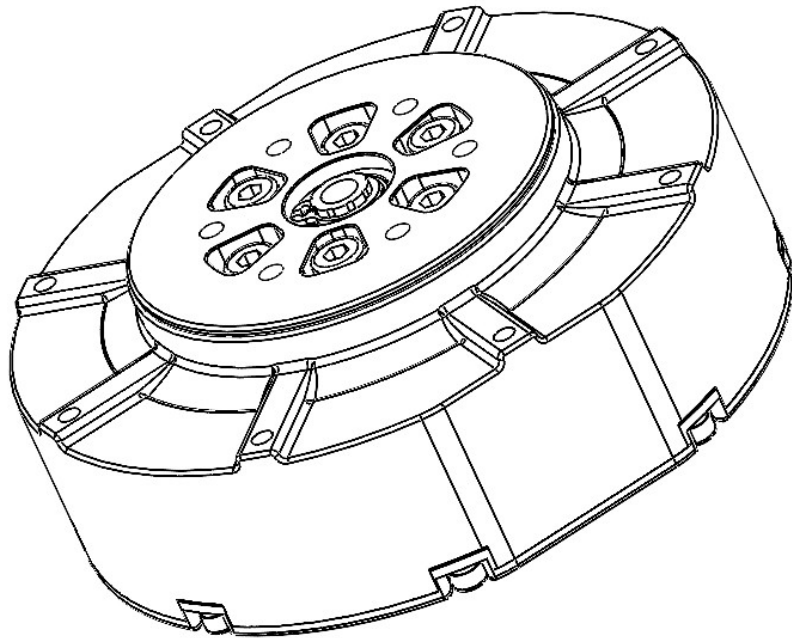


# GO-M8010-6 Motor

User Manual V1.0



**Unitree**

[www.unitree.cc/cn](http://www.unitree.cc/cn)

# Catalogue

Disclaimer .....	2
Symbol Description .....	2
Notes for Using .....	2
1. Motor Description .....	3
2. FOC Introduction .....	3
3. Motor Hardware and Encoder Overview .....	4
4. Hybrid Control of Motor .....	5
5. Motor Wire Connection .....	5
6. Motor Configuration .....	7
6.1 Viewing the Serial Port Name .....	7
6.2 Motor acquisition kit .....	7
6.3 Viewing the Motor ID .....	8
6.4 Change the Motor ID .....	8
6.5 Motor Firmware Upgrade .....	9
6.6 Switch Back to Motor Mode .....	9
7. Motor Control .....	10
7.1 C++ Example Trial Run .....	10
7.2 Position Mode .....	11
7.3 Speed Mode .....	12
7.4 Damping Mode .....	12
7.5 Torque Mode .....	13
7.6 Zero Torque Mode .....	13
7.7 Force Position Mixed Mode .....	13
8. Migrate to Other Upper Computer Platforms .....	14
8.1 Communication Configuration .....	14
8.2 Motor Movement Control Command Format .....	14

## Disclaimer

Thank you for purchasing the GO-M8010-6 motor. Please read this statement carefully before use, and once used, it is deemed to be a recognition and acceptance of the entire contents of this statement. Please strictly follow the manual to use this product. Unitree Robotics will not bear any liability for any loss caused by improper use, installation, and modification by users.

Unitree is a trademark of Unitree Robotics. The product names and brands, etc. appear in this article are trademarks or registered trademarks of the company to which they belong. Unitree Robotics owns the copyright of this product and the user manual. It may not be reproduced or reprinted in any form without our permission.

There may be semantic differences in the disclaimer in different languages. Please follow the Chinese version in China and the English version in other regions.

## Symbol Description



Important Note



Operation, Tips for Use

## Notes for Using



- If used improperly, the large torque of the motor may cause serious injury and damage to personal property. Therefore, when using, it is very important to pay attention to safety.
- The GO-8010-6 motor has a large torque. To avoid potential safety risks, please operate with caution. Non-professional users and persons under the age of 18 should not use it.

1) The maximum allowable voltage of the GO-8010-6 motor is 30V, please be sure to operate following the relevant safety regulations.

2) When using AC-DC/DC-DC switching power supply, it is forbidden to insert the XT30 2+2 power supply connector with power on. Please connect the motor power supply plug before turning on the power supply. (Because the voltage range of some SMPS is insufficient, power-on will cause a strong peak voltage and damage the motor drive)

3) If there is a [power generation condition](#) when the motor is working, please ensure that your power supply has the ability to recover or use battery power to avoid overloading the bus voltage.

4) Please pay attention to controlling the motor parameters when using, do not make the output torque too large or install the large inertia load at the end of the motor.

5) Please check if the motor is normal and if there is a motor stall, etc. before use. If any abnormal, please replace it in time.

