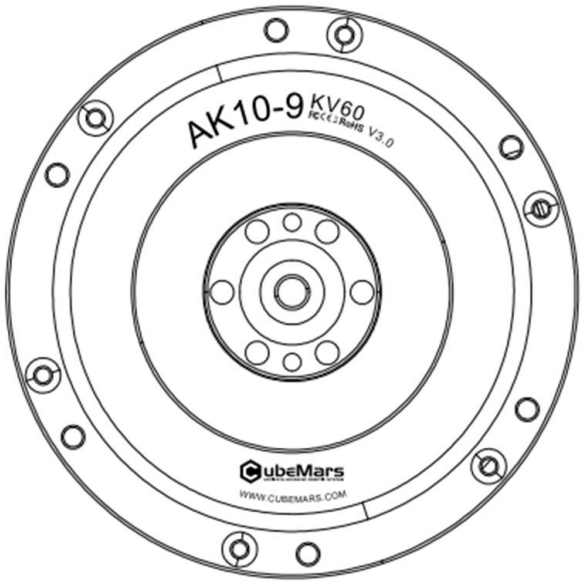
AK Series Module Product Manual

V3.0.0



E-mail:info@cubemars.com
https://www.cubemars.com/

Content

E-mail:info@cubemars.com
https://www.cubemars.com/

E-mail:info@cubemars.com
https://www.cubemars.com/

Precautions

1.Ensure that there are no short circuits in the circuit and that interfaces are connected correctly as required.

2.  The driver board will heat up during output; please use it carefully to avoid burns.

3.  Before use, please check if all parts are intact. If any parts are missing or aged, please stop using it and contact technical support in time.

4.  Please strictly follow the working voltage, current, temperature, and other parameters specified in this document; otherwise, it will cause permanent damage to the product!

Product Features

The AK series motor driver board adopts high-performance drive chips in the same class, uses Field Oriented Control (FOC) algorithm, and is equipped with advanced self-disturbance control technology for speed and angle control. It can be used with CubeMarsTool parameter setting software for parameter setting and firmware upgrades. In terms of hardware, the inner loop uses a 16-bit high-precision encoder, supporting up to 21 bits (custom firmware required), and the CAN communication uses a safer isolated CAN interface, along with a more reliable plug, greatly enhancing the reliability of the product's use and communication; in terms of software, the upper computer CubeMarsTool has been fully upgraded, and there is no need to switch between servo mode and force control mode, the control interface is more concise, and a large number of simplifications have been made in the operation, fully improving the customer's experience.

Disclaimer

Thank you for purchasing the AK series modular motor. Before using it, please read this statement carefully. Once used, it is considered as recognition and acceptance of all the contents of this statement. Please strictly follow the product manual and relevant laws, regulations, policies, and guidelines for the installation and use of the product. During the use of the product, the user promises to be responsible for their own actions and all consequences arising therefrom.

Any losses caused by improper use, installation, or modification of the product by the user, CubeMars will not assume legal responsibility.

