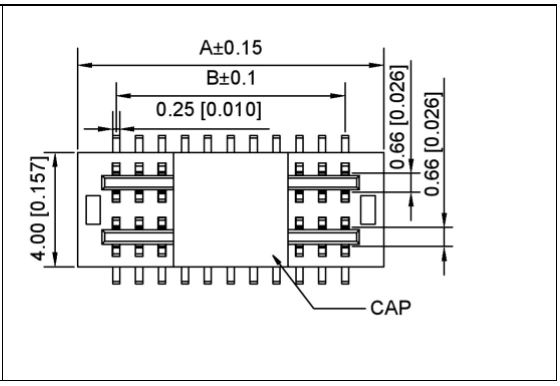


The third party can interact with the driver through SPI, USART, I2C, PWM, ADC, GPIO and other ways to achieve various expansion functions.

The onboard slot model is X0812FVS-20CS-9TV01 (female seat), and the expansion board slot model is X0812WVS-20AS-9TV01 (male seat), the brand manufacturer is Xingkun.

Female socket X0812FVS-20CS-9TV01	Female seat X0812WVS-20AS-9TV01





2.4.6 Motor temperature interface

10K NTC resistor built into motor, two leads welded to MOT_TEMP and GND, no line sequence.

2.4.7 Hold the brake/brake resistor interface

The top two welding holes in the 5-pin interface shown in the figure are the lock/brake resistance interfaces. The needle holes are 2mm apart. Users can weld 2mm straight single-row pins, see 2.4.4.

When it is a lock gate interface, the driver continues to output a current to this interface when powering up, so that the lock gate is open and the motor can operate normally. If the driver is powered off, the current stops, the lock gate is locked, and the motor will be locked at the power off position.

When the brake resistance interface, can be connected to an external brake resistance (or bleed resistance), when the back electromotive force is higher than the threshold voltage, through this brake resistance bleed current, to prevent the failure to brake emergency, or back electromotive force damage to the drive.

2.4.8 Limit switch interface

The driver provides two limit switch interfaces, and provides 12V power for the external limit switch. The pin holes are separated by 2mm. Users can weld 2mm straight single row pins, see 2.4.4.

Where LW1 is the minimum position limit switch, LW2 is the maximum position limit switch. The two wire switch or three wire NPN switch can be connected externally.

2.5 Main components and specifications

Serial number	Devices	Model/specification	Quantity
1	MCU	N32G455REL7	1
2	Driver chip	FD6288Q	1
3	Magnetic encoder chip	MA600, 16bit absolute value	1
4	MOSFET	JMSH1004NG, 100V/120A	6

3 Commissioning Instructions

3.1 Getting Started Guide

3.1.1 Preparation

To get the motor working, you'll need:

✓ A power supply

See Chapter 1 for voltage requirements for power supplies. Regulated power supplies or batteries are recommended. The question often asked by regular users is, how do I choose a power supply? Here are some simple suggestions, just for reference:

Select a few points of the power supply:
◆ Current requirements Generally, it should be at least greater than 5A. The exact number depends on the power requirements and voltage of the system.
◆ Voltage requirements Voltage demand depends on two factors: Kv of the motor and the maximum speed required by the system RPMmax, the maximum required power supply voltage can be referred to the formula: $$V_{max} = \frac{RPM_{max}}{K_v} \times 1.25$$ Where 1.25 is an empirical coefficient that gives the system a safe voltage threshold.
◆ Power requirements For power, simply say, depends on the maximum current value at the highest speed Imax, can refer to the formula: $$P_{max} = I_{max} \times \frac{RPM_{max}}{K_v} \times 1.25$$

✓ Power + communication interface cable

The SG6010C comes with 2+2 power communication sockets, please contact after-sales to recommend a suitable 2+2 cable. Please refer to 2.4.1, be sure to connect the positive and negative terminals of the power supply, otherwise there is the risk of burning the drive, as the drive has no anti-reverse connection capability. At the same time, please pay attention to the defined order of the 2 communication lines, such as the order of CANH/CANL, the wrong connection will lead to abnormal communication.

Warnings
✧ Be sure to avoid touching the communication bus with your hands to avoid electrostatic damage to the interface chip of the drive, especially in dry areas and dry seasons!
✧ Do not plug and unplug the power terminal with power on!
✧ Avoid using the circuit breaker as the switch of the power supply, there is a risk of destroying the power chip on the drive!
✧ Do not exceed 72V power supply voltage!

✓ Type-C data cable

In the early stages of commissioning, it is strongly recommended to test the motor with a USB Type-C data cable. Can use the most commonly used mobile phone Type-C data cable can be used, can not use only charging Type-C cable.

Please note that the Type-C cable does not power the drive, let alone drive the motor to turn!

✓ Power On

Please turn off the power supply, connect the power cable, and then turn on the power supply. Be sure not to plug and insert the power supply, or use the circuit breaker to control the single positive or negative cable to switch on and off. This will cause excessive startup current to burn the drive.

After power-on, you can plug and unplug the USB Type-C data cable at any time.

3.1.2 Get started with the odrivetool

Drive compatible odrive (https://github.com/odriverobotics/odrive.git), so use odrivetool as a PC to conduct test.

Follow these steps to install odrivetool:

➢ **Windows**

1) Install python

Go to the official python website https://www.python.org to download the latest python installer and follow the prompts to install it. Please do not download python versions from third party websites or the Microsoft Store.

2) Install the visual c++ generator

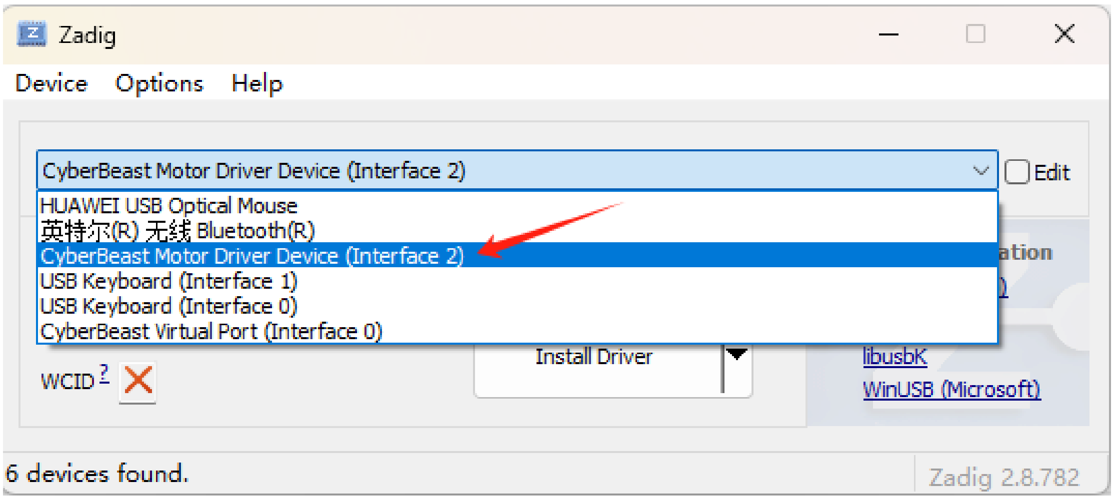
Install visual c + + generation tool https://visualstudio.microsoft.com/visual-cpp-build-tools/, check the installation process "desktop development using c + +", as shown in the figure below.

3) Install odrivetool

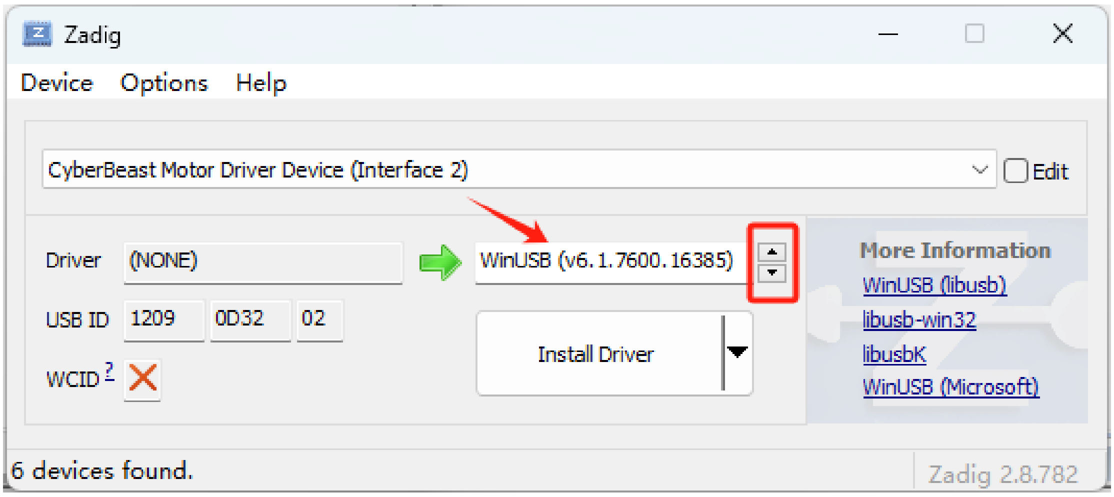
Run Windows PowerShell with Administrator, run pip install odrive in it and hit enter to install. Try again if something goes wrong halfway through. If you make repeated errors, restart your computer and try again.

4) Install a USB drive

Go to https://zadig.akeo.ie, download USB driver tool Zadig, connect the driver to the PC with a Type-C data cable, when the power on the driver lights up, turn on Zadig. Select "CyberBeast Motor Driver Device (Interface 2)" from the drop-down box:



Select a different USB Driver by clicking the up/down button, select the "WinUSB" driver version for this interface, and click "Install Driver" to install the driver for this interface:



- **WSL (Windows Subsystem for Linux)**

1) Install python/usb/odrivetool

If the user has WSL2 installed, go to the command line of WSL2 and follow these steps to install it (assuming user WSL has Ubuntu installed) :

```
sudo apt install python3 python3-pip
sudo apt install libusb-1.0-0
sudo pip install odrive numpy matplotlib
```

The first line of instructions installs python, the second line of instructions installs the usb driver, and the third line of instructions installs the odrivetool upper computer.

2) Connect the drive to WSL

Plug in Windows with a type-C data cable, Windows loads the driver for this USB port by default, and WSL does not. Need to the USB port is loaded into the WSL, please refer to the Microsoft documentation (https://learn.microsoft.com/zh-cn/windows/wsl/connect-usb).