6) Please pay attention to the temperature of the motor and do not touch its surface to avoid burns.

7) When installing and unloading the mechanical structure of the output terminal, be sure to disconnect the motor power.

## 1. Motor Description

The stator of a permanent magnet synchronous motor is a three-phase symmetrical sine wave winding. We know that in a fixed magnetic field, the permanent magnet will rotate and fix in a direction parallel to the magnetic field. A typical example is a compass under the earth magnetic field, which turns in the direction of the north and south poles of the geomagnetic magnet. Similarly, if this fixed magnetic field begins to rotate, then the permanent magnet will also rotate, trying to follow the direction of the parallel magnetic field, so that we can rotate the permanent magnet to a specified angle by rotating the magnetic field. At the same time, the torque generated by the permanent magnet in the magnetic field and the angle between the permanent magnet and the direction of the magnetic field are related, so we can also control the torque generated by the permanent magnet by controlling the angle between the magnetic field and the permanent magnet.

Back to the perspective of the motor, that is, we can control the angular position and output torque of the rotor by controlling the size and on-off of the three winding voltages on the stator. This control method is the vector control of permanent magnet synchronous motor (Field-Oriented Control, hereinafter referred to as FOC).

## 2. FOC Introduction

FOC control has many unique advantages. It allows us to have the pixel-level control of the permanent magnet synchronous motor and obtain many effects that cannot be achieved by traditional motor control methods:

- 1) Precise control can be maintained at low speed.
- 2) Motor commutation rotation can be well realized.
- 3) It can control the motor in three closed loops of torque, speed, and position.
- 4) FOC-controlled permanent magnet synchronous motor has small noise.

As described in the permanent magnet synchronous motors section, the basis of FOC control is the rotating magnetic field. In the magnetic field vector of figure 1, we can see that the three coils of the motor stator a, b, and c can generate magnetic fields in three directions,  $B_a$ ,  $B_b$ , and  $B_c$ , which can synthesize the magnetic field  $B$  in the motor. According to the angle, angular velocity, expected output torque of the current permanent magnet rotor and the current of coils A, B and C obtained by sampling and measurement, the FOC controller calculates the voltage on-off state and on-off time of the three coils, and then controls the voltage on-off of the three coils through MOS tube. In this way, we can synthesize the desired magnetic field  $B$ , and then drag the motor rotor to move in the desired way.

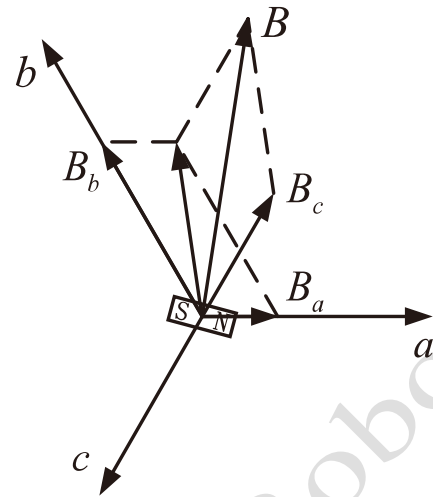


Figure 1 Magnetic Field Vector of FOC

### 3. Motor Hardware and Encoder Overview

The core components of the robot joint motor are the motor drive plate, stator, rotor, and planetary reducer. The motor is suitable for working under the condition of high speed and low torque, while our robot needs low speed and high torque, so the motor rotor needs to be slowed down by a reducer and then output torque.

Encoders are sensors used to measure the angle of rotation. Encoders are divided into incremental encoders, multiturn absolute position encoders and single-turn absolute position encoders and so on. Here we only discuss the single-turn absolute position encoder which is practically used by joint motor.

The single-turn absolute position encoder of the joint motor is mounted on the motor rotor. For a single-turn absolute position encoder (hereinafter referred to as an encoder), we can see it as a "clock dial" and every time we look at it, we can read the current date and time, such as 23:00 on April 1. If 2 hours have passed, the clock will turn to 24:00 on April 1, the date will be increased by 1 day to become April 2, and the time will be re-clocked from 0:00 to 1:00 on April 2. To facilitate the calculation of the elapsed time, we can also say that now it is 25:00 on April 1. It seems that 25 o'clock exceeds the 24-hour range of the day, but it is because the date has increased by 1 day.

The same is true for single-turn absolute position encoders. Every time the system is powered on, the rotor may be in any position. and the encoder will tell us angular position of the rotor (a value between 0 and  $2\pi$ ). If the rotor rotates over the angular position of  $2\pi$ , the encoder can also record that the number of turns increased by 1, and it will output an angular position beyond the range of 0 to  $2\pi$ . It seems that the encoder can also output an angular position of more than one turn, so why we call it the "single-turn" absolute position encoder? The reason is that this kind of encoder cannot store the number of previous rotations after the power outage, let's illustrate with the following example.

