

GIM6010-8 micromotor instruction manual

Table of contents

Version record.....	2
Precautions.....	3
Legal Notices.....	3
After-sales policy.....	3
1 Motor specifications.....	5
1.1 Drawings and dimensions.....	5
1.2 Electrical Characteristics.....	6
1.3 Mechanical properties.....	7
2 Drive information.....	7
2.1 Appearance and three-dimensional dimensions.....	7
2.2 Interface overview.....	8
2.3 Specification.....	8
2.4 Interface detailed definition.....	9
2.5 Main components and specifications.....	13
3 Commissioning instructions.....	13
3.1 PC commissioning.....	13
3.2 Firmware update download.....	25
4 Communication protocol and examples.....	27
4.1 CAN protocol.....	27
4.2 PYTHON SDK.....	36
4.3 ARDUINO SDK.....	39
4.4 ROS SDK.....	46
5 Frequently Asked Questions and Exception Codes (to be updated).....	47
5.1 Frequently Asked Questions (FAQ).....	47
5.2 Exception code.....	47



version record

version number	date	Revision/clarification
0.1	2023.12.5	Support the first version of odrivetool
0.2	2023.12.15	Added instruction list and 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	Add CAN protocol and Python debugging practical cases
0.7	2024.1.8	Added CAN interface 120R matching resistor switch option
0.8	2024.2.15	Added content related to the second encoder and user zero point settings
0.9	2024.2.23	Added CAN instructions that can modify parameters and call interface functions

Precautions

1. Please use it according to the working parameters in this manual, otherwise it will cause irreversible damage to the product!
2. During the operation of the motor, please take protective measures against overcurrent and overvoltage of the power supply to avoid damaging the driver.
3. Please check that all components are intact before use. If components are missing or damaged, please contact technical support in time.
4. The driver does not have 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. Please do not touch the exposed parts of the drive with your hands to avoid static electricity damage!

Legal Notices

Before using this product, please be sure to read this manual carefully and operate the product according to the content. If the user violates the instructions and uses

Our company does not assume any responsibility for any property damage or personal injury caused by the use of this product. Because this product consists of many parts

It is composed of parts. Do not let children come into contact with this product to avoid accidents. In order to extend the service life of the product, do not expose it to high temperatures or high pressures.

environment in which this product is used. This manual has tried its best to include various function introductions and usage instructions at the time of printing. But due to product features

Continuous improvements, design changes, etc. may still be inconsistent 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 make improvements and changes in this manual

necessary for typographical errors, inaccuracies of current information, or improvements to programs and/or equipment at any time without prior notice. Such changes will be uploaded to a new version of

this manual, please contact technical support to obtain it. All pictures are for functional description only, please refer to the actual product.

After-sales policy

The after-sales service of this product is strictly implemented in accordance with the "Consumer Rights and Interests Protection Law of the People's Republic of China" and the "Product Quality

Law of the People's Republic of China". The service content is as follows:

1. Warranty period and content

- 1) Users who place an order to purchase this product through online channels can enjoy a no-reason return service within seven days from the day of receipt.

When returning goods, users must present valid proof of purchase and return the invoice. Users must ensure that the returned goods maintain their original quality and

The function and appearance are intact, and the trademarks and various logos of the product itself and accessories are complete. If there are any gifts, they must be returned together.

If the product is damaged artificially, disassembled manually, the packaging box is missing, or the spare parts are missing, returns will not be processed. The logistics costs incurred

when returning the goods shall be borne by the user. If the user fails to settle the logistics fees, the actual amount will be deducted from the refund amount. The paid price will be

returned 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 banks and payment institutions.
- 2) If a non-man-made performance failure occurs within 7 days from the day after the user signs for it, the company's after-sales service center will handle the return business for the user after

inspection and confirmation. When returning the product, the user must present a valid purchase certificate and return the invoice. If any

Gifts must be returned together.
- 3) If non-human-damaged performance failure occurs within 7 days to 15 days from the day after the user signs for it, after inspection and confirmation by the company's after-sales service center,

the user will be exchanged and the entire set of goods will be replaced. After the exchange, the three-guarantee period of the product itself will be recalculated.

- 4) Within 15 days to 365 days from the day after the user signs for it, after inspection and confirmation by the company's after-sales service center, it is a quality failure of the product itself, and repair services can be provided free of charge. Replaced faulty products shall become the property of the Company. Products that are not defective will be returned in their original condition. This product is shipped after undergoing various strict tests. If there is any quality failure that is not caused by the product itself, we will take responsibility for it.
- The right to refuse the user's return or exchange request.

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

2. Non-warranty regulations

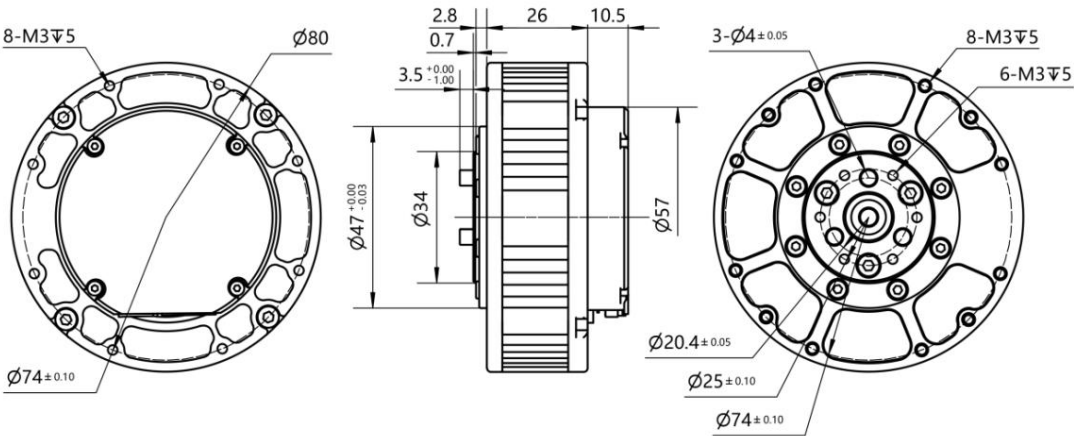
The following situations are not covered by the warranty:

- 1) Exceeds the warranty period limited by the warranty terms.
- 2) Product damage caused by incorrect use without following the instructions.
- 3) Damage caused by improper operation, maintenance, installation, modification, testing and other improper use.
- 4) Routine mechanical loss and wear 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 floods, fires, lightning strikes, earthquakes, etc.) or force majeure.
- 7) Damage caused by use exceeding peak torque.
- 8) It is not an original and genuine product of our company or it is impossible to provide legal proof of purchase.
- 9) Other faults or damage caused by problems other than product design, technology, manufacturing, quality, etc.
- 10) Damage caused by unauthorized disassembly of this product.

If the above situation occurs, users need to pay the fees themselves.

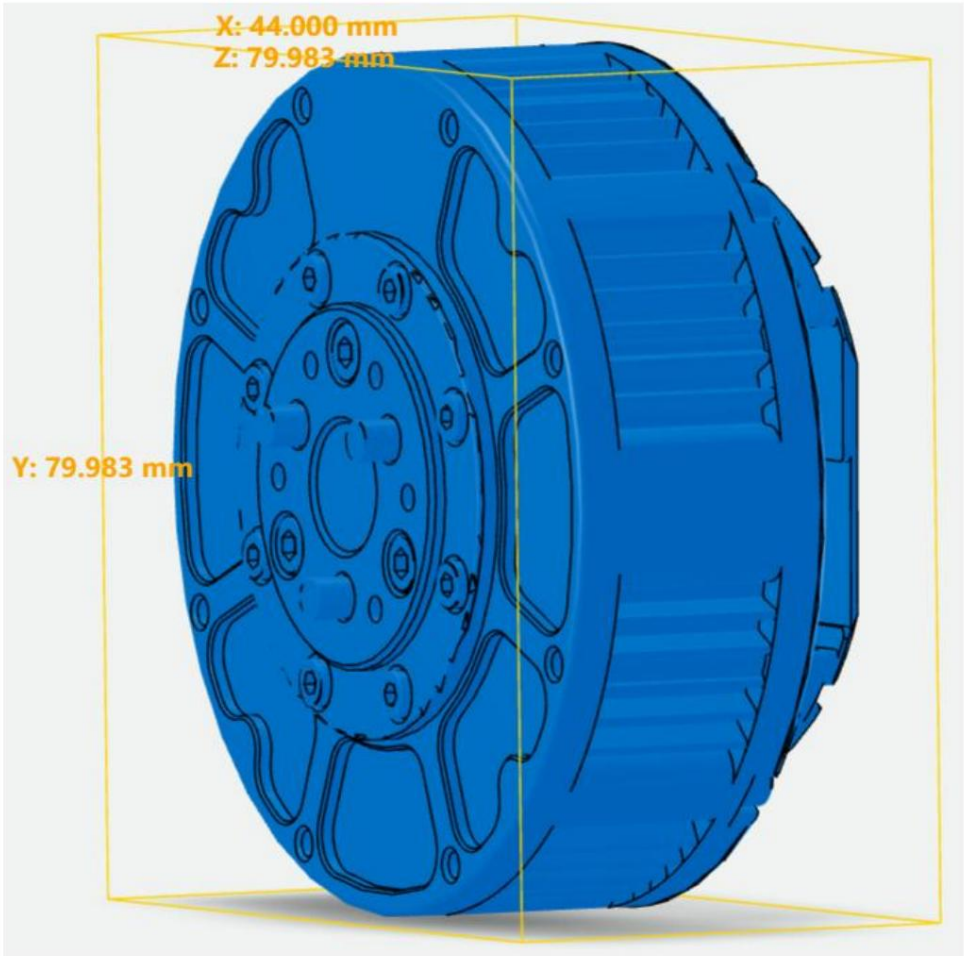
1 Motor specifications

1.1 Drawings and dimensions



未注公差分段						
<=6	>6<=30	>30<=120	>120<=400	>400<=2000	比例	1 : 1
± 0.1	± 0.2	± 0.3	± 0.5	± 1.0	重量kg	无