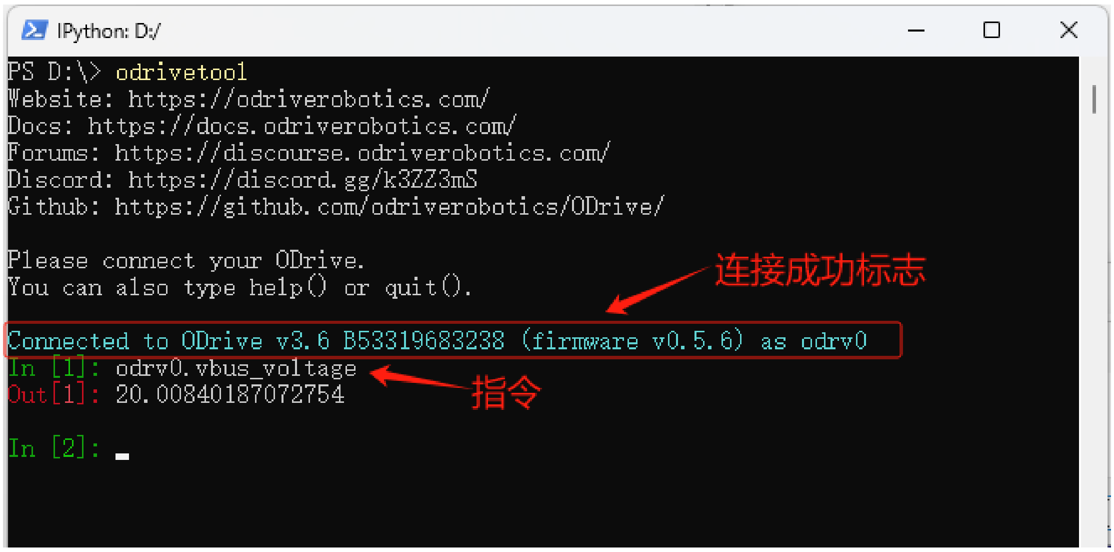
- **Ubuntu**

The installation process under Ubuntu is very similar to that under WSL, see the previous section.

3.1.3 Configure motor parameters properly

warn
It is recommended to read this section carefully, it is the key to successful operation, and avoid burning the motor!

After successfully installing odrivetool according to the previous section, power up the motor and connect the USB Type-C data line, run odrivetool in the shell (Windows PowerShell or Linux Terminal) (Type odrivetool and press enter), The following figure shows the successful connection under Windows (green font shows the connection information) :



Instruction example: odrv0 axis0. Controller. Input_vel, odrv0 represent the current connection of the motor, the default motor called odrv0 first connection, the second call odrv1, and so on; axis0 stands for the first motor the driver is connected to, and only one motor is supported in the current version. What this instruction means is to query the current speed control target value for the drive.

Operation Tips
◆ Often use the TAB key, there will be an instruction prompt, similar to the instruction prompt under linux, you can select the up and down left and right keys;
◆ The up or down key will display the history command
◆ When the command is entered, it will prompt the history similar command, and use the right button to complete it directly

➢ **Set key thresholds (limits)**

● Current threshold

```
odrv0.axis0.motor.config.current_lim = 30
```

The above instruction sets the current threshold to 30A. Note that this current threshold refers to the Q-axis current, not the supply current. This threshold directly limits the output torque. For SG6010C, please do not set this threshold above 50A!

Other factors that affect the current threshold

◆ Motor temperature

If can make the motor temperature protection (odrv0. Axis0. Motor. Motor_thermistor. Config. Enabled = 1), the current temperature can also affect Q axis of the motor current.

◆ Driving plate temperature

If can make drive plate temperature protection (odrv0. Axis0. Motor. Fet_thermistor. Config. Enabled = 1), the current temperature of the driven plate also affects Q shaft current.

The effects of the above two temperatures are very similar and can be expressed by the following formula:

$$I'_{lim} = \frac{T - T_l}{T_u - T_l} \times I_{lim}$$

Where is the final effective current threshold, is the configured current threshold, is the current temperature (motor temperature or drive board temperature), $I'_{lim} I_{lim} T T_l$ Is the set temperature lower limit (motor_thermistor.config.temp_limit_lower and fet_thermistor.config.temp_limit_lower), T_u Is the upper limit of the set temperature (motor_thermistor.temp_limit_upper and fet_thermistor.config.temp_limit_upper).

- Speed threshold

```
odrv0.axis0.controller.config.vel_limit = 30
```

System global speed limit, which the above instruction limits to 30turns /s (revolutions per second). Note that by default this speed limit does not work in pure

```
odrv0.axis0.controller.config.enable_torque_mode_vel_limit = 1
```

torque mode, but the following switches can be turned on to enable it:

- Calibrate current

This current defaults to 5A and is not changed by default, but can be reduced if the user's power supply current is low, otherwise a low voltage alarm will appear on time.

```
odrv0.axis0.motor.config.calibration_current = 2
```

➢ **Set key hardware parameters**

- **Maximum discharge/charge current**

Discharge current refers to the forward current that the power supply supplies to the drive and motor, and charge current refers to the reverse current flowing into the power supply. These two values are related to the power supply, please set to a suitable

```
odrv0.config.dc_max_negative_current
odrv0.config.dc_max_positive_current
```

value to avoid the power supply can not discharge resulting in voltage is pulled down, or damaged by the back electromotive force. However, please note that if these two values are set to a value with a small absolute value, it is easy to generate alarms.

- **The number of poles**

Pole_pairs is the number of magnetic poles in the motor rotor divided by 2. The user must set this value correctly for the calibration to be successful, otherwise there will

```
odrv0.axis0.motor.config.pole_pairs
```

be a calibration alarm.

- **Torque constant**

The torque constant is the torque produced by the motor divided by the Q-axis current, and its relationship to the Kv value of the motor is:.$Torque\_Constant =$

```
odrv0.axis0.motor.config.torque_constant
```

$8.27/K_v$

Whether the torque constant is correct does not affect the operation of the motor, but will affect the user to do torque control of the input value of the unit conversion, if the user wants to use the unit A rather than Nm to torque control, just set this value to 1.

- **Temperature sensing**

There are NTC temperature sensors inside the driver board and motor, and if enabled, the driver will control the output current (torque) according to the

```
# Motor temperature protection
odrv0.axis0.motor.motor_thermistor.config.enabled = 1
odrv0.axis0.motor.motor_thermistor.config.temp_limit_lower = 20
odrv0.axis0.motor.motor_thermistor.config.temp_limit_upper = 100
# Drive board temperature protection
odrv0.axis0.motor.fet_thermistor.config.enabled = 1
odrv0.axis0.motor.fet_thermistor.config.temp_limit_lower = 20
odrv0.axis0.motor.fet_thermistor.config.temp_limit_upper = 100
# Get temperature
odrv0.axis0.motor.motor_thermistor.temperature # motor temperature
odrv0.axis0.motor.fet_thermistor.temperature # Drive board temperature
```

temperature, thus protecting the driver and motor.

PID tuning

The following procedure can provide a reference for the user to adjust the PID

```
Odrv0. Axis0. Controller. Config. Pos_gain = 20.0
Odrv0. Axis0. Controller. Config. Vel_gain = 0.16
Odrv0. Axis0. Controller. Config. Vel_integrator_gain = 0.32
```

parameters:

1. Set the initial PID value
2. Set vel_integrator_gain to 0

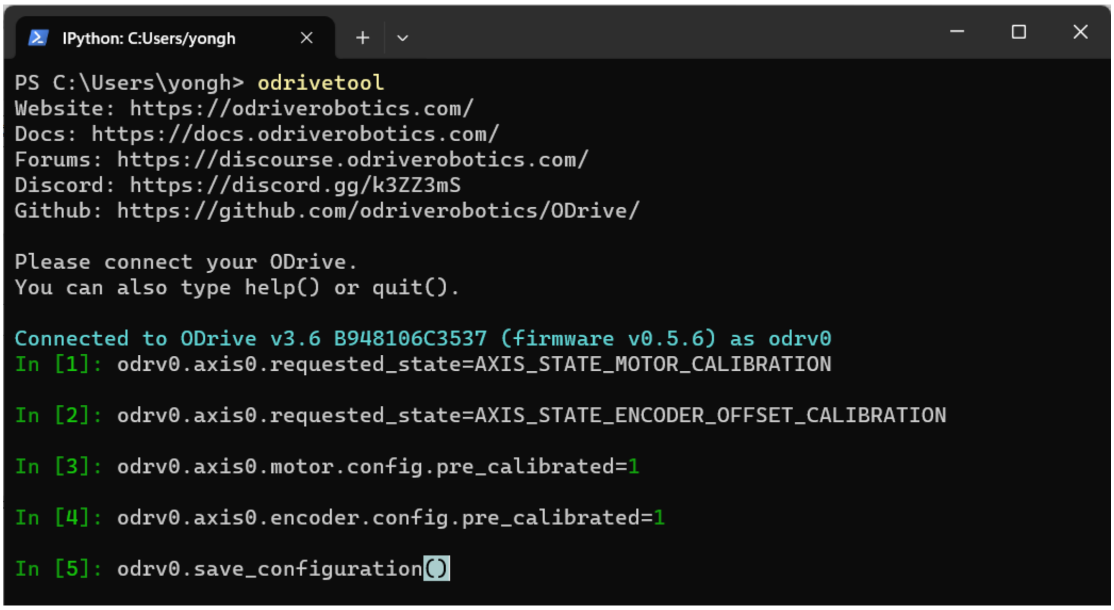
```
odrv0.axis0.controller.config.vel_integrator_gain=0
```

3. Adjust vel_gain method:
   1) Turn the motor with the speed control mode, if the rotation is not smooth, there is jitter or vibration, reduce the vel_gain until the rotation is smooth
   2) Then, increase the vel_gain by about 30% each time until there is a noticeable jitter
   3) At this point, reduce vel_gain by about 50% to stabilize
4. Adjust pos_gain method:
   1) Try turning the motor with position mode, if the rotation is not smooth, there is pull or vibration, reduce pos_gain until the rotation is smooth
   2) Then, increase pos_gain by about 30% each time until the position control is significantly overtuned (i.e. each time the position control motor exceeds the target position and then oscillates back to the target position).
   3) Then, gradually reduce pos_gain until the overmodulation disappears
5. After the four steps above, you can set vel_integrator_gain to 0.5*bandwidth*vel_gain, where bandwidth is the system control bandwidth. What is the control bandwidth? For example, from the user set the target position, to the motor really reach the target position time is 10ms, then the control bandwidth is 100Hz, then vel_integrator_gain=0.5*100*vel_gain.

In the above adjustment process, it is recommended to use the graphical means in 3.1.7 to view the adjustment effect in real time to avoid the error perceived by the naked eye.

3.1.4 Power-on calibration

When the user uses the micromotor for the first time, the motor and the encoder need to be calibrated. Before calibration, please fix the motor, or grip it with your hand, and the output shaft is unloaded. The calibration process is as follows:




```
odrv0.axis0.requested_state = AXIS_STATE_MOTOR_CALIBRATION
dump_errors(odrv0)
odrv0.axis0.requested_state =
AXIS_STATE_ENCODER_OFFSET_CALIBRATION
dump_errors(odrv0)
odrv0.axis0.motor.config.pre_calibrated = 1
odrv0.axis0.encoder.config.pre_calibrated = 1
```

Here's the explanation:

- Step 1: Motor parameter self-identification

Measure the phase resistance and phase inductance of the motor and hear a sharp "beep". The measurement results of the phase resistance and inductance can be viewed

```
odrv0.axis0.motor.config.phase_resistance
odrv0.axis0.motor.config.phase_inductance
```

by the following command:

- Step 2: Check the error code

Look at the system error code after the first step, if any red error code appears, you need to restart the motor and try again, or report after sale.