Suppose the current angle value of the encoder output is  $2.3\pi$ , which means that the encoder passes through  $2\pi$  after power-on, the number of rotations change from 0 to 1, and the encoder is currently located at  $0.3\pi$ . At this

moment, we turn off the encoder, do not rotate, then turn on the encoder, the angle value of the encoder output will become  $0.3\pi$ . Because the number of rotations is reset to 0 after shutdown, the encoder will only output the current position of  $0.3\pi$ .

## 4. Hybrid Control of Motor

The joint motor, as a highly integrated power unit, has been encapsulated with the control algorithm of the motor underlying. As a user, you only need to send the relevant commands to the joint motor, and the motor can complete all the work from receiving the command to the joint torque output.

For the underlying control algorithm of the motor, the only control target required is the output torque. However, for robots, we usually need to set the position, speed, and torque for the joints. Currently, the joint motor needs to be mixed control.

The joint motor of Unitree Robotics includes the following five control commands:

- 1) Feedforward torque:  $\tau_{ff}$ ;
- 2) Expected angle position:  $p_{des}$ ;
- 3) Expected angular velocity:  $\omega_{des}$ ;
- 4) Position stiffness:  $k_p$ ;
- 5) Velocity stiffness (damping):  $k_d$ .

In the hybrid control of the joint motor, the PD controller is used to provide feedback of deviation from the motor at the output position to the torque output:

$$\tau = \tau_{ff} + k_p \times (p_{des} - p) + k_d \times (\omega_{des} - \omega)$$

In the formula,  $\tau$  is the output torque of the motor rotor of the joint motor,  $p$  is the current angular position of the motor rotor, and  $\omega$  is the angular velocity of the motor rotor. In the actual use of the joint motor, it is necessary to pay attention to the conversion of the control target amount of the motor output and the command of the motor rotor sent.

## 5. Motor Wire Connection

We use RS-485 as the physical layer. The so-called physical layer refers to the use of physical phenomena to express and transmit information. In fact, RS-485 can be equivalent to a half-duplex serial port (UART).

RS-485 uses the potential difference between two wires to transmit digital levels. Usually, the two wires are twisted pair wires of equal length. When there is external interference, noise will be generated on the pair of wires at the same time. After the differential level is transferred to the motor, the differential operational amplifier will subtract the input differential level to restore the original level, and then transfer it to the serial port, which makes the communication have high anti-interference ability.

Table: Relationship between differential level and logic level

RS-485 signal states		
Signal	Mark (logic 1)	Space (logic 0)
A	Low	High
B	High	Low

Since there are only two RS-485 data lines, which are used to transmit one level of two differential signals, the line can only communicate in one direction at the same time. This is also the reason why RS-485 is half duplex communication. The full duplex differential transmission protocol of the four data lines is RS-422. But using RS-485 greatly simplifies cable connection, improves hardware communication reliability, and reduces costs.

Generally, half duplex communication requires a mainframe (upper computer) to control the communication rhythm. The motion control command sent by the mainframe contains the target motor ID information, so that each motor will judge whether the ID information is consistent after receiving the motion control command, and the matching motor will send the internal data information to the mainframe, thus completing a round of communication.

The Go-M8010-6 motor supports up to 15 motors (ID 0~14) on one bus, and there must be no motors with the same ID on the bus, otherwise the communication of the whole bus will be abnormal. Please refer to 6.2 for the configuration of motor ID.

To send the movement control command to the joint motor manufactured by Unitree Robotics, we need to send the command through the serial port. The motor communicates with the upper computer through RS-485 interface, and the fixed baud rate is 4Mbps. For the convenience of users, we will provide a USB to RS-485 adapter.

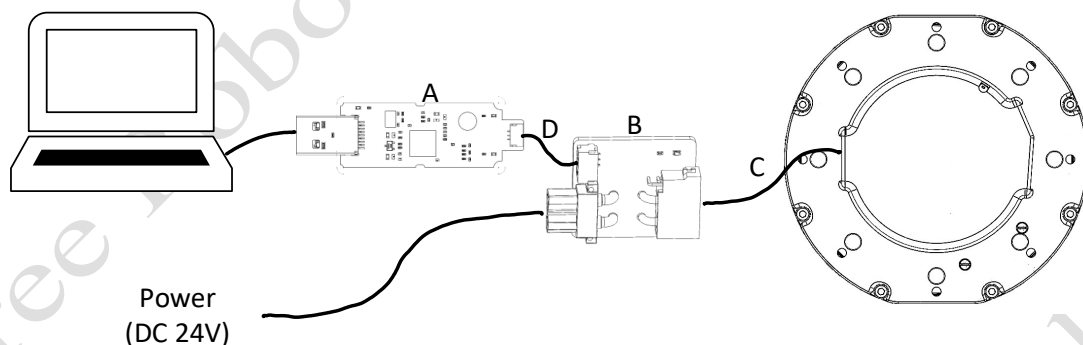


Figure 2 Motor Wire Connection

As shown in figure 2, connect C (XT30(2+2) cable) to B (XT30(2+2) conversion board), and connects to A (485 to USB module) through D (GH1.25-3 cable) and connects to the computer.