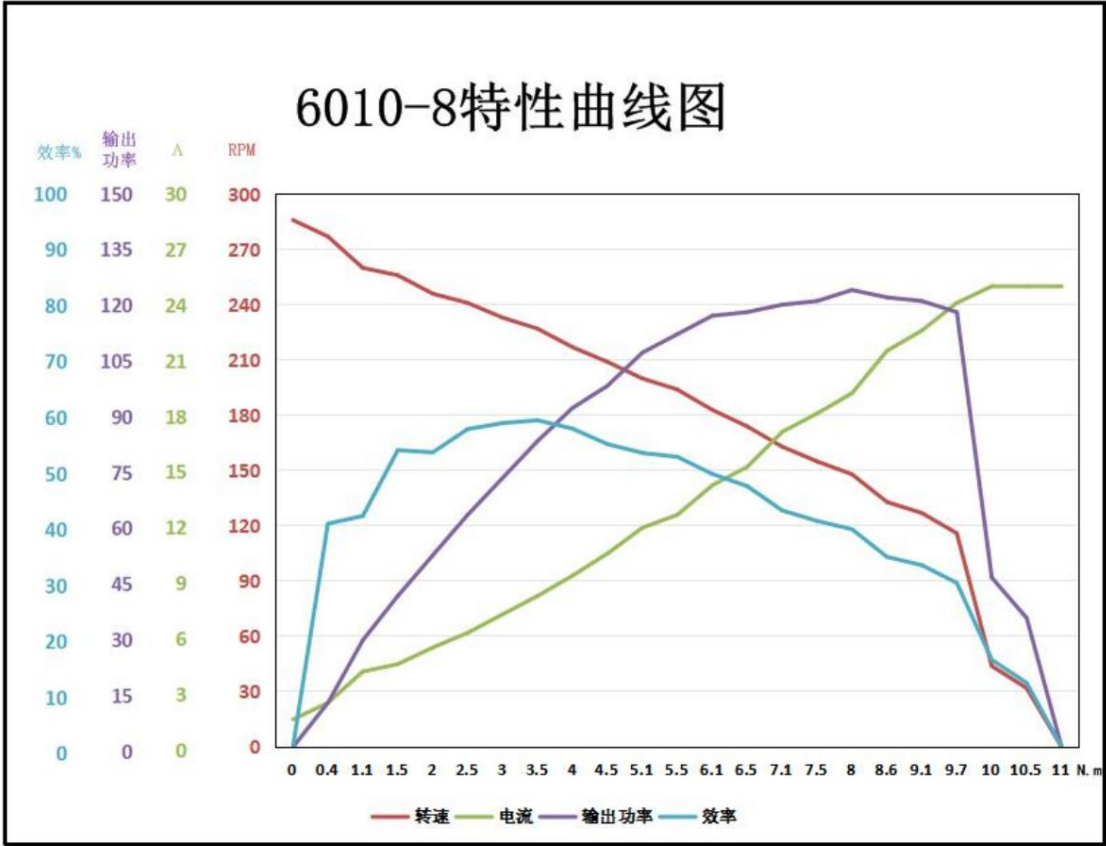
6010-8安装尺寸图



1.2 Electrical characteristics

Rated speed,	120rpm±10%
maximum speed,	420rpm±10%
rated torque,	5 Nm
locked-rotor	11N.m
torque, rated	10.5A
current, locked-	25A
rotor current, no-	0.4A
load current, insulation resistance/	DC 500VAC, 100M Ohms
stator winding high voltage resistance/	600 VAC, 1s, 2mA
stator and chassis	0.054~0.057Vrms/rpm
motor back	0.48ÿ±10%
electromotive	368ÿH±10%
force, phase	12.3rpm/v
resistance, phase inductance, speed constant, torque constant	0.47NM/A

The characteristic curve is as follows:

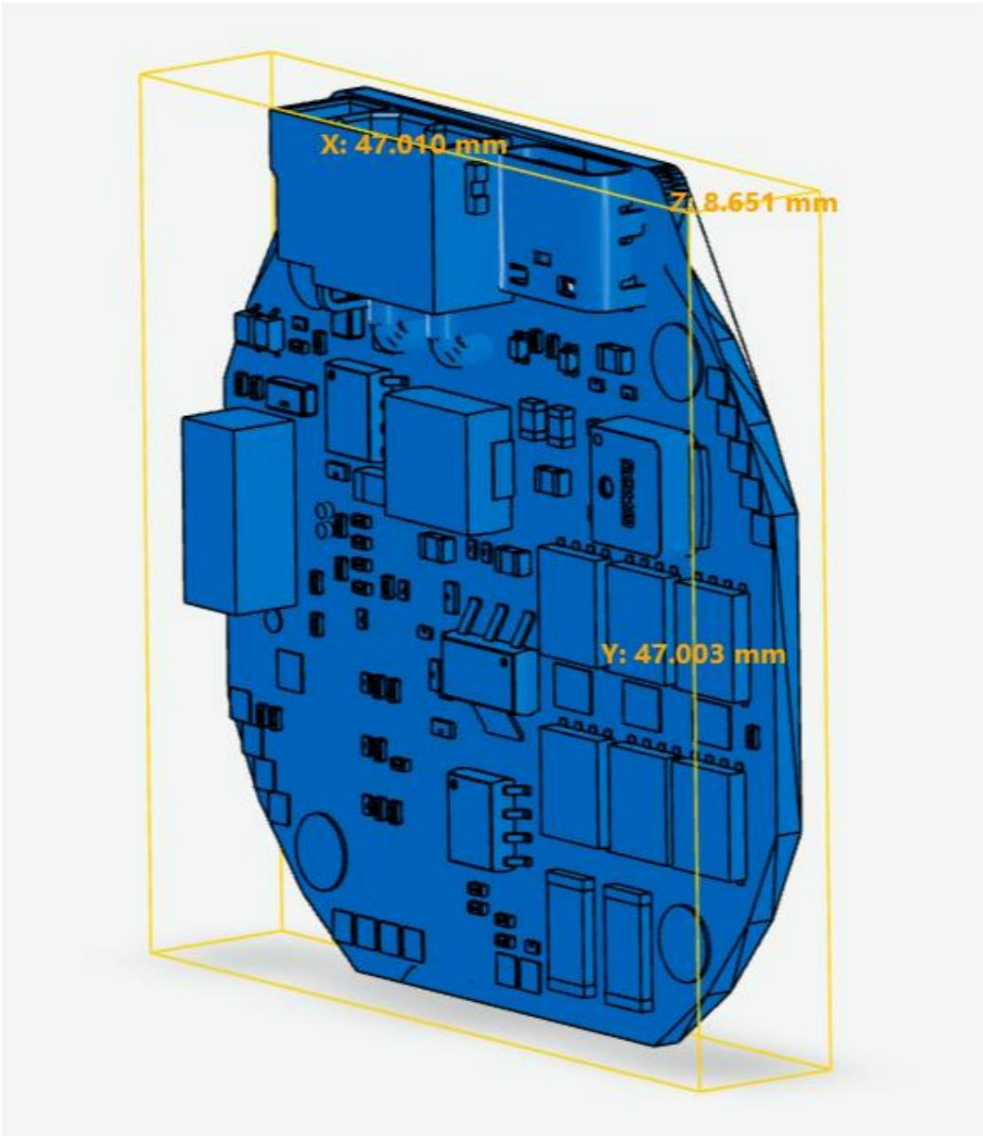


1.3 Mechanical properties

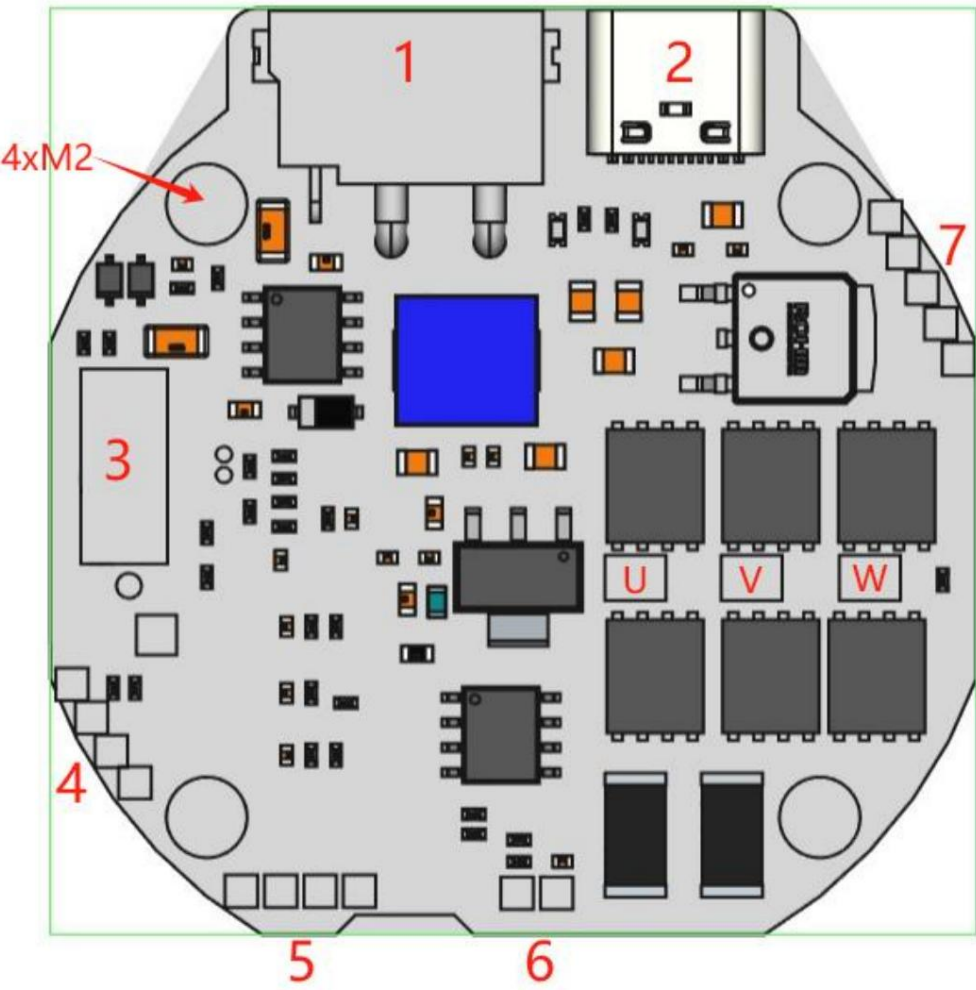
Weight	388g±3
pole	14 pairs
	Phase 3
logarithmic	FIRE
phase drive mode reduction ratio	8:1

2 drive information

2.1 Appearance and three-dimensional dimensions



2.2 Interface overview



Interface serial number	definition
1	15~60V power supply and CAN communication integrated terminal
2	Type-C debugging interface and host computer communication interface
3	Interface expansion slot (can be expanded to RS485, EtherCAT, model aircraft, pulse direction, oil Door control and other interfaces/protocols)
4	SWD debugging and download interface
5	Second encoder interface (supports I2C and UART)
6	Motor temperature interface (NTC)
7	Brake/braking resistor interface, 12V power supply, minimum/maximum limit switch interface
U/V/W	Three-phase winding welding holes
4xM2	Mounting holes