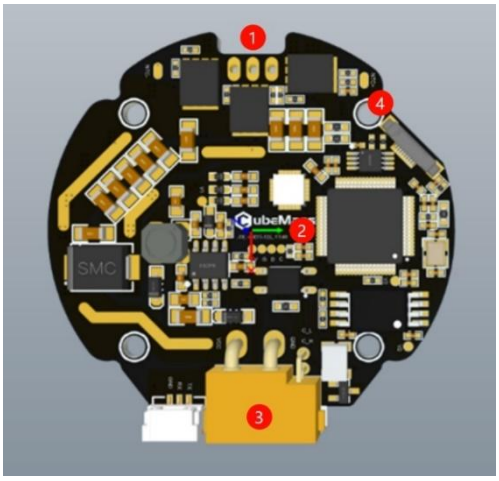
Version Change Record

Date	Version	Change
2023.10.18	Ver.2.0.0	1.First compile 2.Driver version :V3.0 3.CubeMarsTool version:V2.0
2024.07.29	Ver.2.0.1	1.1Changed the maximum working voltage to the allowable working voltage range 4.1.6Modified the example code for setting the origin 4.2 Modified the motion control protocol example code 4.3.2.1 Corrected the outer loop position formula
2024.08.06	Ver.2.0.2	Modified the cover logo 1 Modified driver board product information 2 Added section content 3.4Corrected the firmware update example picture 4.2 Modified parameter range
2024.12.04	Ver.2.0.3	1.2.2 Corrrect drive interface definition 2. Correct R-Link definition 2.1 Add VCC voltage selction switch,modify R-link specifications 3.Change CubeMars's ownership from JIANGXI XINTUO ENTERPRISE CO., Ltd. to Nanchang Kude Intelligent Technology CO., Ltd.
2024.12.11	Ver.3.0.0	The formal version change to3.0.0

E-mail:info@cubemars.com
https://www.cubemars.com/

1. Driver Product Information

1.1Driver Appearance Introduction&Product Specifications（Small Size）



①Three-phase wires connection port

②Hardware version

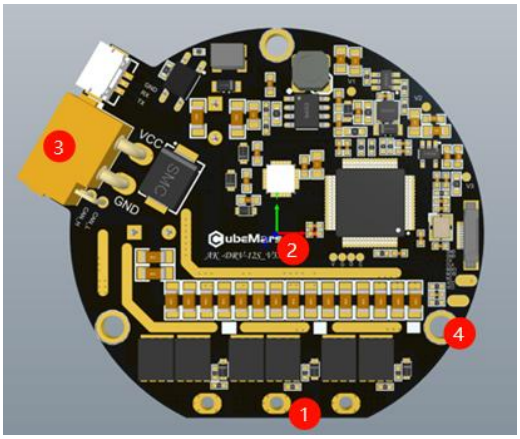
③Connection port

④Mounting holes

Production specifications（small size）	
Rated working voltage	48V
Allowable working voltage	18-52V
Rated working current	10A
Maximum current	30A
Standby power consumption	≤50mA
CAN bus bit rate	1Mbps
Size	54mm×54mm
Working environment temperature	-20℃ to 65℃
Maximum allowable temperature for control board	100℃
Inner loop encoder bits	21bit (single turn absolute)
Outer loop encoder bits (limited to dual encoder model)	15bit (single turn absolute)

E-mail:info@cubemars.com
https://www.cubemars.com/

1.2Driver Appearance Introduction&Product Specifications（Big Size）



①Three-phase wires connection port

②Hardware version

③Connection Port

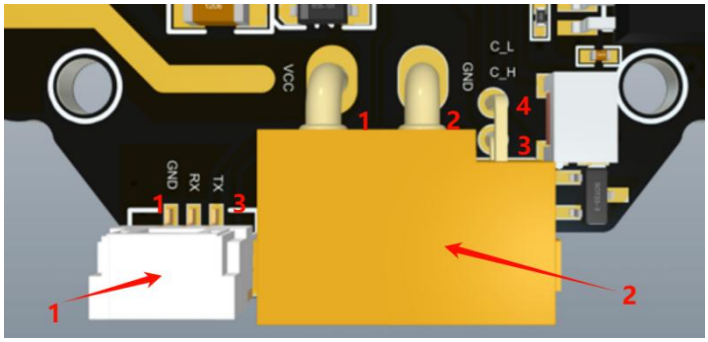
④Mounting holes

Product Specifications（Big size）	
Rated working voltage	48V
Allowable working voltage	18-52V
Rated working current	20A
Maximum current	60A
Standby power consumption	≤50mA
CAN bus bit rate	1Mbps
Size	63mm×57mm
Operation environment temperature	-20℃ to 65℃
Maximum allowable temperature for control board	100℃
Inner loop encoder bits	21bit (single turn absolute)
Outer loop encoder bits (limited to dual encoder model)	15bit (single turn absolute)