When controlling the motor through your own computer, you need to connect the RS-485 interface to the upper computer through the USB to RS-485 interface to send instructions from the upper computer to the motor. After

the 24V DC power supply is connected, the green indicator light of the motor starts to flash, indicating that the motor has been started.



- Please check the power supply before power on. Do not let the power supply exceed 30V, and check whether the power supply capacity is sufficient.

## 6. Motor Configuration

### 6.1 Viewing the Serial Port Name

When the USB to RS-485 interface is connected to the upper computer, the upper computer will assign a serial port name to the serial port. In Linux systems, the serial port name usually starts with "ttyUSB". In Windows systems, the serial port name usually starts with "COM".

In the Linux system, all external devices exist in the form of files. The USB to RS-485 adapter can also be considered as a "file" in the /dev folder. Open any terminal window (Ctrl+Alt+t is the shortcut key under Ubuntu) and run the following command:

```
cd /dev
ls | grep ttyUSB
```

The cd/dev command switches the current folder to /dev, and the ls |grep ttyUSB command displays all files with ttyUSB in the current folder. The character | is above the Enter key on the keyboard. Press and hold Shift+\ to type the "|". After running the above command, you can get the serial port name currently connected to the upper computer. For example, as shown in figure 3, the serial port connected to the current upper computer is named ttyUSB0. Considering the folder path of the serial port, its complete serial port name is /dev/ttyUSB0.

```
bian@bian:~$ cd /dev
bian@bian:/dev$ ls |grep ttyUSB
ttyUSB0
```

Figure 3 View Serial Port Name in Ubuntu System



- The serial number of the serial port name is consistent with the order of inserting the device, which is very helpful for connecting multiple devices.

### 6.2 Motor acquisition kit

Unzip the Unitree MotorTools toolbox. After entering the bin folder, you can see some executable programs. Here are some common tools for motor modification.

```
supereasy@ubuntu:~/Desktop/unitree_motortools/bin$ ll
total 124
drwxrwxr-x 2 supereasy supereasy 4096 Aug  6 11:55 ./
drwxrwxrwx 9 supereasy supereasy 4096 Aug  6 11:56 /
-rwxrwxr-x 1 supereasy supereasy 26744 Aug  6 11:55 changeid*
-rwxrwxr-x 1 supereasy supereasy 22376 Aug  6 11:55 swboot*
-rwxrwxr-x 1 supereasy supereasy 22520 Aug  6 11:55 swmotor*
-rwxrwxr-x 1 supereasy supereasy 38136 Aug  6 11:55 unisp*
```



Currently, we connect the XT30 2+2 interface of the motor to the motor, connect the USB to RS-485 connector to the computer, and supply power to the motor. The next step is to start configuring the motor.



- When using Unitree MotorTools, make sure that there is only one motor on the RS485 bus, and no other mainframe is working (sending data to the bus).

### 6.3 Viewing the Motor ID

To view and modify the motor ID, you need to switch the motor to the factory mode. Before switching, make sure that all the motors have stopped working and the mainframe will no longer send motion control instructions to the motor.

```
sudo ./swboot /dev/ttyUSB0
```

After a while, the green indicator light on the back of the motor that has entered the factory mode will quickly flash three times per second. At this time, all the motors that have entered the factory mode will be displayed on the terminal.

```
supereasy@ubuntu:~/Desktop/unitree_motortools/bin$ sudo ./swboot /dev/ttyUSB0
1.Motor ID 15
-----
Total 1 motors
supereasy@ubuntu:~/Desktop/unitree_motortools/bin$
```

Under normal circumstances, there will be no motors with IDs greater than 15 in the printed list. If this happens, please power up the motor and try again.

If there is a motor with an ID of 15, it means that the motor has not been set with an ID. You can refer to 6.4 for configuration.

### 6.4 Change the Motor ID

To modify the ID of the motor, you need to use the `changeid` command, which is used as follows:

```
changeid [serial port number] [original ID] [ID needs to be modified]
```

```
changeid /dev/ttyUSB0 0 1: set motor ID0 to ID1
```

Before switching, please make sure that all motors have stopped working, and the mainframe will no longer send movement control commands to the motors. For example: modify all motors with ID 15 on the bus to ID 0.