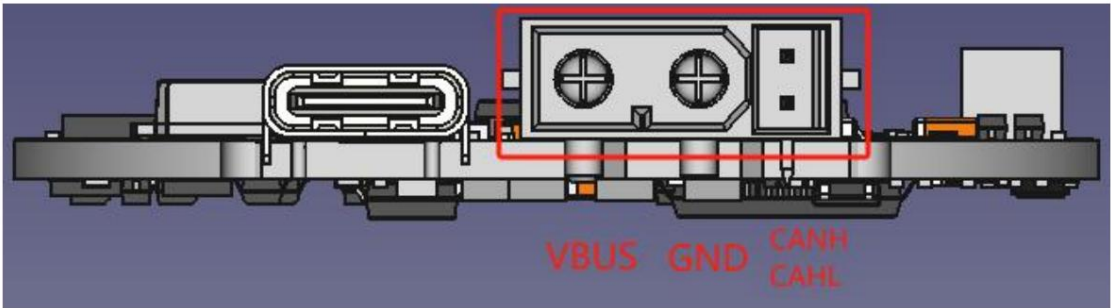
2.3 Specifications

Rated voltage	15~48V DC
---------------	-----------

Min/Max Voltage Rated	12/72V DC
Current Max	6A
Line Current Max	30A
Phase Current	120A
Standby Power Consumption	<10mA
CAN bus baud rate	1Mbps
Type-C rate encoder	10Mbps
resolution Working	16bit (single lap absolute value)
environment temperature	-20~70
alarm Motor temperature	90(adjustable)
alarm Driver board temperature	90(adjustable)

2.4 Detailed definition of interface

2.4.1 Power supply and CAN communication terminals



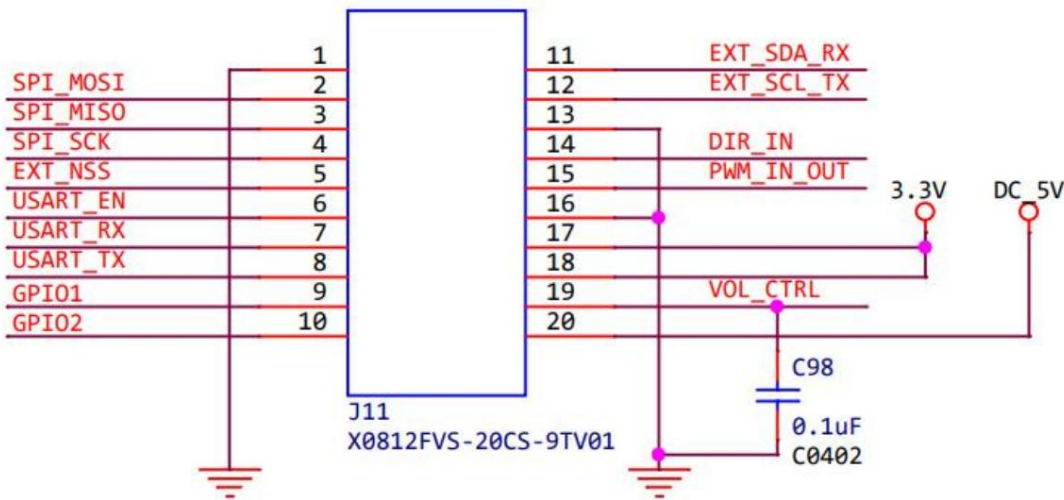
The onboard terminal model is XT30PB(2+2)-M, the line terminal model is XT30(2+2)-F, and the brand manufacturer is AMASS.

2.4.2 Type-C debugging interface

Type-C uses standard data cable specifications, and is compatible with commonly used PC or mobile phone Type-C data cables.

2.4.3 Interface expansion slot

This slot adopts the following design method to provide rich inter-board expansion interfaces, and any expansion board can be developed by a third party:



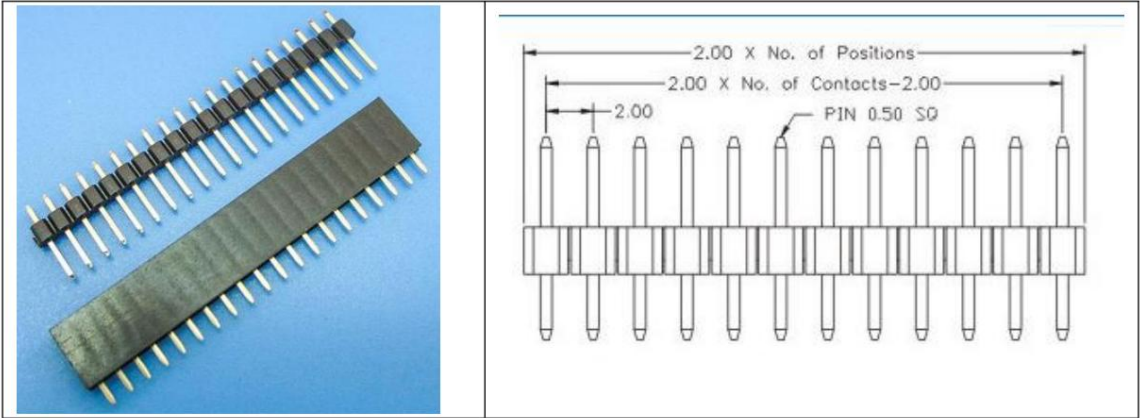
The third party can interact with the driver through SPI, USART, I2C, PWM, ADC, GPIO, etc. to achieve various extensions.

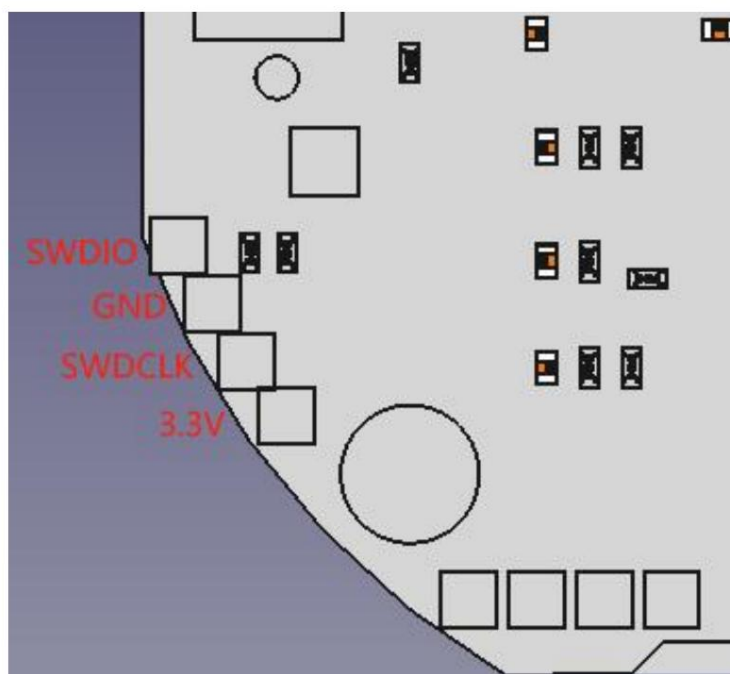
The onboard slot model is X0812FVS-20CS-9TV01 (female socket), and the expansion board slot model is X0812WVS-20AS-9TV01 (public seat), the brand manufacturer is Xingkun.

Female base X0812FVS-20CS-9TV01	Public seat X0812WVS-20AS-9TV01

2.4.4 SWD debugging interface

For pin holes spaced 2mm apart, users can weld 2mm straight-in single row pins, as shown below:

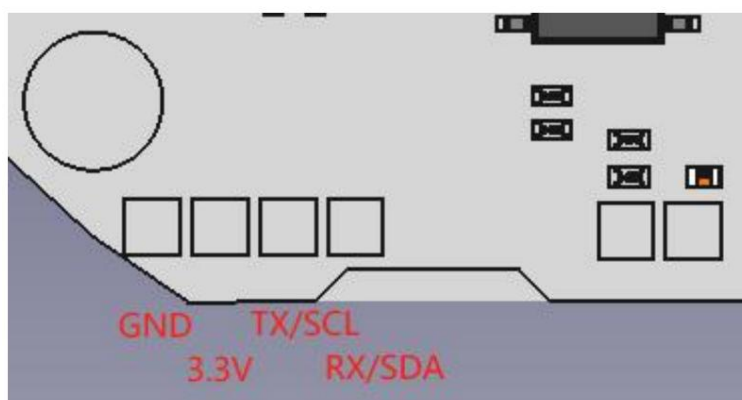




2.4.5 Second encoder interface

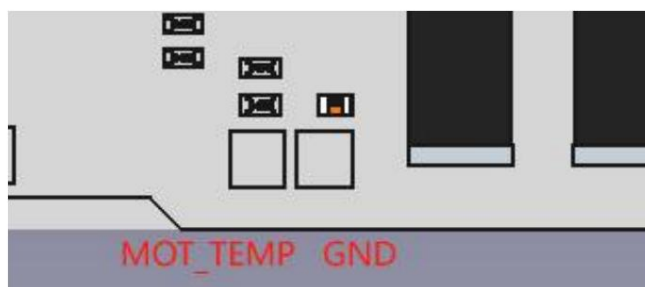
For pin holes spaced 2mm apart, users can weld 2mm straight single-row pins, please refer to 2.4.4.

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



2.4.6 Motor temperature interface

The motor has a built-in 10K NTC resistor, and the two leads are welded to MOT_TEMP and GND, with no line sequence.

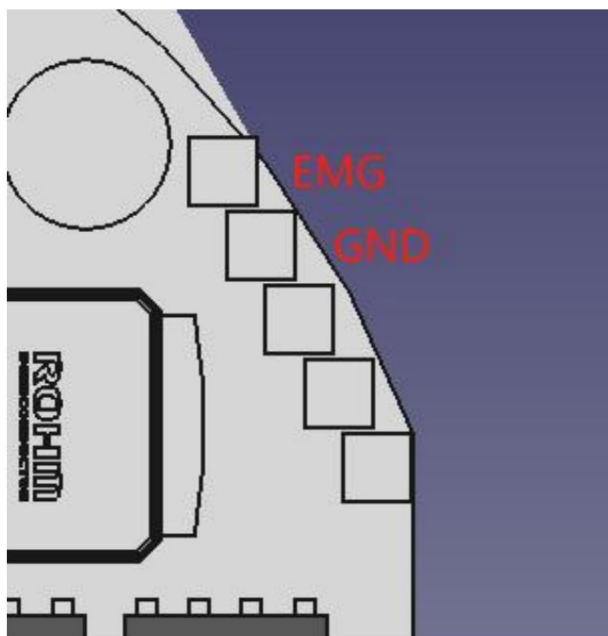


2.4.7 Brake/braking resistor interface

The two upper welding holes in the 5-pin interface shown in the figure are the brake/braking resistor interfaces. The pin holes are spaced 2mm apart. The user can weld 2mm straight single-row pins. Please refer to 2.4.4.

When it is a brake interface, the driver continues to output a current to this interface when powered on, so that the brake is open and the motor can operate normally. If the driver is powered off, the current will stop, the brake will be locked, and the motor will be locked in the power-off position.