- Step 3: Calibrate the encoder

The encoder is calibrated, including the installation Angle of the encoder and the mechanical Angle of the motor, as well as the calibration of the encoder itself. During this calibration, the motor slowly turns forward an Angle and reverses an Angle. If it stops after only turning forward, it indicates that there is an error, please check the error code through step 4.

- Step 4: Check the error code

After the encoder calibration in step 3, look at your system's error code. A common error that occurs is ERROR_CPR_POLEPAIRS_MISMATCH, which means that the CPR of the encoder is set wrong, or the pole number of the motor is set wrong, please view/set

```
odrv0.axis0.encoder.config.cpr
odrv0.axis0.motor.config.pole_pairs
```

by the following command:

- Step 5: Write the motor calibration success sign
- Step 6: Write the encoder calibration success mark
- Step 7: Store the calibration results and restart

```
odrv0.axis0.encoder.config.pre_calibrated = 1
odrv0.axis0.motor.config.pre_calibrated = 1
odrv0.save_configuration()
```

3.1.5 Modify the ID

The drive ID is bus unique and ranges from 0 to 63. The default ID is 0. If the user has multiple motors in series on the bus, you will need to give each motor a different ID. The user can change the ID via USB or bus:

- USB modify ID

After the user has connected the drive through odrivetool, the ID can be modified

```
odrv0.axis0.config.can.node_id = xxx
```

by the following command:

- Bus change ID

See Set_Axis_Node_ID instruction message in 4.1.2.

3.1.6 Store and back up parameters

Be sure to store any parameters after they have been modified, otherwise the changes will be invalidated upon power failure or restart. The drive will restart after the

```
odrv0.save_configuration()
```

parameters are stored.

Backup parameters:

```
odrivetool backup-config "d:/test.json"
```

Where "d:\test.json" is the save path and file name that users can freely modify.

The instructions for parameter recovery are:

```
odrivetool restore-config "d:/test.json"
```

3.1.7 Four control modes

After the above preparation and parameter configuration, you can try to control the motor for different modes of rotation. The SG6010C supports position control, speed control, torque control, and motion control modes.

In the Position Control mode, it supports Filtered Position Control, Trajectory Control, Circular Position Control;

In the Velocity Control mode, it supports direct Velocity Control and Ramped Velocity Control;

In Torque Control mode, support direct Torque Control (Torque Control), ramp torque control (Ramped Torque Control).

Motion control mode is a control mode that integrates position, speed and torque, and is usually used in scenes that require strong instantaneous explosive force, such as robot knee joint. There are also users in the industry called it the MIT control mode, this name comes from the MIT open source mechanical dog, because it uses this motion control mode to control the motor.

In the subsequent detailed description of each control mode, this document uses USB control instructions as an example, but also CAN use communication protocols (such as CAN) to do the same control, the logic is consistent.

How to get the motor running?

◆ Start the motor (enter closed loop control)

In all subsequent control operations, to make the motor turn, it is necessary to put the motor into the closed-loop control state, with the following instructions:

```
odrv0.axis0.requested_state = 8
```

◆ Stop motor (enter idle state)

Users need to let the motor stop running, or want to modify parameters and save parameters, need to let the motor into the idle state first, the instructions are as follows:

```
odrv0.axis0.requested_state = 1
```

- **Filtered Position Control (Filtered Position Control)**

Filtered position control is recommended if the user wants to generate their own position curve and then send the position control instructions at a certain frequency, because this mode will smoothly link the instructions together. If the trapezoidal curve position control is used in this case, it is possible to make the motor rotation produce a sense of halting or graininess.

In this mode, the filter bandwidth needs to be adjusted according to the frequency of sending instructions. A good rule of thumb is to set the bandwidth to half the

```
odrv0.axis0.controller.config.input_filter_bandwidth = 25
```

instruction frequency (unit Hz), such as sending instructions at 50Hz frequency, then:

Enable filter position control:

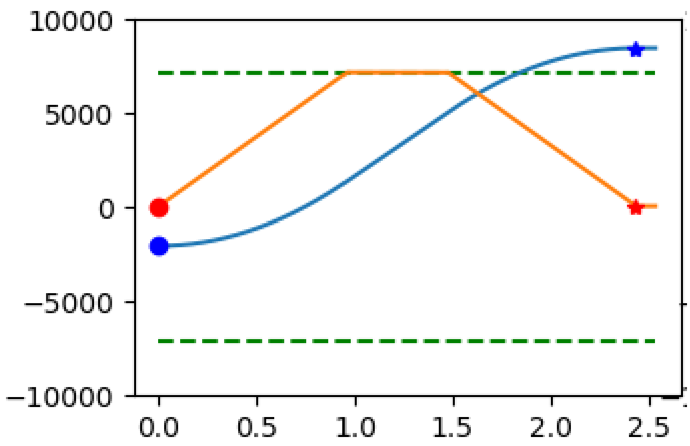
Then do position control:

```
odrv0.axis0.controller.config.control_mode = 3
odrv0.axis0.controller.config.input_mode = 3
```

```
Odrv0. Axis0. Controller. Input_pos = 10 # unit turns
```

- **Trajectory Control**

This mode allows the user to set acceleration, glide speed, and deceleration to control the smooth rotation of the motor from one position to another. The so-called "trapezoid" means that its speed curve looks trapezoid, as shown in the figure below, with orange for speed and blue for position:



Adjustable control parameters:

```
odrv0.axis0.trap_traj.config.vel_limit # Maximum coasting speed in turn/s
odrv0.axis0.trap_traj.config.accel_limit # Maximum acceleration in
turn/s^2
odrv0.axis0.trap_traj.config.decel_limit # maximum deceleration, in
```

Note that inertia x acceleration = torque, this value defaults to 0. This value can improve the system response, but is directly related to the load on the motor. The above four values are all greater than or equal to 0. Also note that the current thresholds and speed thresholds mentioned earlier still apply globally. For example, if the above maximum coasting speed is set to a value higher than vel_limit at the system level, the global vel_limit applies.

To enable trapezoidal curve control mode, first:

```
odrv0.axis0.controller.config.control_mode = 3
odrv0.axis0.controller.config.input_mode = 5
```

Then do position control:

```
Odrv0. Axis0. Controller. Input_pos = 10 # unit turns
```

- **(Circular Position Control)**

This mode is suitable for continuous incremental position control, such as the hub of the robot connected to rotate in one direction for a period of time, or the conveyor track has been running, if the usual position control mode is used, the target position will gradually increase a large value, resulting in inaccurate positioning errors due to floating point accuracy problems.

Enable:

```
odrv0.axis0.controller.config.circular_setpoints = 1
```

In this mode, each small step is in a single turn, and the input_pos range is [0, 1). If input_pos is increased beyond this range, it is converted to a value in a single turn. If the user wishes to exceed a single turn in a single step, the following parameters can be

```
odrv0.axis0.controller.config.circular_setpoint_range = <N>
```

set to a number greater than 1:

➢ **Direct Velocity Control**

This mode is the simplest speed control and is enabled as follows:

```
odrv0.axis0.controller.config.control_mode = 2
odrv0.axis0.controller.config.input_mode = 1
```

Then enter the target speed to control:

```
Odrv0. Axis0. Controller. Input_vel = 10 # unit turn/s
```

➢ **Ramp Velocity Control (Ramped Velocity Control)**

The ramp speed control mode, in which the speed is gradually increased to the target value according to a certain slope, is more gentle than the direct speed control

```
odrv0.axis0.controller.config.control_mode = 2
odrv0.axis0.controller.config.input_mode = 2
```

described above, enabled as follows:

Control the acceleration by adjusting the slope:

Then enter the target speed to control:

```
Odrv0. Axis0. Controller. Input_vel = 10 # unit turn/s
```

➢ **Direct torque Control (Toruqe Control)**

This is the simplest torque (current) control mode, enabled as follows:

```
odrv0.axis0.controller.config.control_mode = 1
odrv0.axis0.controller.config.input_mode = 1
```

The unit of torque control is Nm, and the unit of current in the driver firmware is A, so the torque constant also needs to be set so that the driver can convert Nm into current, thus driving the motor output torque on demand.

```
# torque constant is approximately equal to8.23/K_v
Odrv0. Axis0. Motor. Config. Torque_constant = 8.23/104
```

Then enter the target speed to control:

```
Odrv0. Axis0. Controller. Input_torque = 1.2 Nm # units
```

Also note that if the user wants to limit the maximum speed in torque mode, they can turn on enable_torque_mode_vel_limit and set vel_limit as follows:

```
odrv0.axis0.controller.config.enable_torque_mode_vel_limit = 1
Odrv0. Axis0. Controller. Config. Vel_limit = 30 # unit turn/s
```

- **Ramped Torque Control**

Ramp torque control is very similar to ramp speed control, enabled as follows:

```
odrv0.axis0.controller.config.control_mode = 1
odrv0.axis0.controller.config.input_mode = 6
```

Adjust the slope as follows:

```
Odrv0. Axis0. Controller. Config. Torque_ramp_rate = 0.1 # slope unit is
```

- **Motion Control (MIT Control)**

The motion control mode controls the motor movement to the target position by combining the control position, speed and torque, which can be expressed by the following formula:

$$T_{target} = K_p \times P_{diff} + K_d \times V_{diff} + T_{ff}$$

$$P_{diff} = P_{target} - P_{current}$$

$$V_{diff} = V_{target} - V_{current}$$

Where is the target torque, is the position error, is the speed error, is the position control gain, is the speed control gain (or damping coefficient), is the feedforward torque.$T_{target} P_{diff} V_{diff} K_p K_d T_{ff}$

Motion control mode is enabled as follows:

```
odrv0.axis0.controller.config.control_mode = 3
odrv0.axis0.controller.config.input_mode = 9
```

To adjust the gain:

Then do the motion control by typing input_pos, input_vel, input_torque:

Odrv0. Axis0. Controller. Input_pos = 5 # unit turns
Odrv0. Axis0. Controller. Input_mit_kp = < float > # position gain, Nm/turn

Please note that the position, speed, and torque input in USB control are all on the rotor side, while in MIT control with CAN, the position, speed, and torque in the protocol are on the output shaft side, in order to be consistent with the MIT open source protocol!