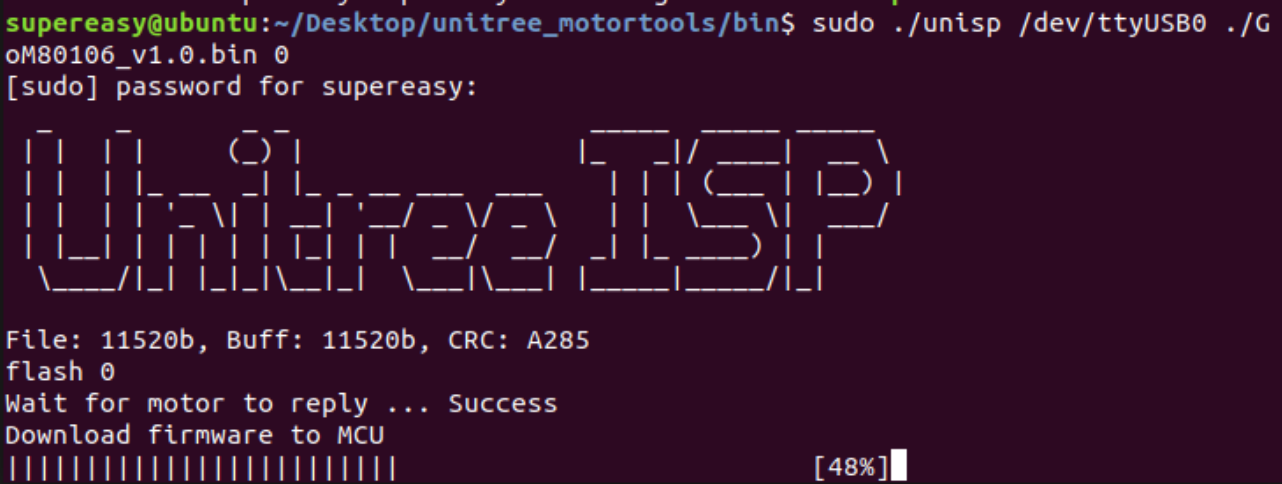
```
sudo ./changeid /dev/ttyUSB0 15 0
```


```
supereasy@ubuntu:~/Desktop/unitree_motortools/bin$ sudo ./swboot /dev/ttyUSB0
1.Motor ID 15
-----
Total 1 motors
supereasy@ubuntu:~/Desktop/unitree_motortools/bin$ sudo ./changeid /dev/ttyUSB0
15 0
supereasy@ubuntu:~/Desktop/unitree_motortools/bin$ sudo ./swboot /dev/ttyUSB0
1.Motor ID 0
-----
Total 1 motors
supereasy@ubuntu:~/Desktop/unitree_motortools/bin$
```

### 6.5 Motor Firmware Upgrade

The Go-M8010-6 motor supports the upgrade of motor firmware, which is convenient for improving motor performance and safe repair in the later period. You can download the firmware file provided by Unitree Robotics to the motor by using the unisp tool. To upgrade the motor firmware, you can use the unisp command as follows:

```
unisp [serial port number] [.bin upgrade file] [motor ID to be flashed]
unisp /dev/ttyUSB0 ./GoM80106_v1.0.bin 0
```



 • Please do not download the motor firmware of unknown origin to the motor, which is very dangerous.  
• The risk is not limited to motor brick changing, accidental injury, motor burning, loss of warranty, etc.

### 6.6 Switch Back to Motor Mode


Viewing and modifying the motor ID will enable the motor to switch to the factory mode. If it is not manually switched back to the motor mode, it will enter the factory mode even if the motor is powered on again.

The green indicator light on the back of the motor in the factory mode will flash 3 times per second.

Currently, use the command. /swmotor to switch to the motor mode. The usage is as following:

```
swmotor [Serial port number]
swmotor /dev/ttyUSB0
```

All motors on the RS485 bus can be switched to motor mode, and then the motor can receive motion control commands.

 • Motors without firmware will not be started and will be displayed on the terminal.

## 7. Motor Control

The SDK currently supports the following platforms and systems:

- 1) Linux system on the x86/ X64 platform

With each support, C++ code examples are provided, and the users only need to copy the examples to complete the motor control. Let's take the Linux system as an example to demonstrate how to control the motor.



- If you need to use SDK in Linux system under ARM32/ARM64 platform, please contact technical support.

### 7.1 C++ Example Trial Run

Next, we try to make the motor spin up.

First open the main.cpp file in the src folder, the first step is to modify the serial port name. If it only runs under one system, just modify the serial port name under this system:

```
SerialPort _ioPort("/dev/ttyUSB0");
```

Next is the statement of the command sent to the motor and the status received from the motor:

```
MotorCmd cmd;
MotorData data.
```

cmd is the control command package sent to the motor, and they are all structures of MotorCmd type. A structure is a data package that contains many different types of data. We will show the operations on the structure right away. Similarly, data is a data package that receives motor status information. It is a structure of MotorData type. For the specific contents of these two structures, please refer to the file include/unitreeMotor/unitreeMotor.h, which will not be repeated here.