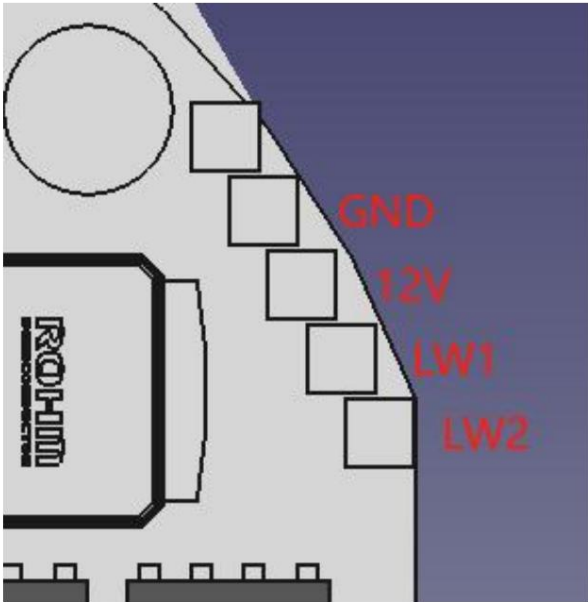
When it is a braking resistor interface, an external braking resistor (or called a bleed resistor) can be connected. When the back electromotive force is higher than the threshold voltage, when the voltage is high, the current is discharged through this braking resistor to prevent emergency braking from failure or back electromotive force damaging the driver.



2.4.8 Limit switch interface

The driver provides two limit switch interfaces and provides 12V power supply for external limit switches. The pin holes are spaced 2mm apart. Households can weld 2mm straight single row pins, please refer to 2.4.4.

Among them, LW1 is the minimum position limit switch, and LW2 is the maximum position limit switch. Can be connected to an external two-wire switch or three-wire switch NPN switch.



2.5 Main components and specifications

serial number	device	Model/Specification	quantity
1	MCU	N32G455REL7	1
2	Drive chip	FD6288Q	1
3	Magnetic encoder chip	MA600, 16bit absolute value	1
4	MOSFET	JMSH1004NGy100V/120A	6

3 Commissioning instructions

3.1 PC commissioning

Use Motor Genie (launched in January 2024) for visual debugging.

The driver is compatible with [odrive](https://github.com/odriverobotics/odrive.git) (<https://github.com/odriverobotics/odrive.git>), so it can also be used [odrivetool](#)

Carry out debugging.

3.1.1 Host computer installation

3.1.1.1 Windows

1. Install python

Enter the python official website <https://www.python.org> Download the latest python installer and follow the prompts to install. please

Do not download python versions from third-party websites or Microsoft Store.

2. Install visual c++ generation tools

Install the visual c++ build tools <https://visualstudio.microsoft.com/visual-cpp-build-tools/> ,Install

During the process, check "Desktop development using C++", as shown in the figure below.



3. Install odrivetool

Run Windows PowerShell as an administrator, run `pip install odrive` and press Enter to install. if halfway

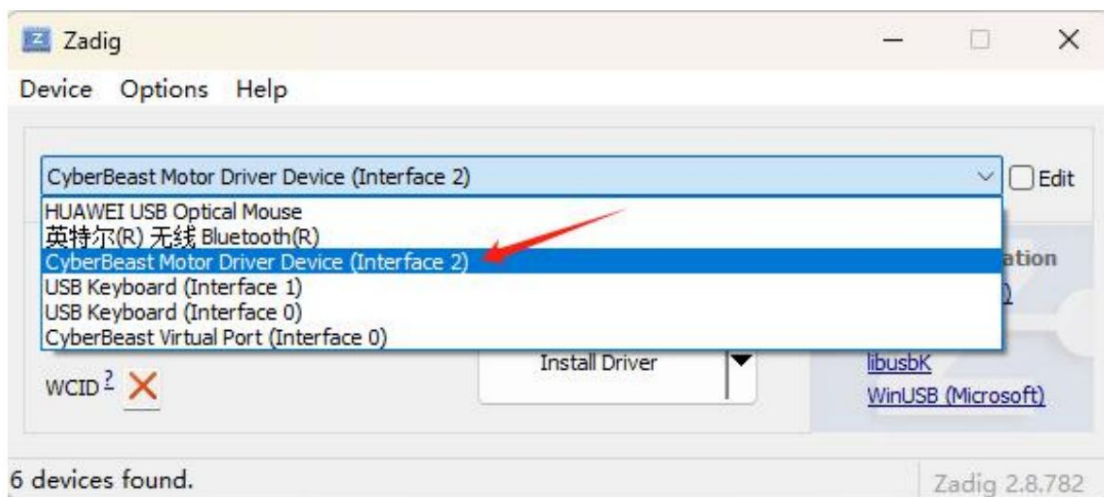
An error occurred. Please try again. If the error persists, please restart your computer and try again.

4. Install USB driver

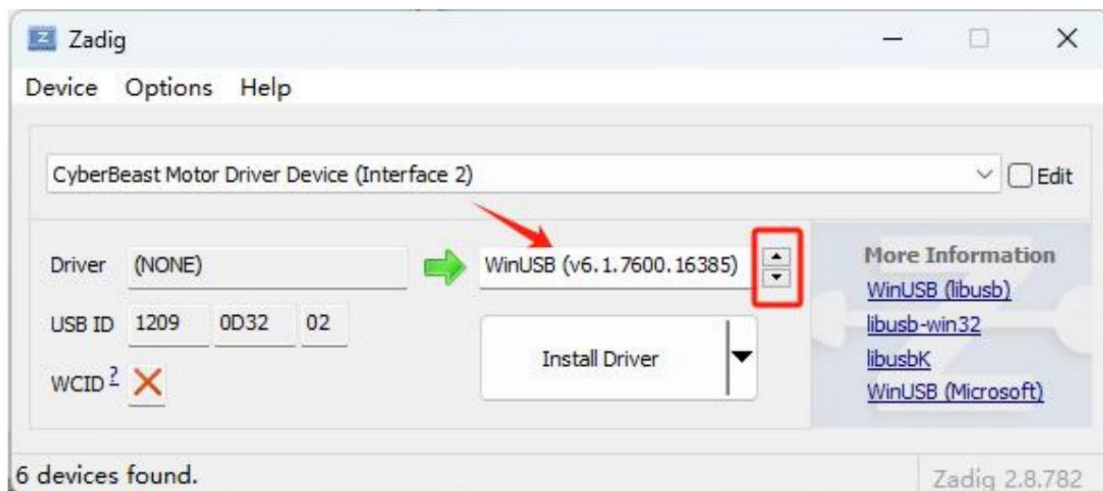
Enter the website <https://zadig.akeo.ie> And download the USB driver tool Zadig, and connect the drive with a Type-C data cable.

Computer, now the drive power light is on, open Zadig, select "CyberBeast Motor Driver Device" through the drop-down box

Interface 2



Select different USB drivers by clicking the up and down keys. Please select the "WinUSB" driver version for this interface and click "Install Driver" to install the driver for this interface:



3.1.1.2 WSL Windows Subsystem for Linux

1. Install python/usb/odrivetool

If the user has installed WSL2, he can enter the WSL2 command line and follow the steps below to install it (assuming the user WSL2

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

Ubuntu is installed):

The first line of instructions installs python, the second line of instructions installs the USB driver, and the third line of instructions installs odrivetool on the host computer.

2. Connect the drive to WSL

Plug into Windows with a type-C data cable. By default, Windows will load the driver for this USB port, but WSL will not. You need to load this USB port into WSL, please refer to Microsoft documentation

<https://learn.microsoft.com/zh-cn/windows/wsl/connect-usb> operation.

3.1.1.3 Ubuntu

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

3.1.2 USB and CAN compatibility

In earlier versions of this product (hardware version less than or equal to 3.7, which can be obtained through the instructions odrv0.hw_version_major and odrv0.hw_version.minor in the next section 3.1.3), USB is not compatible with CAN. You can obtain it through the following methods Switch between the two communication methods (if the hardware version is greater than 3.7, you can ignore this section):

- Switch to CAN during USB communication

When CAN is disabled, users can use the Type-C interface to communicate (see the next section 3.1.3). At this time, the user can use the following

```
odrv0.config.enable_can_a = True
odrv0.axis0.requested_state = AXIS_STATE_IDLE
odrv0.save_configuration()
```


Instructions switch to CAN:

ÿ Switch to USB during CAN communication

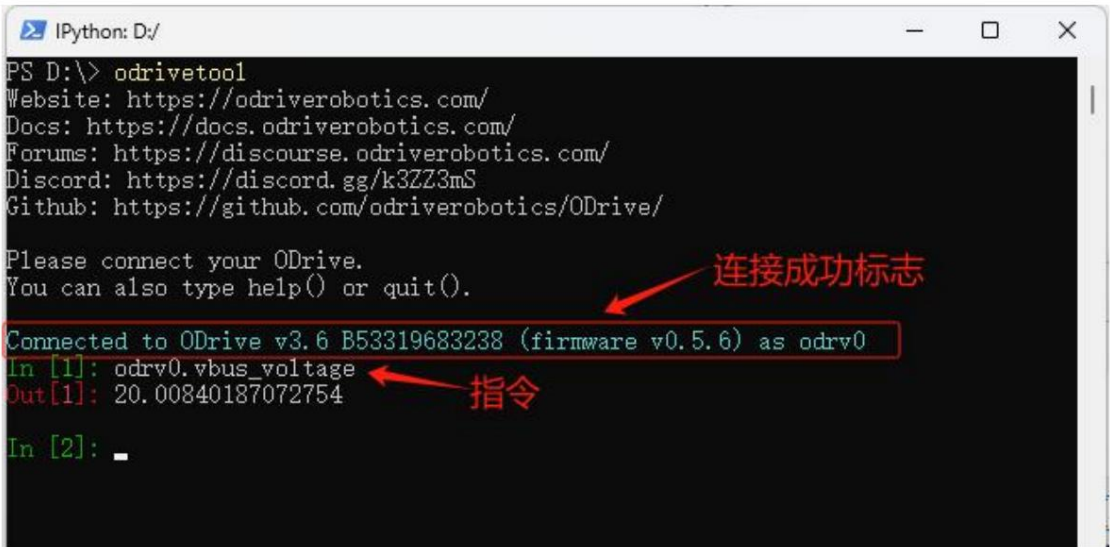
When CAN is enabled, the user first sends the message Set_Axis_State (the parameter is 1, indicating the idle (IDLE) state)

Switch the motor to the idle state and then switch to USB by sending the Disable_Can message (see 4.1.2).

Please note that whether switching from USB to CAN or CAN to USB, the motor must be idle (IDLE) first.
status, otherwise the switch will fail.

3.1.3 PC commissioning

Please connect the main power supply of the motor (the CAN interface does not need to be connected) and the Type-C interface, enter Windows PowerShell on the computer, Run odrivetool. If the connection is successful, it will be displayed in green font as follows:



If the above connection success sign does not appear, please re-plug the USB and confirm that the power connection is normal, or refer to 3.1.2 to check
See how to switch USB and CAN communications.