3.1.8 Upgrade

1) List of common instructions

After the connection is successful, the user can control the motor through the command, and obtain the parameters of the motor operation. The following table is the common instructions and commissioning process and instructions:

Types	Instructions	Instructions
Basic instructions	dump_errors(odrv0)	Print all error messages
	odrv0.clear_errors()	Clear all error messages
	odrv0.save_configuration()	After modifying the parameters, or after the motor automatically recognizes the parameters or calibration, be sure to execute this instruction to save the changes, otherwise all the changes will be lost after power off.
	odrv0.reboot()	Reboot drive
	odrv0.vbus_voltage	Get the power supply voltage (V)
	odrv0.ibus	Get the power supply current (A)
	odrv0.hw_version_major	Hardware major version number, SG6010C currently has a major version number of 3
	odrv0.hw_version_minor	Hardware minor version number, SG6010C The current minor version number is 8
	odrv0.hw_version_variant	For different type numbers in the same hardware configuration, SG6010C corresponds to the type number 1

	odrv0.can.config.r120_gpio_num	The 120R that controls the CAN interface matches the GPIO number of the resistance switch
	odrv0.can.config.enable_r120	120R matching resistance switch that controls the CAN interface
	odrv0.can.config.baud_rate	Baud_rate Settings for CAN
Parameter configuration instructions	odrv0.config.dc_bus_undervoltage_trip_level	Low voltage alarm threshold (V)
	odrv0.config.dc_bus_overvoltage_trip_level	Overvoltage alarm threshold (V)
	odrv0.config.dc_max_positive_current	Maximum line current (positive) (A)
	odrv0.config.dc_max_negative_current	Line current reverse charge maximum (negative) (A)
	odrv0.axis0.motor.config.resistance_calib_max_voltage	Motor parameter identification when the maximum voltage value, generally this value is slightly less than half of the power supply voltage, such as 24V power supply, can be set to 10
	odrv0.axis0.motor.config.calibration_current	Motor parameter identification maximum current value, this value can generally be set to 2~5A, not too large, not too small.
	odrv0.axis0.motor.config.torque_constant	Torque constant of motor (Nm/A)
	odrv0.axis0.min_endstop.config odrv0.axis0.max_endstop.config	Configuration of minimum (LW1)/Maximum (LW2) limit switch: enabled: Whether the switch is enabled or not gpio_num: indicates the corresponding IO number. Set the IO number of the minimum limit to 1 and the IO number of the maximum limit to 2
	odrv0.axis0.encoder.config.index_offset	The zero offset set by the user, which is the offset of the user's zero with respect to the encoder's zero. After setting this offset and saving the Settings, all user-entered position control target values are based on this user's zero.
	odrv0.axis0.motor.motor_thermistor.config odrv0.axis0.motor.fet_thermistor.config	To configure the motor temperature sensor: enabled: Whether to enable or not

		temp_limit_lower: indicates the lower limit of the temperature temp_limit_upper: indicates the upper limit of the temperature
	odrv0.axis0.motor.motor_thermistor.temperature	Motor temperature
	odrv0.axis0.motor.fet_thermistor.temperature	Drive temperature
Calibration instructions	odrv0.axis0.requested_state=4	Parameter identification of the motor, including identification of phase resistance, phase inductance, and calibration of the three-phase current balance. This process takes three to six seconds, and the motor emits a high-pitched sound. After the sound stops, or after 6 seconds there is no sound, run dump_errors(odrv0) to check for errors and confirm that there are no errors before performing any other operations.
	odrv0.axis0.requested_state=7	Calibrate the encoder. Before doing this, make sure the motor output shaft is free of any load and hold the motor in place by hand or other device. After this operation is performed, the motor will rotate forward and backward for a certain amount of time to identify and calibrate the encoder. After the motor stops, run dump_errors(odrv0) to check for errors and confirm that there are no errors before proceeding to other subsequent steps.
	odrv0.axis0.encoder.config.pre_calibrated=1	Write the precalibration successfully, indicating that you do not need to calibrate every power-on. This parameter can be written only after the above calibration is successful, otherwise the writing will fail.
	odrv0.axis0.controller.config.load_encoder_axis=0	Ensure that the current operating motor is the 0th motor. This operation is only necessary in BETA and is not valid in the production version.
	odrv0.axis0.requested_state=1	Stop the motor and go to idle

Control Instructions	odrv0.axis0.requested_state=8	Start the motor and enter the closed loop state
	odrv0.axis0.motor.config.current_lim	The maximum running line current of the motor (A), beyond which an overcurrent alarm will be reported. **Note that this value must not be greater than 100.**
	odrv0.axis0.controller.config.vel_limit	Maximum motor operating speed (turn/s), motor rotor speed above this value will report overspeed alarm.
	odrv0.axis0.controller.config.enable_vel_limit	Speed limit switch, the above vel_limit takes effect when True and does not take effect when False.
	odrv0.axis0.controller.config.control_mode	Control mode. 0: Voltage control 1: torque control 2: Speed control 3: Position control
	odrv0.axis0.controller.config.input_mode	Enter mode. Indicates how the control value entered by the user controls the motor operation: 0: idle 1: Direct control 2: Speed ramp 3: Position filtering 5: Trapezoidal curve 6: Torque ramp 9: Motion Control (MIT)
	odrv0.axis0.controller.config.vel_gain	P-value for speed loop PID control
	odrv0.axis0.controller.config.vel_integrator_gain	The I value of the speed loop PID control
	odrv0.axis0.controller.config.pos_gain	P-value of position loop PID control
	odrv0.axis0.controller.input_mit_kp	Motion control (MIT) position gain
	odrv0.axis0.controller.input_mit_kd	Motion control (MIT) Speed gain (damping coefficient)
	odrv0.axis0.controller.input_torque	Torque control target, or torque feedforward (Nm) for speed control/position control
	odrv0.axis0.controller.input_vel	Target for speed control, or position control for speed feedforward (turn/s)
	odrv0.axis0.controller.input_pos	The object of position control (turns)

	odrv0.axis0.encoder.set_linear_count ()	Sets the absolute position encoder, input 32-bit integer in parentheses, the absolute value of the integer need than odrv0. Axis0. Encoder. Config. CPR
	odrv0.axis0.trap_traj.config	Contains three parameters: ➢ accel_limit: Maximum acceleration (rev/s^2) ➢ decel_limit: Maximum deceleration (rev/s^2) ➢ vel_limit: maximum speed (rev/s) The three parameters in odrv0. Axis0. Controller. Config. Input_mode to trapezoid curve, adjust the deceleration effect of position control.
	odrv0.axis0.controller.config. input_filter_bandwidth	Location filter bandwidth, this parameter in odrv0 axis0. Controller. Config. Input_mode to location filter, adjust the deceleration effect of position control.

2) Graphical debugging

When commissioning the motor, if you need to monitor some running parameters in real time, you can use python's powerful calculation library and graphics library, as well as the high-speed throughput capacity of the Type-C interface to output motor parameters in real time.

1. Environment preparation

Install the compute library and graphics library:

```
pip install numpy matplotlib
```

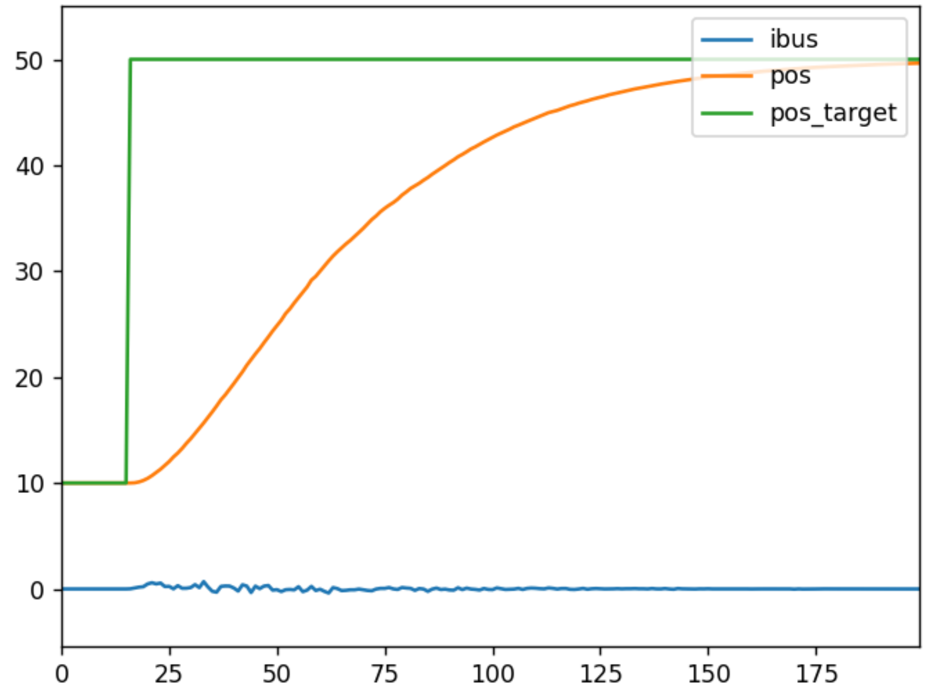
2. Graphical parameter output

In the odrivetool command line interface, tune up the graphics library and read any

```
start_liveplotter(lambda:[odrv0.ibus,odrv0.axis0.encoder.pos_estimate,
odrv0.axis0.controller.input_pos],["ibus","pos","p os_target"])
```

motor running indicators such as:

This command will call up a graphical interface, real-time output of the following three indicators: line current, position, target position. Next, the motor position control, you will see the real-time position control curve of the motor:



3) CAN match resistance switch

On the drive, there is already a 120 ohm impedance matching resistor onboard, the user can turn on or off as needed, example instructions are as follows:

```
odrv0.can.config.r120_gpio_num = 5
odrv0.can.config.enable_r120 = True
```

4) User zero configuration

By default, the position read by the user from the motor and the input value when doing position control are based on the zero of the absolute encoder on the drive. However, in the user scenario, the encoder's zero is not the user's zero most of the time, so the user needs to manually set this zero offset.

In general, the user can locate this zero point in two ways, either through the limit switch or by manually setting the offset, that is, the offset of the user's zero point

```
# After the user rotates to the desired user zero position either
manually or through position control:
odrv0.axis0.encoder.config.index_offset =
```

relative to the encoder's zero:

5) Limit switch

The drive supports two limit switches (LW1 and LW2), where LW1 is the minimum position, which is also the return to zero position, and LW2 is the maximum position. To

```
odrv0.axis0.min_endstop.config.enabled = True
odrv0.axis0.min_endstop.config.gpio_num = 1
odrv0.axis0.max_endstop.config.enabled = True
odrv0.axis0.max_endstop.config.gpio_num = 2
```

use two limit switches, use the following configuration:

When the limit switch is triggered, the system reports a MIN_ENDSTOP_PRESSED or MAX_ENDSTOP_PRESSED error. The upper computer can perform related operations.

Note that the limit switch function is not supported in hardware version 3.7.

3.2 Firmware Update Download

Firmware can be burned via the SWD interface (2.4.4) or Type-C interface (2.4.2), providing the following two ways:

3.2.1 National download software

1. USB (DFU) Burn Write

Please note that the national download software can be burned through the Type-C interface, but also through the SWD interface (only support JLink and DAP) burning, this section mainly takes the Type-C interface burning as an example.

First, download the burning software of USB driver (https://www.cyberbeast.cn/filedownload/789489) and install the corresponding system of driver; Then, download the burning software (https://cyberbeast.cn/filedownload/766844), into any directory, and run.