Next, we modify the cmd. First, explain the data contained in the MotorCmd type structure:

- 1) Id: Target motor ID of current control command.
- 2) mode: The target motor running mode. 0. Stop 1. FOC 2. Motor calibration.
- 3) T: Feedforward torque.
- 4) W: specifies the angular velocity.
- 5) Pos: Specify the angular position.
- 6) K\_P: position stiffness.
- 7) K\_W: velocity stiffness (damping).

When the value of mode is 0, the following five control parameters have no effect. When the value of mode is 2, motor calibration is performed. In this example, we set the value of mode to 1. Here we let the motor rotate at a constant speed. The complete code is:

```
cmd.motorType = MotorType::Go2;
cmd.id = 0;
cmd.mode = 1;
cmd.K_P = 0;
cmd.W = 6.28*6.33;
cmd.K_W = 0.02;
cmd.T = 0.0;
_ioPort.sendRecv(&cmd,&data);
```

Next, we can send commands to the motor. The control command will send a control command to the motor through the sendRecv(&cmd, &data) function of the \_ioPort object, and receive the current state information of the motor once.



● It should be noted here that the commands sent to the motor are all directed to the rotor of the motor before the reducer, that is, the rotating shaft1 in Figure 5. Instead of the output shaft 2 after deceleration. Therefore, in the process of actual control, we must pay attention to the reduction ratio of the motor. In the motor of GO-8010-6, the reduction ratio is 6.33.

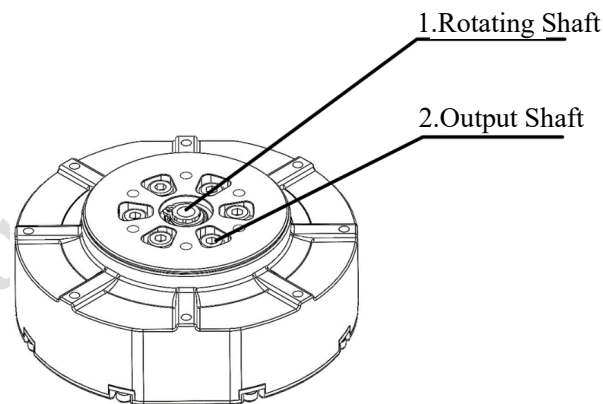


Figure 4 Motor Output Terminal

## 7.2 Position Mode

In position mode, the output shaft of the motor will be stabilized at a fixed position. For example, if we want the motor output to be fixed at 3.14 radians, we can set the control parameters as follows:

```
cmd.T = 0.0;
cmd.W = 0.0;
cmd Pos = 3.14*6.33.
cmd.K_P = 0.2;
cmd.K_W = 0.0;
```

In the above parameter settings, set T and W to 0 to realize PD control for Pos. Where K\_P is the proportional coefficient, K\_W is the differential coefficient, and 6.33 is the deceleration ratio.

After attempting to complete the above modifications, recompile and run the relevant executable files. It can be seen from the returned state of the motor that the position of the motor rotor is stable in radians, that is, the motor output is fixed at 3.14 radians. If the difference between the target position and the current position is large, the torque generated by the motor will also be large, resulting in a large electric current. If the upper limit of the output electric current of the power supply to the motor is small, power protection may occur, that is, the motor stops rotating. Currently, it is necessary to consider making cmd.Pos change slowly to avoid generating instantaneous maximum torque.

### 7.3 Speed Mode

In speed mode, the output shaft of the motor will stabilize at a fixed speed. Make the motor output shaft speed stable at 6.28rad/s:

```
cmd.T = 0.0;
cmd.W = 6.28*6.33;
cmd.Pos = 0.0;
cmd.K_P = 0.0;
cmd.K_W = 0.02;
```

Speed mode T and K\_P must be 0, which constitutes P control of W. Where K\_W is the proportional coefficient of speed.

### 7.4 Damping Mode

Damping mode is a special speed mode. When we set W=0.0, the motor will keep the rotating shaft speed at 0. And when it is rotated by an external force, an impedance torque is generated. The direction of this torque is opposite to the direction of rotation, and its magnitude is proportional to the rotation speed. When the external force rotation is stopped, the motor will stop at the current position. This state is like the linear damper, so it is called damping mode.

```
cmd.T = 0.0;
cmd.W = 0.0;
cmd.Pos = 0.0;
cmd.K_P = 0.0.
cmd.K_W = 0.02;
```

## 7.5 Torque Mode

In torque mode, the motor will continuously output a constant torque. But when the motor is idling, if a larger target torque is given, the motor will continue to accelerate until the maximum speed, and the target torque is still not reached at this time.