3.1.3.1 Instruction list

After the connection is successful, the user can control the motor through instructions and obtain the parameters of the motor operation. The following table is the commonly used instructions
And debugging process and instructions:

type	directive	illustrate
Base instruction	dump_errors(odrv0)	Print all error messages
	odrv0.clear_errors()	Clear all error messages
	odrv0.save_configuration()	After the parameters have been modified, or the motor automatically recognizes the parameters or calibrate After approval, be sure to execute this command to save the modifications, otherwise All modifications will be lost after a power outage.
	odrv0.reboot()	Reboot the drive
	odrv0.vbus_voltage	Get the supply voltage (V)
	odrv0.ibus	Get the power supply current (A)
	odrv0.hw_version_major	Hardware major version number, current major version of GIM6010-8

		The
	odrv0.hw_version_minor	hardware minor version number is 3. The current minor version number of GIM6010-8
	odrv0.hw_version_variant	is 7. Different model numbers under the same hardware configuration. The corresponding model number
	odrv0.can.config.r120_gpio_num	for GIM6010-8 is 1. Controls the 120R matching resistor switch of the CAN interface. The GPIO
	odrv0.can.config.enable_r120	number controls the 120R matching resistor switch of the CAN interface.
	odrv0.can.config.baud_rate	CAN baud rate sets low
Parameter configuration instructions	odrv0.config.dc_bus_undervoltage_trip_level	voltage alarm threshold (V)
	odrv0.config.dc_bus_overvoltage_trip_level	Overvoltage alarm threshold (V)
	odrv0.config.dc_max_positive_current	Line current maximum value (positive value) (A)
	odrv0.config.dc_max_negative_current	Line current reverse charging maximum value (negative value) (A)
	odrv0.axis0.motor.config.resistance_calibration_max_voltage	The maximum voltage value during motor parameter identification. Generally, this value is slightly less than half of the power supply voltage. For example, if the power
	odrv0.axis0.motor.config.calibration_current	supply is 24V, it can be set to 10. The maximum current value during motor parameter identification. This value can generally be set to 2~5A. It should
	odrv0.axis0.motor.config.torque_constant	not be too large or too high. Small. Motor torque constant (Nm/A)
	odrv0.axis0.encoder.config.index_offset	The zero point offset set by the user. This value is the offset value of the user zero point relative to the encoder zero point. After setting this offset value and saving the setting, all position control target values input by the user will be based on this user zero point. Identify the parameters of the motor, including
Calibration instructions	odrv0.axis0.requested_state=4	identifying phase resistance, phase inductance, and calibrating the three-phase current balance. This process will take 3 to 6 seconds, and the motor will make a sharp sound. After the sound stops, or when there is no sound after 6 seconds, run dump_errors(odrv0) to check the errors and confirm that there are no errors before performing other operations. Calibrate the encoder. Before performing this
	odrv0.axis0.requested_state=7	operation, please confirm that there is no load on the motor output shaft and secure the motor with hands or other devices. After this operation is performed, the motor will rotate forward and reverse for a certain period of time to identify and calibrate the encoder. After the motor stops, run dump_errors(odrv0) to check the errors and confirm that there are no errors before proceeding with other subsequent steps. If the write-in pre-calibration is
	odrv0.axis0.encoder.config.pre_calibrate_d=1	successful, it means that calibration does not need to be performed every time the power is turned on. This parameter can only be written after the above calibration is successful,
	odrv0.axis0.controller.config.load_encoder_axis=0	otherwise the writing will fail. Make sure that the current operating motor is the 0th motor. This operation is only necessary in BETA and has no effect in mass production versions.

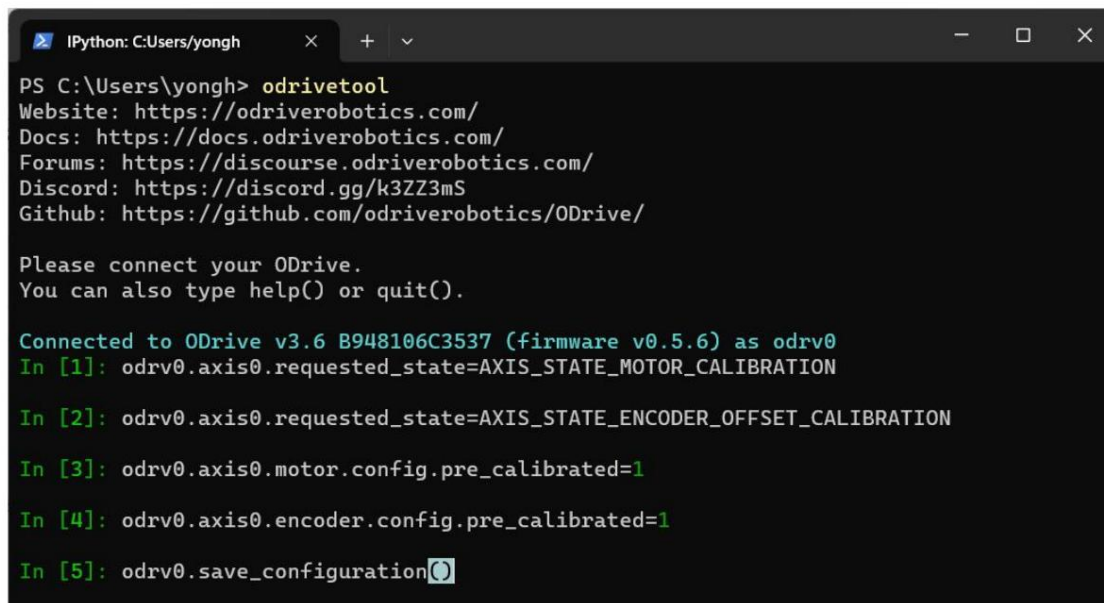
Control instruction	state odrv0.axis0.requested_state=1	Stop the motor and enter the idle
	odrv0.axis0.requested_state=8	Start the motor and enter the closed-loop state
	odrv0.axis0.motor.config.current_lim	The maximum line current of the motor (A), exceeding this value will cause An overflow alarm is reported . Note that this value cannot be greater than 100.
	odrv0.axis0.controller.config.vel_limit	is the maximum speed of the motor (turns/s). If the motor rotor speed exceeds this
	odrv0.axis0.controller.config.enable_vel_limit	value, an overspeed alarm will be reported. Speed limit switch, when it is True, the
	odrv0.axis0.controller.config.control_mode	above
	of	vel_limit is effective, when it is False, it is ineffective.
	odrv0.axis0.controller.config.input_mode	control mode. 0: Voltage control 1: Torque control 2: Speed control 3: Position control
	It is	input mode. Indicates how the control value input by the user controls the
	motor operation: 0: Idle 1: Direct control 2: Speed ramp 3: Position filter 5: Trapezoidal	
	odrv0.axis0.controller.config.vel_gain	Speed loop P value
	controlled by PID Speed loop I value odrv0.axis0.controller.config.vel_integrator_gain	curve 6: Torque ramp
	controlled by PID odrv0.axis0.controller.config.pos_gain	P value controlled by position loop PID
	odrv0.axis0.controller.input_vel	odrv0.axis0.controller.input_torque Target of torque control, Or
	control/position control (Nm) Target for	torque feedforward for speed
	speed control, or speed feedforward for position control (turns/s) Target for position control (turns)	odrv0.axis0.controller.input_pos odrv0.axis0.encoder.set_linear_count() sets the absolute position odrv0.axis0.encoder.config.cpr contains
	odrv0.axis0.trap_traj.config	three parameters: \ddot{y} accel_limit: maximum acceleration (rev/s ²) \ddot{y} decel_limit: maximum deceleration (rev/s ²) \ddot{y} vel_limit: maximum speed (rev/s) These three parameters work when odrv0.axis0.controller.config.input_mode is a trapezoidal curve to adjust the acceleration and deceleration effect
	odrv0.axis0.controller.config.input_filter_bandwidth	of position control. Position filter bandwidth, this parameter is in odrv0.axis0.controller.config.input_mode

		It works when position filtering is used to adjust the acceleration and deceleration effect of position control.
--	--	--

3.1.3.2 Practical combat: power-on calibration

When the user uses the micromotor for the first time, the motor and encoder need to be calibrated. Before calibrating, please fix the motor.

Or hold it tightly by hand, the output shaft is unloaded, and the calibration process is as follows:



```

IPython: C:\Users\yongh
PS C:\Users\yongh> odrivetool
Website: https://odriverobotics.com/
Docs: https://docs.odriverobotics.com/
Forums: https://discourse.odriverobotics.com/
Discord: https://discord.gg/k3ZZ3mS
Github: https://github.com/odriverobotics/ODrive/

Please connect your ODrive.
You can also type help() or quit().

Connected to ODrive v3.6 B948106C3537 (firmware v0.5.6) as odrv0
In [1]: odrv0.axis0.requested_state=AXIS_STATE_MOTOR_CALIBRATION

In [2]: odrv0.axis0.requested_state=AXIS_STATE_ENCODER_OFFSET_CALIBRATION

In [3]: odrv0.axis0.motor.config.pre_calibrated=1

In [4]: odrv0.axis0.encoder.config.pre_calibrated=1

In [5]: odrv0.save_configuration
  
```

```

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 odrv0.save_configuration()
  
```

The explanation is as follows:

Step 1: Self-identification of motor parameters

When measuring the phase resistance and phase inductance of the motor, you will hear a sharp "beep" sound. The measurement results of phase resistance and phase inductance can be obtained by

```

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

View the following instructions:

Step 2: Check the error code

Check the system error codes after the first step. If any red error codes appear, you need to restart the motor and try again.

Or report after sales.

ÿ Step 3: Encoder calibration

Calibrate the encoder, including the calibration of the encoder's installation angle and the mechanical angle of the motor, as well as the calibration of the encoder itself.

During this calibration process, the motor will slowly rotate forward by an angle and then reverse by an angle. If it only rotates forward and then stops, it means

If there is an error, please go to step 4 to check the error code.

ÿ Step 4: Check the error code

After the encoder calibration in the third step, check the system error code. The usual error is

ERROR_CPR_POLEPAIRS_MISMATCH, indicating that the CPR setting of the encoder is wrong, or the number of pole pairs of the motor is set wrong,

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

Please view/set via the following commands:

ÿ Step 5: Write the motor calibration success flag

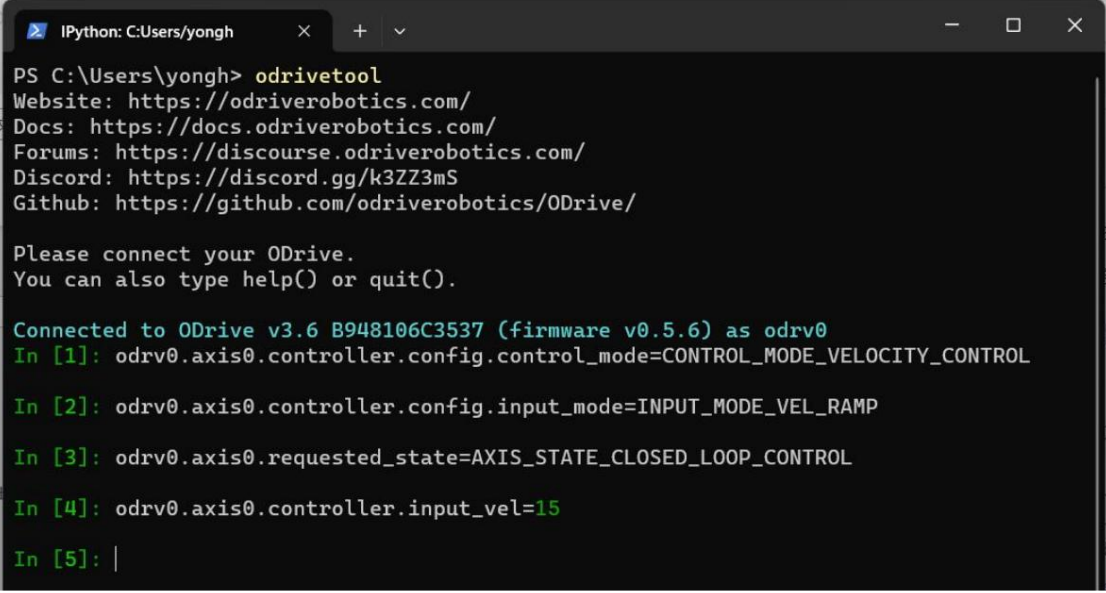
ÿ Step 6: Write the encoder calibration success flag

ÿ Step 7: Store calibration results and restart

3.1.3.3 Practical combat: speed control

The following instructions are examples of speed control:

```
odrv0.axis0.controller.config.control_mode=CONTROL_MODE_VELOCITY_CONTROL
odrv0.axis0.controller.config.input_mode=INPUT_MODE_VEL_RAMP
odrv0.axis0.requested_state=AXIS_STATE_CLOSED_LOOP_CONTROL
odrv0.axis0.controller.input_vel=15
```



```
IPython: C:\Users\yongh  x  +  v

PS C:\Users\yongh> odrivetool
Website: https://odriverobotics.com/
Docs: https://docs.odriverobotics.com/
Forums: https://discourse.odriverobotics.com/
Discord: https://discord.gg/k3ZZ3mS
Github: https://github.com/odriverobotics/ODrive/

Please connect your ODrive.
You can also type help() or quit().

Connected to ODrive v3.6 B948106C3537 (firmware v0.5.6) as odrv0
In [1]: odrv0.axis0.controller.config.control_mode=CONTROL_MODE_VELOCITY_CONTROL

In [2]: odrv0.axis0.controller.config.input_mode=INPUT_MODE_VEL_RAMP

In [3]: odrv0.axis0.requested_state=AXIS_STATE_CLOSED_LOOP_CONTROL

In [4]: odrv0.axis0.controller.input_vel=15

In [5]: |
```

3.1.3.4 Practical combat: position control

The following command controls the motor to rotate to the rotor position of 10 revolutions:

```

IPython: C:\Users\yongh
PS C:\Users\yongh> odrivetool
Website: https://odriverobotics.com/
Docs: https://docs.odriverobotics.com/
Forums: https://discourse.odriverobotics.com/
Discord: https://discord.gg/k3ZZ3mS
Github: https://github.com/odriverobotics/ODrive/

Please connect your ODrive.
You can also type help() or quit().

Connected to ODrive v3.6 B948106C3537 (firmware v0.5.6) as odrv0
In [1]: odrv0.axis0.controller.config.control_mode=CONTROL_MODE_POSITION_CONTROL

In [2]: odrv0.axis0.controller.config.input_mode=INPUT_MODE_POS_FILTER

In [3]: odrv0.axis0.requested_state=AXIS_STATE_CLOSED_LOOP_CONTROL

In [4]: odrv0.axis0.controller.input_pos=10

In [5]: |
  
```

```

odrv0.axis0.controller.config.control_mode=CONTROL_MODE_POSITION_CONTROL odrv0.axis0.controller.config.input_mode=INPUT_MODE_POS_FILTER

odrv0.axis0.requested_state=AXIS_STATE_CLOSED_LOOP_CONTROL odrv0.axis0.controller.input_pos=10
  
```

3.1.4 Graphical debugging

When debugging the motor, if you need to monitor certain operating parameters in real time, you can use python's powerful calculation library and Graphics library, and the high-speed throughput capability of the Type-C interface output motor parameters in real time.

1. Environment preparation

```
pip install numpy matplotlib
```

Install the calculation library and graphics library:

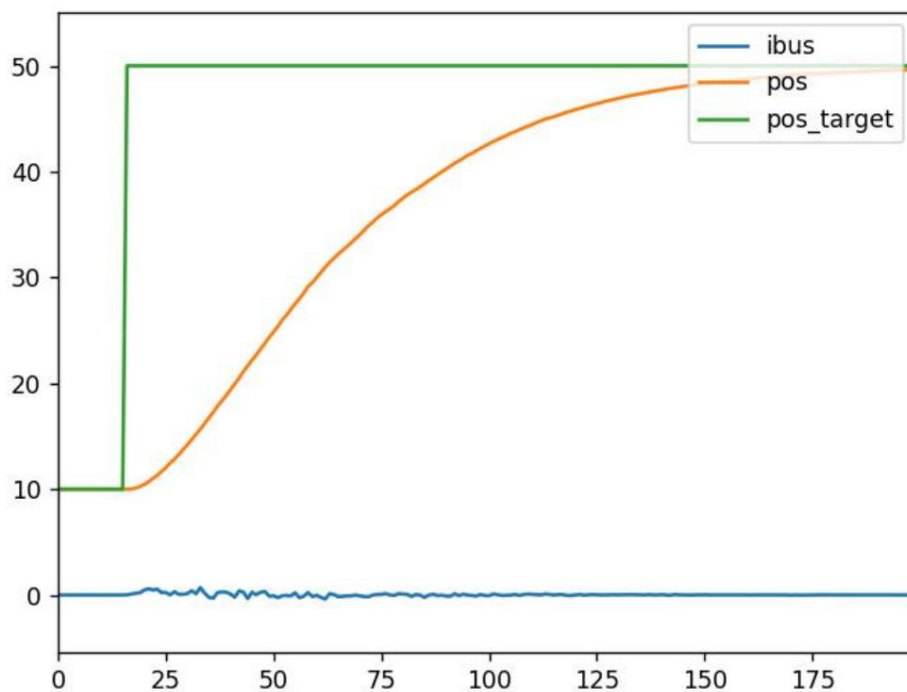
2. Graphical parameter output

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

In the odrivetool command line interface, call up the graphics library and read any motor operating indicators, such as:

This command will bring up a graphical interface and output the following three indicators in real time: line current, position, and target position. Next,

If you perform position control on the motor, you will see the real-time position control curve of the motor:



3.1.5 CAN matching resistor switch

The driver has a 120 ohm impedance matching resistor onboard, which the user can turn on or off as needed. Example of instructions.

as follows:

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

3.1.6 Second encoder

The function of the second encoder is to determine the initial position of the rotor based on the position of the output shaft when power is turned on, thereby ensuring that the motor is shut down.

When electrically restarting, the single-turn position of the output shaft can be correctly identified. You can use the following command to determine whether there is a second encoder and the second

Whether the encoder is working properly: `odrv0.axis0.encoder.poll_sec_enc()`, returns True/False.

3.1.7 User zero point configuration

By default, the position read by the user from the motor and the input value during position control are based on the zero point of the absolute encoder on the driver. However, in user scenarios, the zero point of the encoder is not the user zero point most of the time.

So the user needs to set this zero offset manually.

Generally speaking, users can locate this zero point through two means, one is through the limit switch, and the other is manually set the zero offset, that is, the offset value of the user zero point relative to the encoder zero point:

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

3.1.8 PID adjustment

```
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
```

The following process can provide a reference for users to adjust PID parameters:

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

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

3. Adjust vel_gain method:

- 1) Use speed control mode to rotate the motor. If the rotation is not smooth, there is jitter or vibration, reduce vel_gain until to rotate smoothly
- 2) Next, increase vel_gain by about 30% each time until obvious jitter appears.
- 3) At this time, reduce vel_gain by about 50% to stabilize

4. Adjust pos_gain method:

- 1) Use position mode to try to rotate the motor. If the rotation is not smooth, there is pulling or vibration, reduce pos_gain until to rotate smoothly
- 2) Then, increase pos_gain by about 30% each time until there is obvious overshoot in the position control (i.e. each time the position The control motor will exceed the target position and then oscillate back to the target position)
- 3) Then, gradually reduce pos_gain until the overshoot phenomenon disappears

5. After adjusting the above 4 steps, vel_integrator_gain can be set to $0.5 \times \text{bandwidth} \times \text{vel_gain}$, where bandwidth is the system control bandwidth. What is bandwidth control? For example, from the user setting the target position to the motor actually The time to reach the target position is 10ms, then the control bandwidth is 100Hz, then $\text{vel_integrator_gain} = 0.5 \times 100 \times \text{vel_gain}$

During the above parameter adjustment process, it is recommended to use the graphical method in 3.1.4 to view the parameter adjustment effect in real time to avoid errors in naked eye perception.

Difference.

3.1.9 Motor parameter backup and recovery

Using odrivetool, you can back up the parameters of a fully debugged motor:

```
odrivetool backup-config d:\test.json
```


Among them, "d:\test.json" is the saving path and file name that the user can freely modify.

The command to restore parameters is:

```
odrivetool restore-config d:\test.json
```

3.2 Firmware update download

Firmware can be burned through the SWD interface (2.4.4) or Type-C interface (2.4.2). The following three methods are provided:

3.2.1 pyocd

pyocd is the python version of openOCD, which can support common debugging tools such as STLink, JLink, and DAP for erasing,

Operations such as programming and resetting. Please note that **the drive must be connected using the SWD connector. For** the line sequence of SWD, please refer to 2.4.4.

There is a 3.3V power supply in the SWD interface. Please do not connect the wrong wiring sequence to avoid damaging the driver!

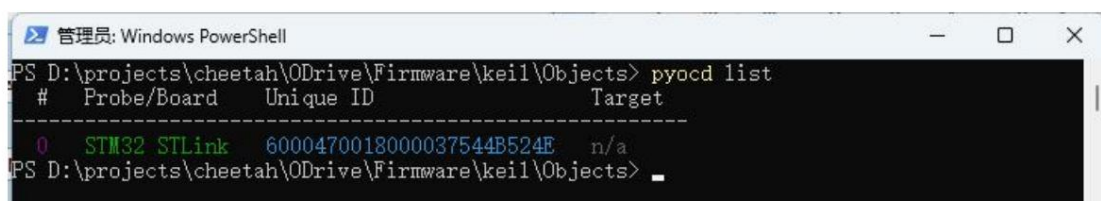
```
pip install pyocd
```

1. Installation

2. Programming