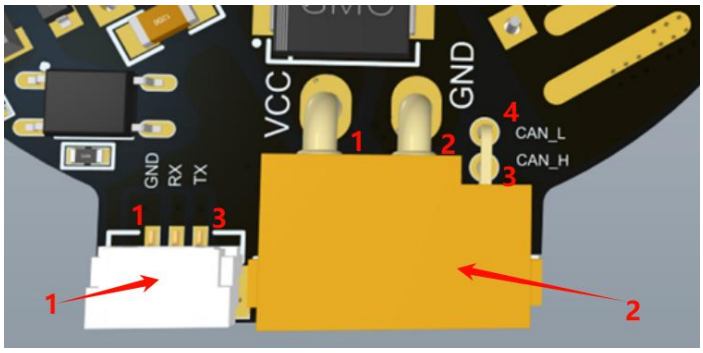
E-mail:info@cubemars.com
https://www.cubemars.com/

1.2 Driver Interface and Definition

1.2.1 Driver Interface Diagram



Small Size



Big Size

1.2.2 Driver Interface Pin Definition

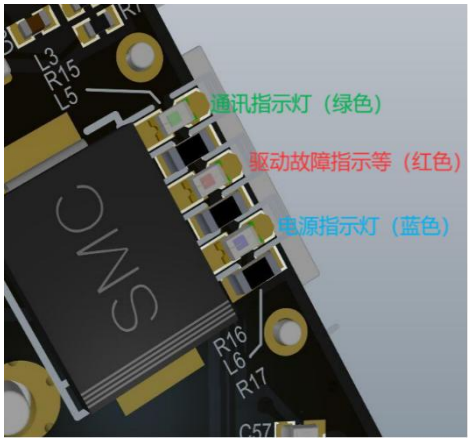
No.	Interface Function	Pin	Description	Color
1	**Serial communication**	1	Serial signal ground (GND)	Black
		2	Serial signal input (RX)	Yellow
		3	Serial signal output (TX)	Green
2	**Power supply input and CAN communication**	1	Positive pole（+）	Red
		2	Negative pole（-）	Black
		3	CAN communication high side（CAN_H）	White
		4	CAN communication low side（CAN_L）	Blue

E-mail:info@cubemars.com
https://www.cubemars.com/

1.2.3 Recommended Brands and Models for Driver Interface

No.	Onboard interface model	Brand	Terminal interface model	Brand
1	A1257WR-S-3P	CJT	A1257H-3P	CJT
2	XT30PW(2+2)-M	AMASS	XT30(2+2)-F	AMASS

1.3Driver Indicator Light Definitions



Indicator Light		
1.Power Indicator Light (Blue when lit)）	Light on	The driver board is powered
	Light off	The driver board is not powered
2.Operation Indicator Light (Green when lit)	Light on	The motor is working
	Light off	The motor is not working
3.Drive Fault Indicator Light (Red when lit)	Light on	Driver board fault
	Light off	Driver board funtion normally

⚠：After the driver board is powered, the blue light should remain on in the normal state, and the green and red lights should light up for 2 seconds before going out.

E-mail:info@cubemars.com
https://www.cubemars.com/

1.4 Main Accessories and Specifications

No.	Item	Specifications		Quantity	Remarks
1	Power and signal plug	Power and CAN cable	16AWG - Red and black silicone wire and white and blue - Teflon 30#-OD0.64-100±10mm-4-XT30(2+2)-F-one end connector XT30(2+2)-F,the other end stripped and tinned 3±1mm	1PCS Each	±2MM
2		Serial port cable	Teflon30# Wire OD0.64-200±10mm-3-GH1.25-3PINmale to FC crimp terminal horn plug 2*4PIN	1PCS Each	±2MM
3	Thermistor	MF51B103F3950-10K-3950		2PCS	
4	Electrolytic capacitor	120Uf-63V-10x12MM		2PCS	AK10-9 V3.0 standard
5	Power MOS	HYG035N08NS2C2-80V-2.8mΩ		12PCS	

E-mail:info@cubemars.com
https://www.cubemars.com/

2.Connection

2.1 R-link Appearance Introduction&Product Specifications and Interface Definition

Product Specifications	
Rated working voltage	5V
VCC voltage selection switch	3.3v (default)
Standby power cosumption	≤30mA
Size	73.8x23.6x14.5MM
Working environment temperature	-20℃ to 65℃
Maximum Allowable Temperature for Control Board	85℃