Then, connect the Type-C interface, go to odrivetool, and put the drive in DFU mode by executing the following command:

```
odrv0.enter_dfu_mode()
```

Finally, use the national burning software to burn, as shown in the following picture. Please note that after the burn is finished, please click "common operations" and then click "Reset" to restart the drive and connect the test through odrivetool normally.

2. SWD (JLink or DAP) burn write

Download using SWD is similar to DFU mode, but you need to connect through the SWD debugging interface (2.4.4) and select the appropriate debugging tool (JLink or DAP) in the image above.

3.2.2 pyocd

pyocd is the python version of openOCD, which can support STLink, JLink, DAP and other common debugging tools for erasing, burning, resetting and other operations. **Note that the drive must be connected with the SWD interface.** See 2.4.4 for line ordering of SWD. **There is 3.3V power supply in the SWD interface, please do not connect the wrong wire sequence, so as not to damage the drive!**

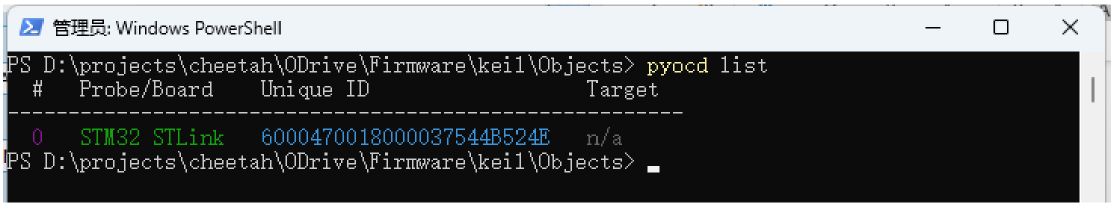
1. Install

```
pip install pyocd
```

2. Burn to Write

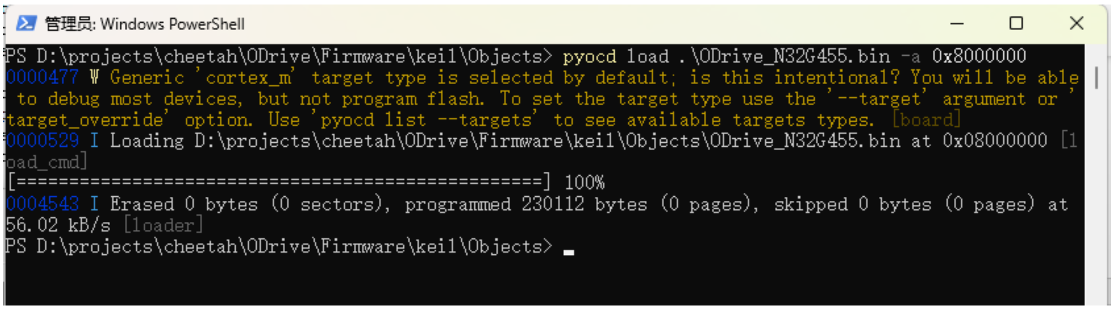
First, list the connected debugger tools:

```
pyocd list
```



Then, burn the bin file by executing the following command:

```
pyocd load .\ODrive_N32G455.bin -a 0x8000000
```



4 Integration Notes

4.1 CAN Protocol

The default communication interface is CAN, the maximum communication rate is 1Mbps (can be read and set through odrv0.can.config.baud_rate), the factory default rate is 500Kbps.

4.1.1 Protocol Frame Format

CAN communication uses standard frame format, data frame, 11-bit ID, 8-byte data, as shown in the following table (MSB on the left, LSB on the right) :

Data Field	CAN ID (11bits)		Data (8 bytes)
Segment	Bit10 ~ Bit5	Bit4 ~ Bit0	Byte0 ~ Byte7
Description	node_id	cmd_id	Communication data

- Node_id: represents the only ID of the motor on the bus, can be used in odrivetool odrv0. Axis0. Config. Can the node_id to read and Settings.
- cmd_id: instruction code, which represents the message type of the protocol, see this section of savings.
- Communication data: 8 bytes, the parameters carried in each message will be encoded as integers or floating-point numbers, byte order is small endian, The floating point number is carried out in accordance with the IEEE 754 standard coding (through https://www.h-schmidt.net/FloatConverter/IEEE754.html test code).

Take the Set_Input_Pos message described in 4.1.2 as an example, assuming that its three parameters are: Input_Pos=3.14, Vel_FF=1000 (representing 1rev/s), Torque_FF=5000 (representing 5Nm), and the CMD ID of the Set_Input_Pos message =0x00C, assuming that the node (node_id) of the drive is set to 0x05, then:

- 11 bit CAN ID=(0x05<<5)+0x0C=0xAC
- According to the description of Set_Input_Pos in 4.1.2, Input_Pos starts with the 0 th byte and is encoded as C3 F5 48 40 (floating point 3.14 is encoded as 32-bit 0x4048f5c3 by IEEE 754 standard). Vel_FF in the 4th byte beginning 2 bytes, encoded E8 03 (1000=0x03E8), Torque_FF in the 6th byte beginning 2 bytes, encoded 88 13 (5000=0x1388), then 8 bytes of communication data is:

Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
C3	F5	48	40	E8	03	88	13

4.1.2 Frame Messages

The following table lists all the available messages:

CMD ID	Name	Directions	Parameters
0x001	Heartbeat	Motor host→	Axis_Error Axis_State Motor_Flag Encoder_Flag Controller_Flag Traj_Done Life
0x002	Estop	Host motor→	
0x003	Get_Error	Motor host→	Error_Type
0x004	RxSdo	Motor host→	
0x005	TxSdo	Motor host→	
0x006	Set_Axis_Node_ID	Main engine motor→	Axis_Node_ID
0x007	Set_Axis_State	Main engine motor→	Axis_Requested_State
0x008	Mit_Control	Main motor→	
0x009	Get_Encoder_Estimates	Motor host→	Pos_Estimate Vel_Estimate
0x00A	Get_Encoder_Count	Motor host→	Shadow_Count Count_In_Cpr
0x00B	Set_Controller_Mode	Host motor→	Control_Mode Input_Mode
0x00C	Set_Input_Pos	Main engine motor→	Input_Pos Vel_FF Torque_FF
0x00D	Set_Input_Vel	Main engine motor→	Input_Vel Torque_FF
0x00E	Set_Input_Torque	Main engine motor→	Input_Torque
0x00F	Set_Limits	Main motor→	Velocity_Limit Current_Limit
0x010	Start_Anticogging	Main engine motor→	
0x011	Set_Traj_Vel_Limit	Main engine motor→	Traj_Vel_Limit
0x012	Set_Traj_Accel_Limits	Main motor→	Traj_Accel_Limit Traj_Decel_Limit

0x013	Set_Traj_Inertia	Main engine motor→	Traj_Inertia
0x014	Get_Iq	Motor host→	Iq_Setpoint Iq_Measured
0x015	Get_Sensorless_Estimates	Motor host→	Pos_Estimate Vel_Estimate
0x016	Reboot	Main engine motor→	
0x017	Get_Bus_Voltage_Current	Motor host→	Bus_Voltage Bus_Current
0x018	Clear_Errors	Host motor→	
0x019	Set_Linear_Count	Main engine motor→	Linear_Count
0x01A	Set_Pos_Gain	Main engine motor→	Pos_Gain
0x01B	Set_Vel_Gains	Main engine motor→	Vel_Gain Vel_Integrator_Gain
0x01C	Get_Torques	Motor host→	Torque_Setpoint Torque
0x01D	Get_Powers	Motor host→	Electrical_Power Mechanical_Power
0x01E	Disable_Can	Main motor→	
0x01F	Save_Configuration	Main motor→	

A detailed description of all the messages is as follows:

- Heartbeat

CMD ID: 0x001 (Motor host)→

Heartbeats with firmware versions less than (including) 0.5.11 have the following format:

Start byte	Name	Type	odrivetool Access
0	Axis_Error	uint32	odrv0.axis0.error
4	Axis_State	uint8	odrv0.axis0.current_state
5	Motor_Flag	uint8	1: odrv0.axis0.motor.error is not 0 0: odrv0.axis0.motor.error is0
6	Encoder_Flag	uint8	1: odrv0 axis0. Encoder. The error is not zero Zero: odrv0 axis0. Encoder. The error is 0
7	Controller_Flag	uint8	Bit7: odrv0 axis0. Controller. Trajectory_done bit0: 1: odrv0 axis0. Controller. The error is not zero Zero: odrv0 axis0. Controller. The error is 0

A heartbeat with a firmware version greater than (including) 0.5.12 has the following format:

Start byte	Name	Type	odrivetool Access
0	Axis_Error	uint32	odrv0.axis0.error
4	Axis_State	uint8	odrv0.axis0.current_state
5	Flags	uint8	bit0: odrv0.axis0.motor.error is0 Bit1: odrv0 axis0. Encoder. Whether the error is 0 Bit2: odrv0 axis0. Controller. Whether the error is 0 Bit7: odrv0 axis0. Controller. Trajectory_done, namely the position curve is completed
6	Reserved	uint8	Retain
7	Life	uint8	Lifetime of a periodic message, plus 1 for each heartbeat message. The value ranges from 0 to 255. If this lifetime is discontinuous, it indicates that heartbeat messages are lost, that is, communication is unstable.

The format for heartbeats with firmware versions greater than (or including) 0.5.13 is as follows:

Start byte	Name	Type	odrivetool Access
0	Axis_Error	uint32	odrv0.axis0.error
4	Axis_State	uint8	odrv0.axis0.current_state
5	Flags	uint8	bit0: odrv0.axis0.motor.error is0 Bit1: odrv0 axis0. Encoder. Whether the error is 0 Bit2: odrv0 axis0. Controller. Whether the error is 0 bit3: odrv0.error is 0 Bit7: odrv0 axis0. Controller. Trajectory_done, namely the position curve is completed
6	Reserved	uint8	Retain
7	Life	uint8	Lifetime of a periodic message, plus 1 for each heartbeat message. The value ranges

			from 0 to 255. If this lifetime is discontinuous, it indicates that heartbeat messages are lost, that is, communication is unstable.

- Estop

CMD ID: 0x002 (Host motor) No parameter No data.→

This command causes an emergency motor shutdown and reports an exception for ESTOP_REQUESTED.

- Get_Error

CMD ID: 0x003 (Motor host)→

Enter (Host motor) :→

Starting byte	Name	Type	Instructions
0	Error_Type	uint8	0: Get motor exception 1: Get the encoder exception 2: Get the noninductive exception 3: Get controller exception 4: Obtain system exception

Output (motor host) :→

Starting byte	name	Type	odrivetool Access
0	Error	uint32	Different input Error_Type: 0: odrv0.axis0.motor.error 1: odrv0. Axis0. Encoder. The error 2: odrv0.axis0.sensorless_estimater.error 3: odrv0. Axis0. Controller. The error 4: odrv0.error

- RxSdo

CMD ID: 0x004 (Host motor)→

Enter:

Starting byte	name	Type	Instructions

0	opcode	uint8	0: Read 1: Write
1	Endpoint_ID	uint16	Please download all the parameters and the interface function corresponding to the ID of the JSON file: https://www.cyberbeast.cn/filedownload/837298
3	Reserved	uint8	
4	Value	uint8[4]	It varies depending on the Endpoint_ID, as described in the JSON above. If Endpoint_ID corresponds to a read-write float value, the 4 bytes here are IEEE encoded float values. This value is written to the float value when opcode=1.