```
pyocd list
```

First, list the connected debugging tools:

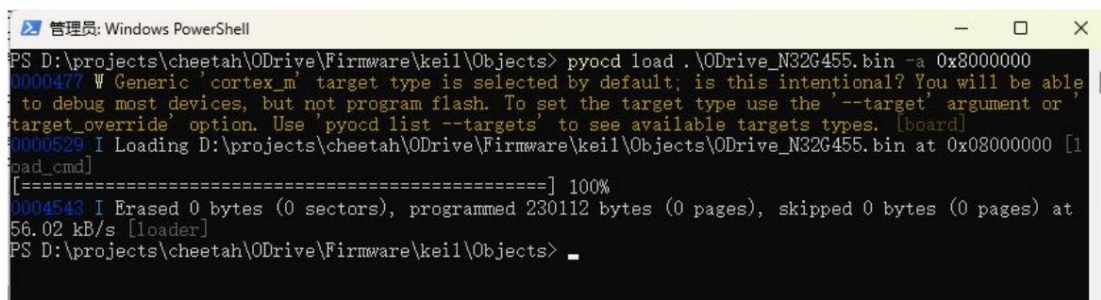


```

管理员: Windows PowerShell
PS D:\projects\cheetah\ODrive\Firmware\keil\Objects> pyocd list
#  Probe/Board      Unique ID          Target
-----
0  STM32 STLink      6000470018000037544B524E  n/a
PS D:\projects\cheetah\ODrive\Firmware\keil\Objects>
  
```

Then, execute the following command to burn the bin file:

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



```

管理员: Windows PowerShell
PS D:\projects\cheetah\ODrive\Firmware\keil\Objects> pyocd load .\ODrive_N32G455.bin -a 0x8000000
0000477 W Generic 'cortex_m' target type is selected by default; is this intentional? You will be able
to debug most devices, but not program flash. To set the target type use the '--target' argument or
'target_override' option. Use 'pyocd list --targets' to see available targets types. [board]
0000529 I Loading D:\projects\cheetah\ODrive\Firmware\keil\Objects\ODrive_N32G455.bin at 0x08000000 [1
load_cmd]
[=====] 100%
0004543 I Erased 0 bytes (0 sectors), programmed 230112 bytes (0 pages), skipped 0 bytes (0 pages) at
56.02 kB/s [loader]
PS D:\projects\cheetah\ODrive\Firmware\keil\Objects>
  
```

3.2.2 National download software

1. USB (DFU) programming

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

First, download the national burning software (<https://cyberbeast.cn/filedownload/766844>) and unzip it to any directory. and run.

Then, connect the Type-C interface, enter odrivetool, and execute the following command to put the drive into DFU mode:

```
odrv0.enter_dfu_mode()
```

Finally, use the national burning software to perform programming, as shown in the figure below. Please note that after programming is completed, please click "Common Operations" "Operation" and then click "Reset" to restart the drive and connect and debug normally through odrivetool.



2. SWD (JLink or DAP) programming

Downloading using SWD is similar to DFU mode, but it needs to be connected through the SWD debugging interface (2.4.4), and in Select the corresponding debugging tool (JLink or DAP) in the picture above.

3.2.3 Motor Wizard (coming soon)

4 Communication protocols and examples

4.1 CAN protocol

The default communication interface is CAN, with a maximum communication rate of 1Mbps (can be read and set through `odrv0.can.config.baud_rate` settings). **Please note: USB is incompatible with CAN in early hardware versions (less than or equal to 3.7). Please refer to 3.1.2 How to download from USB Switch to CAN.**

4.1.1 Protocol frame format

CAN communication adopts standard frame format, data frame, 11-bit ID, 8-byte data, as shown in the table below (MSB on the left, MSB on the right LSBÿÿ

Data field	CAN IDÿ11bitsÿ		Dataÿ8 bytesÿ
segmentation Bit10 ~ Bit5	Bit4 ~ Bit0		Byte0 ~ Byte7
description node_id	cmd_id		communication data

ÿ node_id: represents the unique ID of this motor on the bus, which can be used in odrivetool

`odrv0.axis0.config.can.node_id` to read and set.

ÿ cmd_id: command encoding, indicating the message type of the protocol, please refer to the rest of this section.

ÿ Communication data: 8 bytes. The parameters carried in each message will be encoded into integers or floating point numbers. The byte order is

Little endian, where floating point numbers are encoded according to the IEEE 754 standard (available via the website

<https://www.h-schmidt.net/FloatConverter/IEEE754.html> test encoding).

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

ÿ 11-bit CAN ID=(0x05<<5)+0x0C=0xAC

ÿ According to the description of Set_Input_Pos in 4.1.2, Input_Pos is the 4 bytes starting from the 0th byte.

The encoding is C3 F5 48 40 (the floating point number 3.14 is encoded as 32-digit number 0x4048f5c3 using the IEEE 754 standard), Vel_FF

The 2 bytes starting from the 4th byte are encoded as E8 03 (1000=0x03E8), and Torque_FF is at the 6th byte.

The first 2 bytes of the byte are encoded as 88 13 (5000=0x1388), then the 8-byte communication data is:

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

4.1.2 Frame message

The following table lists all available messages:

CMD ID name			parameter
0x001	Heartbeat	Directional motorÿhost	Axis_Error Axis_State Motor_Flag Encoder_Flag Controller_Flag
0x002	Stop	Hostÿmotor	
0x003	Get_Error	MotorÿHost	Error_Type
0x004	RxSdo	MotorÿHost	
0x005	TxSdo	MotorÿHost	
0x006	Set_Axis_Node_ID	HostÿMotor	Axis_Node_ID
0x007	Set_Axis_State	HostÿMotor	Axis_Requested_State
0x009	Get_Encoder_Estimates	MotorÿHost	Pos_Estimate Or_Estimate
0x00B	Set_Controller_Mode	Hostÿmotor	Control_Mode Input_Mode
0x00C	Set_Input_Pos	Hostÿmotor	Input_Pos Well_FF Torque_FF
0x00D	Set_Input_Vel	Hostÿmotor	Input_Vel Torque_FF
0x00E	Set_Input_Torque	HostÿMotor	Input_Torque
0x00F	Set_Limits	HostÿMotor	Velocity_Limit Current_Limit
0x010	Start_Anticogging	Hostÿmotor	
0x011	Set_Traj_Vel_Limit	HostÿMotor	Traj_Vel_Limit
0x012	Set_Traj_Accel_Limits	HostÿMotor	Traj_Accel_Limit Traj_Decel_Limit
0x013	Set_Traj_Inertia	HostÿMotorÿHost	Traj_Inertia
0x014	Get_Iq		Iq_Setpoint Iq_Measured
0x015	Get_Sensorless_EstimatesMotor ÿHost		Pos_Estimate Or_Estimate
0x016	Reboot	Hostÿmotor	
0x017	Get_Bus_Voltage_Current motorÿhost		Bus_Voltage Bus_Current
0x018	Clear_Errors	Hostÿmotor	
0x019	Set_Linear_Count	HostÿMotor	Linear_Count
0x01A	Set_Pos_Gain	HostÿMotor	Pos_Gain
0x01B	Set_Vel_Gains	HostÿMotor	Vel_Gain Vel_Integrator_Gain
0x01C	Get_Torques	MotorÿHost	Torque_Setpoint Torque

0x01D	Get_Powers	Motor↔Host	Electrical_Power Mechanical_Power
0x01E	Disable_Can	Host↔motor	
0x01F	Save_Configuration	Host↔motor	

Detailed descriptions of all messages are as follows:

↔ Heartbeat

CMD ID: 0x001 (motor↔host)

Starting 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 is 0
6	Encoder_Flag	uint8	1: odrv0.axis0.encoder.error is not 0 0: odrv0.axis0.encoder.error is 0
7	Controller_Flag	uint8	bit7↔odrv0.axis0.controller.trajectory_done bit0↔ 1: odrv0.axis0.controller.error is not 0 0: odrv0.axis0.controller.error is 0

↔ Stop

CMD ID: 0x002 (host↔motor) has no parameters and no data.

This command will cause the motor to stop in an emergency and report an ESTOP_REQUESTED exception.

↔ Get_Error

CMD ID: 0x003 (motor↔host)

Input (host↔motor):

Starting byte	name	type	illustrate
0	Error_Type	uint8	0: Get motor exception 1: Get encoder exception 2: Obtain insensitive exceptions 3: Get controller 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.error

			2ȳodrv0.axis0. 3ȳodrv0.axis0.controller.error
--	--	--	--

ȳ RxSdo

CMD ID: 0x004 (HostȳMotor)

Input:

Starting byte name		type	illustrate
0	opcode	uint8	0: read 1: write
1	Endpoint_ID	uint16 Please download the IDs corresponding to all parameters and interface functions.	JSON file: https://cyberbeast.cn/filedownload/784822
3	reserved	uint8	
4	Value	uint8[4] varies according to Endpoint_ID, please refer to the above	Description in JSON. For example, Endpoint_ID corresponds to a Float value that can be read and written, then the 4 bytes here are IEEE The encoded float value, when opcode=1, this value Write this float value.

Output (when the above opcode=0):

Starting byte name	opcode	type	illustrate
0		uint8	Fixed to 0
1	Endpoint_ID	uint16 Please download the IDs corresponding to all parameters and interface functions.	JSON file: https://cyberbeast.cn/filedownload/784822
3	reserved	uint8	
4	Value	uint8[4] varies according to Endpoint_ID, please refer to the above	Description in JSON. For example, Endpoint_ID corresponds to a Readable uint32 value, then the 4 bytes here are little endian Endian uint32.

ȳ TxSdo

CMD ID: 0x005 (motortȳhost)

The usage is the same as RxSdo when opcode=1.

ȳ Set_Axis_Node_ID

CMD ID: 0x006 (HostȳMotor)

Starting byte name		type	odrivetool access
0	Axis_Node_ID	uint32	odrv0.axis0.config.can.node_id

ÿ Set_Axis_State

CMD ID: 0x007 (HostÿMotor)

Starting byte	name	type	odrivetool access
0	Axis_Requested_State	uint32	odrv0.axis0.requested_state

ÿ Get_Encoder_Estimates

CMD ID: 0x009 (motorÿhost)

Starting byte	name	type	unit	odrivetool access
0	Pos_Estimate	float32	rev	odrv0.axis0.encoder.pos_estimate
4	Or_Estimate	float32	rev/s	odrv0.axis0.encoder.vel_estimate

ÿ Get_Encoder_Count

CMD ID: 0x00A (motorÿhost)

Starting 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

ÿ Set_Controller_Mode

CMD ID: 0x00B (HostÿMotor)

Starting 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

ÿ Set_Input_Pos

CMD ID: 0x00C (HostÿMotor)

Starting byte	name	type	unit	odrivetool access
0	Input_Pos	float32	rev	odrv0.axis0.controller.input_pos
4	Well_FF	int16	0.001rev/s	odrv0.axis0.controller.input_vel
6	Torque_FF	int16	0.001Nm	odrv0.axis0.controller.input_torque

ÿ Set_Input_Vel

CMD ID: 0x00D (HostÿMotor)

Starting 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
---	-----------	------------	--	-------------------------------------

ÿ Set_Input_Torque

CMD ID: 0x00E (HostÿMotor)

Starting byte	name	type unit		odrivetool access
0	Input_Torque	float32 Nm		odrv0.axis0.controller.input_torque

ÿ Set_Limits

CMD ID: 0x00F (HostÿMotor)

Starting byte	name	Type unit	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

ÿ Start_Anticogging

CMD ID: 0x010 (HostÿMotor)

Perform torque ripple calibration.

ÿ Set_Traj_Vel_Limit

CMD ID: 0x011 (HostÿMotor)

Starting byte	name	Type unit	odrivetool access
0	Traj_Vel_Limit	float32 rev/s	odrv0.axis0.traj_traj.config.vel_limit

ÿ Set_Traj_Accel_Limits

CMD ID: 0x012 (HostÿMotor)

Starting byte	name	Type unit	odrivetool access
0	Traj_Accel_Limit	float32 rev/s^2	odrv0.axis0.traj_traj.config.accel_limit
4	Traj_Decel_Limit	float32 rev/s^2	odrv0.axis0.traj_traj.config.decel_limit

ÿ Set_Traj_Inertia

CMD ID: 0x013 (HostÿMotor)

Starting byte	name	type unit		odrivetool access
0	Traj_Inertia	float32 Nm/(rev/s^2)	odrv0.axis0.controller.config.inertia	

ÿ Get_Iq

CMD ID: 0x014 (motorÿhost)

Starting byte	name	Type	unit	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

ÿ Get_Sensorless_Estimates

CMD ID: 0x015 (motorÿhost)

Starting byte	name	Type	unit	odrivetool	access
0	Pos_Estimate	float32	rev		odrv0.axis0.sensorless_estimator.pll_pos
4	Or_Estimate	float32	rev/s	odrv0.axis0	sensorless_estimator.vel_estimate

ÿ Reboot

CMD ID: 0x016 (HostÿMotor)

ÿ Get_Bus_Voltage_Current

CMD ID: 0x017 (motorÿhost)

Starting byte	name	Type	unit	odrivetool	access
0	Bus_Voltage	float32		IN	odrv0.vbus_voltage
4	Bus_Current	float32	A		odrv0.ibus

ÿ Clear_Errors

CMD ID: 0x018 (HostÿMotor)

Clear all errors and exceptions.

ÿ Set_Linear_Count

CMD ID: 0x019 (HostÿMotor)

Set the encoder absolute position.

Starting byte	name	type	odrivetool	access
0	Linear_Count	int32	odrv0.axis0.encoder.set_linear_count()	

ÿ Set_Pos_Gain

CMD ID: 0x01A (HostÿMotor)

Starting byte	name	type	unit		odrivetool	access
0	Pos_Gain	float32		(rev/s)/rev	odrv0.axis0.controller.config.pos_gain	

• Set_Vel_Gains

CMD ID: 0x01B (Host→Motor)

Starting byte	name	type	unit	odrivetool access
0	Vel_Gain	float32	Nm/(rev/s)	odrv0.axis0.controller.config.vel_gain
4	Speed_Integrated r_Gain	float32	Nm/rev	odrv0. axis0.controller.config.vel_integrator_gain

• Get_Torques

CMD ID: 0x01C (motor→host)

Starting 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)

Starting 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 practice

4.1.3.1 Practical combat: power-on calibration

The sequence for sending CAN messages is as follows:

CAN ID	frame type	frame data 0x007 data	illustrate
frame 04 00 00	00 00 00 00 00		Message: Set_Axis_State Parameters: 4 Calibrate the motor

0x007 Data frame	07 00 00 00 00 00 00 00	Message: Set_Axis_State Parameters: 7 Calibrate the encoder
------------------	-------------------------	---

4.1.3.2 Practical combat: speed control

The sequence for sending CAN messages is as follows:

CAN ID	frame type	frame data 0x00B data	
frame 02	00 00 00 00 02 00 00 00		Description message: Set_Controller_Mode Parameter: 2/2 Set the control mode to speed control, input mode to speed ramp
0x007 Data frame	08 00 00 00 00 00 00 00		message: Set_Axis_State Parameter: 8 Enter closed-loop control
0x0D data frame	00 00 20 41 00 00 00 00		status message: Set_Input_Vel Parameter: 10/0 Set the target speed and torque feedforward, where The target speed is 10 (floating point number: 0x41200000), and the torque feedforward is 0 (floating point number: 0x00000000)

4.1.3.3 Practical combat: position control

The sequence for sending CAN messages is as follows:

CAN ID	frame type	frame data 0x00B data	
frame 03	00 00 00 00 03 00 00 00		Description message: Set_Controller_Mode Parameter: 3/3 Set the control mode
0x007 Data frame	08 00 00 00 00 00 00 00		to position control and the input mode to position filtering
message: Set_Axis_State	Parameter: 8	Enter the closed-loop control state	0x0C Data frame CD CC 0C 40 00 00 00 00 Message: Set_Input_Pos Parameter: 2.2/0/0 Set the target position, speed feedforward and torque feedforward, where the target position is 2.2 (floating point number: 0x400CCCD), and the torque feedforward and speed feedforward are 0

4.1.4 CANOpen Compatibility

If the node ID is properly assigned, it can communicate with CANOpen. The following table lists the valid nodes for CANOpen and this protocol

ID combination:

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 messages

The user can configure the motor to periodically send messages to the host computer instead of the host computer sending request messages to the motor. accessible

A series of configurations under `odrv0.axis0.config.can` to turn on/off periodic messages (a value of 0 means turned off, other values means

Cycle time, unit is ms), as shown in the following table:

information	odrivetool configuration	default value
Heartbeat	<code>odrv0.axis0.config.can.heartbeat_rate_ms</code>	100
Get_Encoder_Estimates	<code>odrv0.axis0.config.can.encoder_rate_ms</code>	10
Get_Motor_Error	<code>odrv0.axis0.config.can.motor_error_rate_ms</code>	0
Get_Encoder_Error	<code>odrv0.axis0.config.can.encoder_error_rate_ms</code>	0
Get_Controller_Error	<code>odrv0.axis0.config.can.controller_error_rate_ms</code>	0
Get_Sensorless_Error	<code>odrv0.axis0.config.can.sensorless_error_rate_ms</code>	0
Get_Encoder_Count	<code>odrv0.axis0.config.can.encoder.count_rate_ms</code>	0
Get_Iq	<code>odrv0.axis0.config.can.iq_rate_ms</code>	0
Get_Sensorless_Estimates	<code>odrv0.axis0.config.can.sensorless_rate_ms</code>	0
Get_Bus_Voltage_Current	<code>odrv0.axis0.config.can.bus_vi_rate_ms</code>	0

By default, the first two periodic messages are turned on when shipped, so when users monitor the CAN bus, they will see both types of messages.

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

Information is broadcast at a set period. Users can turn them off with the following command:

For details of each message, see 4.1.2.

4.2 Python SDK

Please first follow the steps in Section 3.1 to install `odrivetool` (`pip install --upgrade odrive`). See 3.1.3, available

Use all the instructions described in this section for Python development.

Here are three examples:

```
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()
```

4.2.1 Practical combat: power-on calibration

```
import odrive
import time

odrv0 = odrive.find_any()
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.VELOCITY_CONTROL
odrv0.axis0.controller.config.input_mode=odrive.utils.InputMode.VEL_RAMP
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
```

4.2.2 Practical combat: speed control

```
import odrive

odrv0 = odrive.find_any()
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.POSITION_CONTROL
odrv0.axis0.controller.config.input_mode=odrive.utils.InputMode.POSITIVE_INTEGER
odrv0.axis0.requested_state=odrive.utils.AxisState.CLOSED_LOOP_CONTROL
odrv0.axis0.controller.input_pos=10
```

4.2.3 Practical combat: position control

4.2.4 Practical combat: data collection

During the R&D integration process, users often need to collect motor operation data, such as collecting voltage and current changes, position and speed changes, etc. The Python SDK integrates powerful data capture capabilities and can use simple scripts to capture massive operating data, making development and integration easier.

The following code is based on the previous position control example, adding the function of real-time position and current data capture, and

```
import odrive
import numpy as np

odrv0 = odrive.find_any()
cap =
odrive.utils.BulkCapture(lambda:[odrv0.axis0.motor.current_control.lq_
measured,odrv0.axis0.encoder.pos_estimate],data_rate=500,duration=2.5)
odrv0.axis0.controller.config.control_mode=odrive.utils.ControlMode.POSI
TION_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
np.savetxt("d:/test.csv",cap.data,delimiter=',')
```

Save the data into a csv file.

In the statement for BulkCapture, data_rate represents the sampling frequency, the unit is hz, and duration represents sampling.

Time, the unit is seconds, the lambda expression 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, and the underlying protocol is as described in 4.1. Compatible hardware/libraries:

- Arduino with built-in CAN interface, such as Arduino UNO R4 Minima, Arduino UNO R4 WIFI, etc.

- Teensy development boards with built-in CAN interface can use the adapted FlexCAN_T4 library (Teensy 4.0 and Teensy 4.1)

to visit

- Other Arduino compatible boards can be accessed using the MCP2515 based CAN expansion board

Here is an example showing how to configure a motor to respond to Arduino position control commands:

- Configure motor

In addition to the basic configuration in 3.1.2, configure the control to have a control bandwidth of 20rad/s (the sending speed of Arduino Uno is limited,

Therefore, the control bandwidth does not need to be too high. If you use a faster Arduino, you can increase this bandwidth value):

- Configure CAN

Configure CAN as follows:

- Install ODriveArduino library

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

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

ÿ 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

// frequency 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 (ldrv0.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 have been tested and verified on Ubuntu 23.04 and ROS2 Iron, but are not supported on MAC and Windows platforms.

It has not been verified on other ROS2 versions and needs to be corrected before it can be used.

4.4.1 Install odrive_can package

1. Create a new ROS2 workspace (see <https://docs.ros.org/en/iron/index.html>)
2. Use git clone https://github.com/odriverobotics/odrive_can Download the code to the src directory in the above workspace directory

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

3. Go to the root directory of the workspace in the terminal and run:

4. Environment preparation before running:

5. Run the routine node:

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

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

4.4.2 Calling services and viewing messages

Assume that the above `odrive_can_node` node runs in the namespace `odrive_axis0` (can

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
```

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

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

5 Frequently Asked Questions and Exception Codes (to be updated)

5.1 Frequently Asked Questions (FAQ)

5.2 Exception code