The following provides a relatively safe torque mode parameter setting under no load:

```
cmd.T = 0.05;  
cmd.W = 0.0;  
cmd.Pos = 0.0;  
cmd.K_P = 0.0;  
cmd.K_W = 0.0;
```

Under this parameter, the gradual acceleration process of the motor under constant torque can be observed. Because there are slight differences between each motor, if the motor cannot rotate smoothly, the value of T can be increased appropriately.

## 7.6 Zero Torque Mode

Zero torque mode is a special torque mode. When we change the command to the following setting, the motor will hold the torque of the rotating shaft to 0:

```
cmd.T = 0.0;  
cmd.W = 0.0;  
cmd.Pos = 0.0;  
cmd.K_P = 0.0;  
cmd.K_W = 0.0;
```

Currently, the motor does not stop running, but actively generates torque to resist its own friction torque. In the zero-torque mode, try to rotate the output shaft, and the resistance of the output shaft will be significantly less than that when the machine is not turned on.

## 7.7 Force Position Mixed Mode

In the actual control of quadruped robot, the robot movement controller often sends feedforward torque, target angle and target angular velocity to the joint simultaneously. Currently, the control mode is the force level hybrid control, which is the most used control mode in the later practical application.

## 8. Migrate to Other Upper Computer Platforms

Considering that some users will use a special upper computer platform to control the motor, we will also explain how to write our own motor control program here. According to the method described in this section, the readers can send control commands to the motor and receive the motor status on any platform that meets the hardware requirements.

### 8.1 Communication Configuration

The GO-M8010-6 motor adopts serial communication, communication standard is RS-485, and baud rate is 4.0 Mbps. The data bit of the serial port is 8 bit, there is no parity bit, and the stop bit is 1 bit. It should be noted that to improve the communication frequency of the motor, we use a very high baud rate of 4.0 Mbps. The users need to check whether their hardware supports such a high baud rate.

If your hardware cannot support this baud rate, you can use the USB switch to RS-485 module provided by Unitree Robotics.

### 8.2 Motor Movement Control Command Format

When controlling the motor, we will send a 17-byte command to the motor through the serial port, and then the motor will return a 16-byte message. If no command is sent to the motor, the motor will not return to the state. The format of the command sent to the motor is shown in Table 1, and the format of the status returned by the motor is shown in Table 2. These two tables detail the meaning of each byte in the received and sent message. We will explain some of the details below.

The first is the 11th and 12th bytes in the command sent to the motor. These two bytes represent the feedforward torque of the motor. Obviously, the feedforward torque is a floating-point number, namely, float type. A float type floating point number needs to occupy 4 bytes. To save communication bandwidth, we use 2 bytes to represent floating point numbers. We use the shift operation method. We will not explain the specific principle here. From the perspective of application, the reader can think that we multiplied the feedforward torque by 256, and then assigned it to a signed short int of two bytes, that is, a signed short integer variable. During this assignment, the whole number will be forced, so that we can only use two bytes to send the feedforward torque. When the motor receives this data, it only needs to divide by 256 to obtain the feedforward torque value. Although this operation will lose some accuracy, it is completely sufficient for practical application. Another thing to note is that a variable with a length of 2 bytes has 16 characters in total, that is, 16 bits. One bit is used to represent positive and negative, which is a sign bit. Therefore, only 15 bits are used to represent the size of the value, which means that the T value in the command cannot be greater than 215. Considering that we once multiplied the original data by 256, the absolute value has an upper limit:

$$|\tau_{ff}| < \frac{2^{15}}{256} = 128$$

At the same time, it should be noted that due to the forced rounding in the assignment process, the larger the value, the lower the decimal precision saved. The description of X 256 times mentioned in the description of variable T in Table 1 refers to the multiplication of 256 mentioned above, and the description of certain times mentioned in other variables is also the same.

In addition, for a 2-byte variable, its low bit is first, that is, the 13th byte, and its high bit is last, that is, the 14th byte. At the end of the sending command and receiving status, we can see a 2-byte CRC\_CCITT check. Before sending the command, we will calculate the CRC check value of these command bytes before sending and send them to the motor together with the command. After receiving the command, the motor will also calculate the CRC check value after sending according to the received command.

If no errors occur during data transmission, the pre-send CRC check value is equal to the post-send CRC check value. If there is a data error in the process of data transmission, then the calculation result of CRC check value after sending is not equal to the CRC check value before sending. The joint motor can know that the data is damaged and ignore this wrong command to help us avoid wrong data. Here we do not check the CRC algorithm to launch a specific introduction, the reader can directly refer to the Linux kernel on the `crc_ccitt` function source code.