Output (when opcode=0 above) :

Start byte	Name	Type	Instructions
0	opcode	uint8	Fixed to 0
1	Endpoint_ID	uint16	Please download all the parameters and the interface function corresponding to the ID of the JSON file: https://www.cyberbeast.cn/filedownload/837298
3	Reserved	uint8	
4	Value	uint8[4]	It varies depending on the Endpoint_ID, as described in the JSON above. If Endpoint_ID corresponds to a readable uint32, the four bytes are small-endian bytes.

- TxSdo

CMD ID: 0x005 (Motor host)→

Same as RxSdo for opcode=1.

- Set_Axis_Node_ID

CMD ID: 0x006 (host motor)→

Start byte	Name	Type	odrivetool Access
0	Axis_Node_ID	uint32	odrv0.axis0.config.can.node_id

- Set_Axis_State

CMD ID: 0x007 (host motor)→

Start byte	Name	Type	odrivetool Access
0	Axis_Requested_State	uint32	odrv0.axis0.requested_state

- Mit_Control

CMD ID: 0x008

This is an implementation that emulates the MIT open source Motion Control Protocol (https://github.com/mit-biomimetics/Cheetah-Software).

Please note that the position, speed, and torque entered in the USB control refer to the rotor side, while the position, speed, and torque in the protocol refer to the output shaft side when using CAN for MIT control, in order to be consistent with the MIT open source protocol!

✓ Main engine motor→

CAN data frame bits	Meaning	Instructions
BYTE0 BYTE1	**Position**: A total of 16 bits, with BYTE0 being the high 8 bits and BYTE1 the low 8 bits The multi-turn position of the output shaft, in radians (RAD)	The actual **position** is double type, which needs to be converted to 16-bit int type, and the conversion process is: pos_int = (pos_double + 12.5)* 6535/25
BYTE2 BYTE3 BYTE4	**Speed**: Total 12 bits, BYTE2 for its high 8 bits, BYTE3[7-4] (high 4 bits) for its low 4 bits. Represents the angular velocity of the output axis in RAD/s **KP value**: Total 12 bits, BYTE3[3-0] (low 4 bits) for its high 4 bits and BYTE4 for its low 8 bits.	The actual **speed** is double and needs to be converted to 12-bit int. The conversion process is as follows: vel_int = (vel_double + 65) * 4095 / 130 The **KP value** is actually a double and needs to be converted to a 12-bit int. The conversion process is: kp_int = kp_double * 4095 / 500

BYTE5 BYTE6 BYTE7	**KD value**: A total of 12 bits, with BYTE5 being its high 8 bits and BYTE6[7-4] (high 4 bits) being its low 4 bits. **Torque**: Total 12 bits, BYTE6[3-0] (low 4 bits) for its high 4 bits and BYTE7 for its low 8 bits. The units are N.m.	The **KD value** is **actually** a double and needs to be converted to a 12-bit int. The conversion process is: kd_int = kd_double * 4095 / 5 The actual **moment** is a double and needs to be converted to a 12-bit int. The conversion process is: t_int = (t_double + 50)* 4095 / 100 The torque constant is measured in N.m/A

✓ Motor host→

CAN data frame bits	Meaning	Instructions
BYTE0	node id	Driver node id
BYTE1 BYTE2	**Position**: A total of 16 bits, with BYTE1 being the high 8 bits and BYTE2 the low 8 bits The multi-turn position of the output shaft, in radians (RAD)	The actual **position** is double and needs to be converted from 16-bit int. The conversion process is: pos_double = pos_int * 25/65,535-12.5
BYTE3 BYTE4 BYTE5	**Speed**: A total of 12 bits, with BYTE3 for its high 8 bits and BYTE4[7-4] (high 4 bits) for its low 4 bits. Represents the angular velocity of the output axis in RAD/s **Torque**: Total 12 bits, BYTE4[3-0] (low 4 bits) for its high 4 bits and	The actual **speed** is double and needs to be converted from 12-bit int. The conversion process is: vel_double = vel_int * 130 / 4095 – 65

	BYTE5 for its low 8 bits. The units are N.m.	The actual **torque** is double and needs to be converted from 12-bit int. The conversion process is: t_double = t_int * 100 / 4095 - 50 The torque constant is measured in N.m/A

➢ Get_Encoder_Estimates

CMD ID: 0x009 (Motor host)→

Start byte	Name	Type	Units	odrivetool access
0	Pos_Estimate	float32	rev	odrv0.axis0.encoder.pos_estimate
4	Vel_Estimate	float32	rev/s	odrv0.axis0.encoder.vel_estimate

➢ Get_Encoder_Count

CMD ID: 0x00A (motor host)→

Start byte	Name	Type	odrivetool Access
0	Shadow_Count	int32	odrv0.axis0.encoder.shadow_count
4	Count_In_Cpr	int32	odrv0.axis0.encoder.count_in_cpr

➢ Set_Controller_Mode

CMD ID: 0x00B (Host motor)→

Start byte	Name	Type	odrivetool Access
0	Control_Mode	uint32	odrv0.axis0.controller.config.control_mode
4	Input_Mode	uint32	odrv0.axis0.controller.config.input_mode

➢ Set_Input_Pos

CMD ID: 0x00C (Host motor)→

Start byte	Name	Type	Units	odrivetool access
0	Input_Pos	float32	rev	odrv0.axis0.controller.input_pos
4	Vel_FF	int16	0.001 rev/s	odrv0.axis0.controller.input_vel
6	Torque_FF	int16	0.001 Nm	odrv0.axis0.controller.input_torque

➢ Set_Input_Vel

CMD ID: 0x00D (host motor)→

Start byte	Name	Type	unit	odrivetool access
0	Input_Vel	float32	rev/s	odrv0.axis0.controller.input_vel
4	Torque_FF	float32	Nm	odrv0.axis0.controller.input_torque

➢ Set_Input_Torque

CMD ID: 0x00E (Host motor)→

Start byte	Name	type	Units	odrivetool access
0	Input_Torque	float32	Nm	odrv0.axis0.controller.input_torque

➢ Set_Limits

CMD ID: 0x00F (host motor)→

Start byte	Name	Type	Units	odrivetool access
0	Velocity_Limit	float32	rev/s	odrv0.axis0.controller.config.vel_limit
4	Current_Limit	float32	A	odrv0.axis0.motor.config.current_lim

➢ Start_Anticogging

CMD ID: 0x010 (Host motor)→

Perform torque ripple calibration.

➢ Set_Traj_Vel_Limit

CMD ID: 0x011 (host motor)→

Start byte	Name	type	Units	odrivetool access
0	Traj_Vel_Limit	float32	rev/s	odrv0.axis0.trap_traj.config.vel_limit

➢ Set_Traj_Accel_Limits

CMD ID: 0x012 (Host motor)→

Start byte	Name	Type	Units	odrivetool access
0	Traj_Accel_Limit	float32	rev/s^2	odrv0.axis0.trap_traj.config.accel_limit

4	Traj_Decel_Limit	float32	rev/s^2	odrv0.axis0.trap_traj.config.decel_limit

➢ Set_Traj_Inertia

CMD ID: 0x013 (host motor)→

Start byte	Name	Type	Units	odrivetool access
0	Traj_Inertia	float32	Nm/(rev/s^2)	odrv0.axis0.controller.config.inertia

➢ Get_Iq

CMD ID: 0x014 (Motor host)→

Start byte	Name	Type	Units	odrivetool access
0	Iq_Setpoint	float32	A	odrv0.axis0.motor.current_control.Idq_setpoint
4	Iq_Measured	float32	A	odrv0.axis0.motor.current_control.Iq_measured

➢ Get_Sensorless_Estimates

CMD ID: 0x015 (Motor host)→

Start byte	Name	Type	Units	odrivetool access
0	Pos_Estimate	float32	rev	odrv0.axis0.sensorless_estimator.pll_pos
4	Vel_Estimate	float32	rev/s	odrv0.axis0.sensorless_estimator.vel_estimate

➢ Reboot

CMD ID: 0x016 (Host motor)→

➢ Get_Bus_Voltage_Current

CMD ID: 0x017 (motor host)→

Start byte	Name	Type	Units	odrivetool access
0	Bus_Voltage	float32	V	odrv0.vbus_voltage
4	Bus_Current	float32	A	odrv0.ibus

➢ Clear_Errors

CMD ID: 0x018 (host motor)→

Clean up all errors and exceptions.

- Set_Linear_Count

CMD ID: 0x019 (host motor)→

Set the encoder absolute position.

Start byte	Name	Type	odrivetool Access
0	Linear_Count	int32	odrv0.axis0.encoder.set_linear_count()

- Set_Pos_Gain

CMD ID: 0x01A (Host motor)→

Start byte	Name	Type	Units	odrivetool access
0	Pos_Gain	float32	(rev/s)/rev	odrv0.axis0.controller.config.pos_gain

- Set_Vel_Gains

CMD ID: 0x01B (Host motor)→

Start byte	Name	Type	Units	odrivetool access
0	Vel_Gain	float32	Nm/(rev/s)	odrv0.axis0.controller.config.vel_gain
4	Vel_Integrator_Gain	float32	Nm/rev	odrv0. axis0.controller.config.vel_integrator_gain

- Get_Torques

CMD ID: 0x01C (Motor host)→

Start byte	Name	Type	odrivetool Access
0	Torque_Setpoint	float32	odrv0.axis0.controller.torque_setpoint
4	Torque	float32	None. Indicates the current torque value.

- Get_Powers

CMD ID: 0x01D (Motor host)→

Start byte	Name	Type	odrivetool Access

0	Electrical_Power	float32	odrv0.axis0.controller.electrical_power
4	Mechanical_Power	float32	odrv0.axis0.controller.mechanical_power

➢ Disable_Can

CMD ID: 0x01E (Host motor)→

Disable CAN, and restart the drive.

➢ Save_Configuration

CMD ID: 0x01F (host motor)→

Store the current configuration, take effect and restart.