Table 1 Mainframe Side Control Protocol

Type	Bit	Symbol	Illustration	Value
Packet Header (2Byte)	0-15 (2Byte)	HEAD	Packet Header	0xFE 0xEE
Mode Setting (1Byte)	16-19 [4bit]	ID	Target Motor ID	0,1,2,3 ... 13,14 15 Indicates broadcasting data to all motors (there is no return value on the bus at this moment)
	20-22 [3bit]	STATUS	Motor Operating Mode	0.Lock (Default) 1.FOC closed loop 2.Encoder calibration 3-7.Reserved
	23 [1bit]	Reserved		
Control Parameters (12Byte)	24-39 (2Byte)	$\tau_{set}$	Expected Motor Torque	$\tau_{ff} = 0.75 (N.m)$ $\tau_{set} = \tau_{ff} * 256$ $= 192$ $-127.99N.m \leq T_{ff} \leq 127.99N.m$
	40-55 (2Byte)	$\omega_{set}$	Expected Motor Speed	$\omega_{des} = 3.14159 (rad/s)$ $\omega_{set} = \frac{\omega_{des}}{2\pi} * 256$ $= 128$ Note: $2\pi/s = 6.28rad/s = 60RPM$ $-804.0 rad/s \leq \omega_{des} \leq 804.0 rad/s$
	56-87 (4Byte)	$\theta_{set}$	Expected Motor Output Position (multi turn accumulation)	$\theta_{des} = 90^\circ = \frac{\pi}{2} = 1.57 (rad)$ $\theta_{set} = \frac{\theta_{des}}{2\pi} * 32768$ $= 8187$ Note: $2\pi = 360^\circ = 6.2831rad$
	88-103 (2Byte)	$K_{pos}$	Motor Stiffness Coefficient/ Proportional Coefficient of Position Error	$K_p = 0.1$ $K_{pos} = K_p * 1280$ $= 128$ Note: $0 \leq K_p \leq 25.599$
	104-119 (2Byte)	$K_{spd}$	Motor amping Coefficient/ Speed Error Proportional Coefficient	$K_w = 0.2$ $K_{spd} = K_w * 1280$ $= 256$ Note: $0 \leq K_w \leq 25.599$



Verification Part (2Byte)	120-135 (2Byte)	CRC16	CRC16 Verification Results	CRC16_CCITT polynomial calculation result of 0-119 bits of data
Total:	17 Byte			

Table 2 Feedback Data of Motor Terminal

Type	Bit	Symbol	Illustration	Value
Packet Header (2Byte)	0-15 (2Byte)	HEAD	Packet Header	0xFD 0xEE Note: Different from the packet header of host machine.
Mode Information (1Byte)	16-19 [4bit]	ID	Target Motor ID	0,1,2,3 ... 13,14 15 eserved
	20-22 [3bit]	STATUS	Motor Operating Mode	0. Lock (Default) 1.FOC closed loop 2. Encoder calibration 3-7. Reserved
	23 [1bit]	Reserved		
Feedback Data (11Byte)	24-39 (2Byte)	$\tau_{fbk}$	Actual Joint Output Torque	$\tau(N.m) = \frac{\tau_{fbk}}{256}$
	40-55 (2Byte)	$\omega_{fbk}$	Actual Joint Output Speed	$\omega(rad/s) = \frac{\omega_{fbk}}{256} * 2\pi$
	56-87 (4Byte)	$\theta_{fbk}$	Actual Joint Output Position (multi turn accumulation)	$\theta(rad) = \frac{\theta_{fbk}}{32768} * 2\pi$
	88-95 [1Byte]	TEMP	Motor Temperature	unit:degree centigrade(int8_t) -128~127°C, the temperature protection is triggered when the temperature is 90 °C, and it can be controlled only after the motor is powered on again.
	96-98 [3bit]	MERROR	Error Identification	0. Normal 1. Overheating 2. Overcurrent 3. Overvoltage 4.Encoder fault 5-7. Reserved
	99-110 [12bit]	FORCE	Foot End Force	12bit original data (0-4095) Physical unit/range, to be confirmed
	111 [1bit]	Reserved		
Verification Part (2Byte)	112-127 (2Byte)	CRC16	CRC16 Verification Results	CRC16_CCITT Polynomial Calculation Results of 0-111 Bit Data
Total:	16 Byte			



- All data types in the communication protocol are integer. Please refer to the description of "bit" in the table above for specific size.
- Make sure that the controlling host processor platform is in little endian (LSB) mode.

Reserve the right to update the contents without prior notice.

**You can check the latest version of the User Manual on  
the Unitree Robotics official website.**

<https://www.unitree.com/cn>



Scan the WeChat

Follow Unitree Robotics official account