4.1.3 CAN Protocol Actual Combat

1) Actual Combat: Power-on calibration

The sequence for sending CAN messages is as follows:

CAN ID	Frame type	Frame data	Instructions
0x007	Data Frames	04 00 00 00 00 00 00 00 00 00	Message: Set_Axis_State Parameter: 4 Calibrate the motor
0x007	Data Frames	07 00 00 00 00 00 00 00 00 00 00	Message: Set_Axis_State Parameter: 7 Calibrate the encoder

2) Actual Combat: Speed control

The sequence for sending CAN messages is as follows:

CAN ID	Frame type	Frame data	Instructions
0x00B	Data Frames	02 00 00 00 00 02 00 00 00 00 00	Message: Set_Controller_Mode Parameter: 2/2 Set control mode to Speed control and input mode to Speed ramp
0x007	Data Frames	08 00 00 00 00 00 00 00 00 00 00	Message: Set_Axis_State Parameter: 8

			Enter closed loop control
0x0D	Data Frames	00 00 20 41 00 00 00 00 00 00 00 00	Message: Set_Input_Vel Parameter: 10/0 Set target speed and torque feedforward where target speed is 10 (floating point: 0x41200000) and torque feedforward is 0 (floating point: 0x00000000)

3) Actual combat: Position control

The sequence of sending CAN messages is as follows:

CAN ID	Frame type	Frame data	Instructions
0x00B	Data Frames	03 00 00 00 00 03 00 00 00 00	Message: Set_Controller_Mode Parameter: 3/3 Set control mode to position control and input mode to position filtering
0x007	Data Frames	08 00 00 00 00 00 00 00 00 00 00	Message: Set_Axis_State Parameter: 8 Enter closed loop control
0x0C	Data Frames	CD CC 0C 40 00 00 00 00	Message: Set_Input_Pos Parameter: 2.2/0/0 Set target position, velocity feedforward and torque feedforward, where the target position is 2.2 (floating point: 0x400CCCCD) and torque feedforward and velocity feedforward are 0

4.1.4 CANOpen compatibility

Interoperable with CANOpen if the node ID is assigned properly. The following table lists the valid node ID combinations for CANopen and this protocol:

CANOpen node IDs	node IDs of this protocol
32… 127	0x10, 0x18, 0x20, 0x28
64… 127	0x10, 0x11, 0x18, 0x19, 0x20, 0x21, 0x28, 0x29

96… 127	0x10, 0x11, 0x12, 0x18, 0x19, 0x1A, 0x20, 0x21, 0x22, 0x28, 0x29, 0x2A

4.1.5 Periodic message

Users can configure the motor to periodically send messages to the upper computer, instead of sending request messages to the motor from the upper computer. Periodic messages can be turned on/off with a series of configurations under odrv0.axis0.config.can (values 0 for off, other values for cycle time in ms), as shown in the following table:

Messages	odrivetool configuration	Default
Heartbeat	odrv0.axis0.config.can.heartbeat_rate_ms	100
Get_Encoder_Estimates	odrv0.axis0.config.can.encoder_rate_ms	10
Get_Motor_Error	odrv0.axis0.config.can.motor_error_rate_ms	0
Get_Encoder_Error	odrv0.axis0.config.can.encoder_error_rate_ms	0
Get_Controller_Error	odrv0.axis0.config.can.controller_error_rate_ms	0
Get_Sensorless_Error	odrv0.axis0.config.can.sensorless_error_rate_ms	0
Get_Encoder_Count	odrv0.axis0.config.can.encoder_count_rate_ms	0
Get_Iq	odrv0.axis0.config.can.iq_rate_ms	0
Get_Sensorless_Estimates	odrv0.axis0.config.can.sensorless_rate_ms	0
Get_Bus_Voltage_Current	odrv0.axis0.config.can.bus_vi_rate_ms	0

By default, the first two cycle messages are factory turned on, so when the user monitors the CAN bus, they will see both messages for the set cycle broadcast. The user

```
odrv0.axis0.config.can.heartbeat_rate_ms = 0
odrv0.axis0.config.can.encoder_rate_ms = 0
```

can turn them off with the following instructions:

For details of the individual messages, see 4.1.2.

4.2 Python SDK

First install odrivetool (pip install - upgrade odrive) by following the steps in section 3.1. See 3.1.8 for python development using all the instructions described in this section.

Here are three examples:

4.2.1 Actual combat: Power-on calibration

```
import odrive
import time

odrv0 = odrive.find_any()
odrive.utils.dump_errors(odrv0)
odrv0.clear_errors()
odrv0.axis0.requested_state=odrive.utils.AxisState.MOTOR_CALIBRATION
time.sleep(5)
while (odrv0.axis0.current_state! = 1) :
  Time. Sleep (0.5)
odrive.utils.dump_errors(odrv0)
odrv0.axis0.requested_state=odrive.utils.AxisState.ENCODER_OFFSET_CALI
BRATION
time.sleep(6)
while (odrv0.axis0.current_state! = 1) :
  Time. Sleep (0.5)
odrive.utils.dump_errors(odrv0)
odrv0.axis0.motor.config.pre_calibrated=1
odrv0.axis0.encoder.config.pre_calibrated=1
odrv0.save_configuration()
```

4.2.2 Actual Combat: Speed control

```
import odrive
import time

odrv0 = odrive.find_any()
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.VE
LOCITY_CONTROL
odrv0.axis0.controller.config.input_mode=odrive.utils.InputMode.VEL_RA
MP
odrv0.axis0.controller.config.vel_ramp_rate=50
odrv0.axis0.requested_state=odrive.utils.AxisState.CLOSED_LOOP_CONTROL
odrv0.axis0.controller.input_vel=15
odrive.utils.dump_errors(odrv0)
time.sleep(5)
odrv0.axis0.controller.input_vel=0
```

4.2.3 Actual combat: Position control

```
import odrive

odrv0 = odrive.find_any()
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.PO
SITION_CONTROL
odrv0.axis0.controller.config.input_mode=odrive.utils.InputMode.POS_FI
LTER
odrv0.axis0.requested_state=odrive.utils.AxisState.CLOSED_LOOP_CONTROL
odrv0.axis0.controller.input_pos=10
```

4.2.4 Actual Combat: Data acquisition

In the process of R & D integration, users often need to collect motor operation data, such as collecting voltage and current changes, position and speed changes, etc. Python SDK integrates a powerful data capture ability, can use simple scripts to achieve massive operation data capture, making R & D and integration more simple.

The following code builds on the previous location control example by adding real-time location and current data scraping and saving the data into a csv file.

```
import odrive
import numpy as np

odrv0 = odrive.find_any()
cap =
odrive.utils.BulkCapture(lambda:[odrv0.axis0.motor.current_control.Iq_
measured,odrv0.axis0.encoder.pos_estimate],data_ra Te = 500, duration
= 2.5)
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.PO
SITION_CONTROL
odrv0.axis0.controller.config.input_mode=odrive.utils.InputMode.POS_FI
LTER
odrv0.axis0.requested_state=odrive.utils.AxisState.CLOSED_LOOP_CONTROL
odrv0.axis0.controller.input_pos=10
```

In the statements of BulkCapture, data_rate represents the sampling frequency (unit: hz) and duration represents the sampling time (unit: second). The lambda expression in this statement can be inserted into any mathematical operation to facilitate data analysis.

4.3 Arduino SDK

Users CAN use Arduino to control the motor through the CAN bus, the underlying protocol is described in 4.1. Compatible hardware/libraries:

- ✓ Arduino with a built-in CAN interface, such as Arduino UNO R4 Minima, Arduino UNO R4 WIFI, etc
- ✓ The Teensy development board with built-in CAN interface can be accessed using the adapted FlexCAN_T4 libraries (Teensy 4.0 and Teensy 4.1)
- ✓ Other Arduino-compatible boards CAN be accessed using the McP2515-based CAN expansion board

Here is an example showing how to configure the motor to respond to Arduino's position control instructions:

- ➢ Configuring the motor

In addition to the basic configuration of 3.1, configure the control to have a control bandwidth of 20rad/s (the transmission speed of the Arduino Uno is limited, so the control bandwidth need not be too high, if you use a faster Arduino, you can increase this bandwidth value) :

- ➢ Configure CAN

Configure CAN as follows:

- ➢ Install the ODriveArduino library

Follow these steps to install the OdriveArduino library (assuming the user already has the Arduino IDE installed) :

1) Open the Arduino IDE
2) Sketch -> Include Library -> Manage Libraries
3) Type in "ODriveArduino" to search
4) Click on the ODriveArduino library found in the search to install

- ➢ Arduino source code

```
#include <Arduino.h>
#include "ODriveCAN.h"

// Documentation for this example can be found here:
// https://docs.odriverobotics.com/v/latest/guides/arduino-can-guide.html
```

```
/* Configuration of example sketch -------------------------------------------*/

// CAN bus baudrate. Make sure this matches for every device on the bus
#define CAN_BAUDRATE 500000

// ODrive node_id for odrv0
#define ODRV0_NODE_ID 0

// Uncomment below the line that corresponds to your hardware.
// See also "Board-specific settings" to adapt the details for your hardware
setup.

// #define IS_TEENSY_BUILTIN // Teensy boards with built-in CAN interface (e.g.
Teensy 4.1). See below to select which interface to use.
// #define IS_ARDUINO_BUILTIN // Arduino boards with built-in CAN interface (e.g.
Arduino Uno R4 Minima)
// #define IS_MCP2515 // Any board with external MCP2515 based extension module.
See below to configure the module.



/* Board-specific includes ---------------------------------------------------*/

#if defined(IS_TEENSY_BUILTIN) + defined(IS_ARDUINO_BUILTIN) +
defined(IS_MCP2515) ! = 1
#warning "Select exactly one hardware option at the top of this file."

#if CAN_HOWMANY > 0 || CANFD_HOWMANY > 0
#define IS_ARDUINO_BUILTIN
#warning "guessing that this uses HardwareCAN"
#else
#error "cannot guess hardware version"
#endif

#endif

#ifdef IS_ARDUINO_BUILTIN
// See https://github.com/arduino/ArduinoCore-API/blob/master/api/HardwareCAN.h
// and https://github.com/arduino/ArduinoCore-
renesas/tree/main/libraries/Arduino_CAN

#include <Arduino_CAN.h>
#include <ODriveHardwareCAN.hpp>
#endif // IS_ARDUINO_BUILTIN
```

```
#ifdef IS_MCP2515
// See https://github.com/sandeepmistry/arduino-CAN/
#include "MCP2515.h"
#include "ODriveMCPCAN.hpp"
#endif // IS_MCP2515

#ifdef IS_TEENSY_BUILTIN
// See https://github.com/tonton81/FlexCAN_T4
// clone https://github.com/tonton81/FlexCAN_T4.git into /src
#include <FlexCAN_T4.h>
#include "ODriveFlexCAN.hpp"
struct ODriveStatus; // hack to prevent teensy compile error
#endif // IS_TEENSY_BUILTIN





/* Board-specific settings ---------------------------------------------------*/



/* Teensy */

#ifdef IS_TEENSY_BUILTIN

FlexCAN_T4<CAN1, RX_SIZE_256, TX_SIZE_16> can_intf;

bool setupCan() {
  can_intf.begin();
  can_intf.setBaudRate(CAN_BAUDRATE);
  can_intf.setMaxMB(16);
  can_intf.enableFIFO();
  can_intf.enableFIFOInterrupt();
  can_intf.onReceive(onCanMessage);
  return true;
}

#endif // IS_TEENSY_BUILTIN



/* MCP2515-based extension modules -*/

#ifdef IS_MCP2515

MCP2515Class& can_intf = CAN;
```

```
// chip select pin used for the MCP2515
#define MCP2515_CS 10

// interrupt pin used for the MCP2515
// NOTE: not all Arduino pins are interruptable, check the documentation for your
board!
#define MCP2515_INT 2

// freqeuncy of the crystal oscillator on the MCP2515 breakout board.
// common values are: 16 MHz, 12 MHz, 8 MHz
#define MCP2515_CLK_HZ 8000000


static inline void receiveCallback(int packet_size) {
  if (packet_size > 8) {
    return; // not supported
  }
  CanMsg msg = {.id = (unsigned int)CAN.packetId(), .len = (uint8_t)packet_size};
  CAN.readBytes(msg.buffer, packet_size);
  onCanMessage(msg);
}

bool setupCan() {
  // configure and initialize the CAN bus interface
  CAN.setPins(MCP2515_CS, MCP2515_INT);
  CAN.setClockFrequency(MCP2515_CLK_HZ);
  if (! CAN.begin(CAN_BAUDRATE)) {
    return false;
  }

  CAN.onReceive(receiveCallback);
  return true;
}

#endif // IS_MCP2515


/* Arduinos with built-in CAN */

#ifdef IS_ARDUINO_BUILTIN

HardwareCAN& can_intf = CAN;
```

```
bool setupCan() {
  return can_intf.begin((CanBitRate)CAN_BAUDRATE);
}

#endif



/* Example sketch ------------------------------------------------------------*/

// Instantiate ODrive objects
ODriveCAN odrv0(wrap_can_intf(can_intf), ODRV0_NODE_ID); // Standard CAN message
ID
ODriveCAN* odrives[] = {&odrv0}; // Make sure all ODriveCAN instances are
accounted for here

struct ODriveUserData {
  Heartbeat_msg_t last_heartbeat;
  bool received_heartbeat = false;
  Get_Encoder_Estimates_msg_t last_feedback;
  bool received_feedback = false;
};

// Keep some application-specific user data for every ODrive.
ODriveUserData odrv0_user_data;

// Called every time a Heartbeat message arrives from the ODrive
void onHeartbeat(Heartbeat_msg_t& msg, void* user_data) {
  ODriveUserData* odrv_user_data = static_cast<ODriveUserData*>(user_data);
  odrv_user_data->last_heartbeat = msg;
  odrv_user_data->received_heartbeat = true;
}

// Called every time a feedback message arrives from the ODrive
void onFeedback(Get_Encoder_Estimates_msg_t& msg, void* user_data) {
  ODriveUserData* odrv_user_data = static_cast<ODriveUserData*>(user_data);
  odrv_user_data->last_feedback = msg;
  odrv_user_data->received_feedback = true;
}

// Called for every message that arrives on the CAN bus
void onCanMessage(const CanMsg& msg) {
  for (auto odrive: odrives) {
    onReceive(msg, *odrive);
  }
```

```
}

void setup() {
  Serial.begin(115200);

  // Wait for up to 3 seconds for the serial port to be opened on the PC side.
  // If no PC connects, continue anyway.
  for (int i = 0; i < 30 && ! Serial; ++i) {
    delay(100);
  }
  delay(200);


  Serial.println("Starting ODriveCAN demo");

  // Register callbacks for the heartbeat and encoder feedback messages
  odrv0.onFeedback(onFeedback, &odrv0_user_data);
  odrv0.onStatus(onHeartbeat, &odrv0_user_data);

  // Configure and initialize the CAN bus interface. This function depends on
  // your hardware and the CAN stack that you're using.
  if (! setupCan()) {
    Serial.println("CAN failed to initialize: reset required");
    while (true); // spin indefinitely
  }

  Serial.println("Waiting for ODrive..." );
  while (! odrv0_user_data.received_heartbeat) {
    pumpEvents(can_intf);
    delay(100);
  }

  Serial.println("found ODrive");

  // request bus voltage and current (1sec timeout)
  Serial.println("attempting to read bus voltage and current");
  Get_Bus_Voltage_Current_msg_t vbus;
  if (! odrv0.request(vbus, 1)) {
    Serial.println("vbus request failed!" );
    while (true); // spin indefinitely
  }

  Serial.print("DC voltage [V]: ");
  Serial.println(vbus.Bus_Voltage);
```

```
  Serial.print("DC current [A]: ");
  Serial.println(vbus.Bus_Current);

  Serial.println("Enabling closed loop control..." );
  while (odrv0_user_data.last_heartbeat.Axis_State ! =
ODriveAxisState::AXIS_STATE_CLOSED_LOOP_CONTROL) {
    odrv0.clearErrors();
    delay(1);
    odrv0.setState(ODriveAxisState::AXIS_STATE_CLOSED_LOOP_CONTROL);

    // Pump events for 150ms. This delay is needed for two reasons;
    // 1. If there is an error condition, such as missing DC power, the ODrive
might
    //    briefly attempt to enter CLOSED_LOOP_CONTROL state, so we can't rely
    //    on the first heartbeat response, so we want to receive at least two
    //    heartbeats (100ms default interval).
    // 2. If the bus is congested, the setState command won't get through
    //    immediately but can be delayed.
    for (int i = 0; i < 15; ++i) {
      delay(10);
      pumpEvents(can_intf);
    }
  }

  Serial.println("ODrive running!" );
}

void loop() {
  pumpEvents(can_intf); // This is required on some platforms to handle incoming
feedback CAN messages

  float SINE_PERIOD = 2.0f; // Period of the position command sine wave in
seconds

  float t = 0.001 * millis();

  float phase = t * (TWO_PI / SINE_PERIOD);

  odrv0.setPosition(
    sin(phase), // position
    cos(phase) * (TWO_PI / SINE_PERIOD) // velocity feedforward (optional)
  );

  // print position and velocity for Serial Plotter
```

```
  if (odrv0_user_data.received_feedback) {
    Get_Encoder_Estimates_msg_t feedback = odrv0_user_data.last_feedback;
    odrv0_user_data.received_feedback = false;
    Serial.print("odrv0-pos:");
    Serial.print(feedback.Pos_Estimate);
    Serial.print(",");
    Serial.print("odrv0-vel:");
    Serial.println(feedback.Vel_Estimate);
  }
}
```

4.4 ROS SDK

The following steps are tested and validated on Ubuntu 23.04 and ROS2 Iron, but are not supported on MAC and Windows platforms, and are not validated on other ROS2 versions, and will need to be corrected to work.

4.4.1 Install the odrive_can package

1. Create a new ROS2 workspace (see https://docs.ros.org/en/iron/index.html)
2. With git clone https://github.com/odriverobotics/odrive_can to download code to the SRC directory in the workspace directory
3. In terminal go to the root directory of workspace and run:

```
colcon build --packages-select odrive_can
```

4. Environment preparation before running:

```
source ./install/setup.bash
```

5. Run routine node:

```
ros2 launch odrive_can example_launch.yaml
```

4.4.2 Call services and view messages

Assume that the above odrive_can_node node is running in namespace odrive_axis0 (set in./launch/example_launch.yaml). Once the node in 4.4.1 is running, you can view the published topic messages, such as:

```
ros2 topic echo /odrive_axis0/controller_status
ros2 topic echo /odrive_axis0/odrive_status
```

And call the open service interface, such as the following call to start the motor calibration:

```
ros2 service call /odrive_axis0/request_axis_state
/odrive_can/srv/AxisState "{axis_requested_state: 4}"
```

5 Frequently asked Questions and exception codes (to be updated)

5.1 Frequently Asked Questions (FAQ)

5.2 Exception codes

Error category	Error Code	odrivetool display	Description
System Exception	0x00000002	DC_BUS_UNDER_VOLTAGE	The PSU voltage is too low
	0x00000004	DC_BUS_OVER_VOLTAGE	The power supply voltage is too high.
	0x00000008	DC_BUS_OVER_REGEN_CURRENT	The power supply reverse (charge) current is too high
	0x00000010	DC_BUS_OVER_CURRENT	The forward (discharge) current of the power supply is too high
Abnormal drive	0x00000001	INVALID_STATE	Drive status error
	0x00000040	MOTOR_FAILED	Motor abnormal
	0x00000100	ENCODER_FAILED	Encoder exception
	0x00000200	CONTROLLER_FAILED	Controller exception
	0x00001000	MIN_ENDSTOP_PRESSED	Low limit trigger
	0x00002000	MAX_ENDSTOP_PRESSED	High limit triggered
	0x00004000	ESTOP_REQUESTED	Emergency stop
	0x00020000	HOMING_WITHOUT_ENDSTOP	Return to zero but no limit switch

	0x00080000	UNKNOWN_POSITION	No location information
Motor anomaly	0x00000001	PHASE_RESISTANCE_OUT_OF_RANGE	Phase resistance out of normal range
	0x00000002	PHASE_INDUCTANCE_OUT_OF_RANGE	Phase inductance out of normal range
	0x00000010	CONTROL_DEADLINE_MISSED	The FOC frequency is too high
	0x00000080	MODULATION_MAGNITUDE	SVM modulation anomaly
	0x00000400	CURRENT_SENSE_SATURATION	Phase current saturation
	0x00001000	CURRENT_LIMIT_VIOLATION	Excessive motor current
	0x00020000	MOTOR_THERMISTOR_OVER_TEMP	Excessive motor temperature
	0x00040000	FET_THERMISTOR_OVER_TEMP	The drive temperature is too high.
	0x00080000	TIMER_UPDATE_MISSED	FOC processing was not timely
	0x00100000	CURRENT_MEASUREMENT_UNAVAILABLE	Phase current sampling lost
	0x00200000	CONTROLLER_FAILED	Control exception
	0x00400000	I_BUS_OUT_OF_RANGE	The bus current exceeds the limit
	0x00800000	BRAKE_RESISTOR_DISARMED	The bleeder resistor drive is abnormal
	0x01000000	SYSTEM_LEVEL	System_level exception
	0x02000000	BAD_TIMING	Phase current sampling is not timely
	0x04000000	UNKNOWN_PHASE_ESTIMATE	Motor location unknown
	0x08000000	UNKNOWN_PHASE_VEL	Motor speed unknown
	0x10000000	UNKNOWN_TORQUE	Torque unknown

	0x20000000	UNKNOWN_CURRENT_COMMAND	Torque control unknown
	0x40000000	UNKNOWN_CURRENT_MEASUREMENT	The current sampling value is unknown
	0x80000000	UNKNOWN_VBUS_VOLTAGE	Voltage sampling value is unknown
	0x100000000	UNKNOWN_VOLTAGE_COMMAND	Voltage control unknown
	0x200000000	UNKNOWN_GAINS	Current loop gains unknown
	0x400000000	CONTROLLER_INITIALIZING	Controller initialization exception
	0x800000000	UNBALANCED_PHASES	Three phase unbalance
Control anomalies	0x00000001	OVERSPEED	Excessive speed
	0x00000002	INVALID_INPUT_MODE	The control input mode is incorrect
	0x00000004	UNSTABLE_GAIN	PLL gain is unstable
	0x00000020	INVALID_ESTIMATE	Position/speed is unstable
		SPINOUT_DETECTED	Mechanical power and electrical power do not match (encoder is not calibrated correctly, or magnetic steel is unstable)