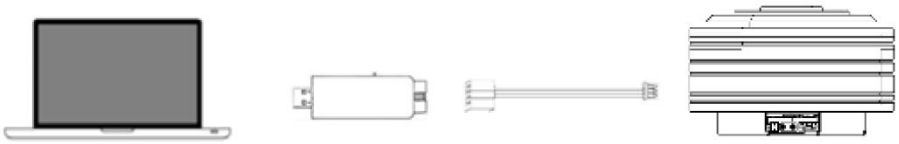
No	Interface Function	Pin	Description
1	Serial Interface	1	Serial signal imput（RXD）
		2	Serial signal output（TXD）
		3	Serial signal ground (GND)
		4	VCC power output（VCC）
		5	Data sent request（RTS）
		6	Clear to send（CTS）
		7	Power ground（GND）
		8	5V Power output（5V）

E-mail:info@cubemars.com
https://www.cubemars.com/

2.2 R-link Indicator Light Definitions

No.	Color	Description
1	Red	PWR Power Indicator, indicates the power status of the R-link. Under normal conditions, it will light up red when power is connected. If it does not light up when power is connected, please remove the power source immediately and do not attempt to power on again.
2	Red	TXD Serial Communication Output (TX), usually off, blinks when there is data output from the R-link serial port.
3	Red	RXD Serial Communication Input (RX), usually off, blinks when there is data input to the R-link serial port.

2.3 Driver and R-link Connection and Precautions

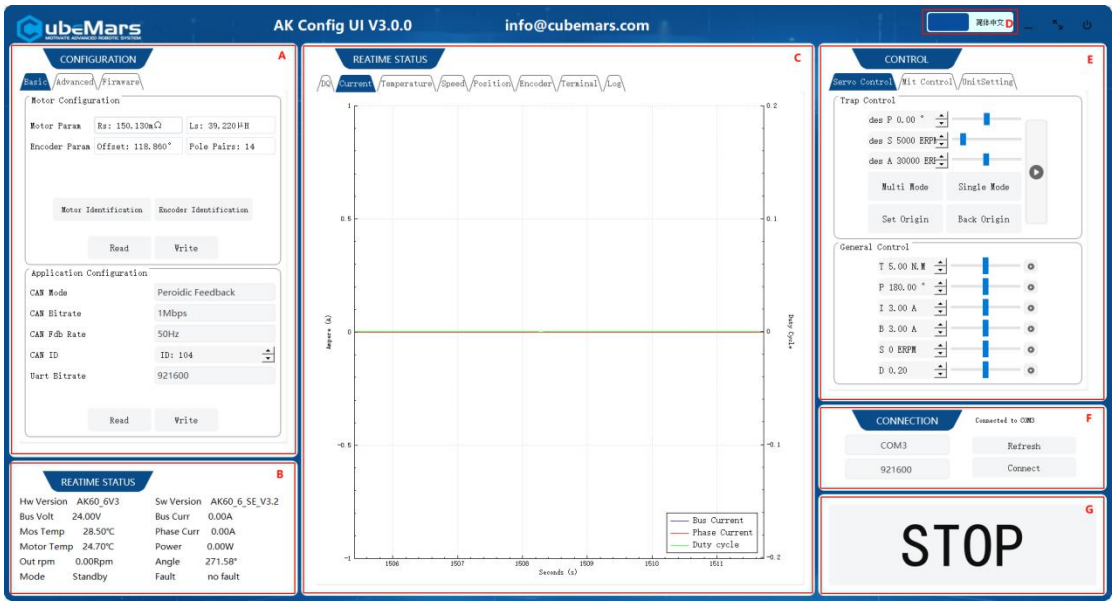


- USB on R-Link ---> PC End
- 8Pin Port ---> RLink 8Pin Port End
- 3Pin Terminal (UART Port) ---> 3Pin Port on Motor (UART)

E-mail:info@cubemars.com
https://www.cubemars.com/

3.Upper Computer Instruction

3.1Upper Computer Interface and Explanation



A. Configuration

B. Real-time Status

C. Real-time Data

D. Chinese-English Switch

E. Control

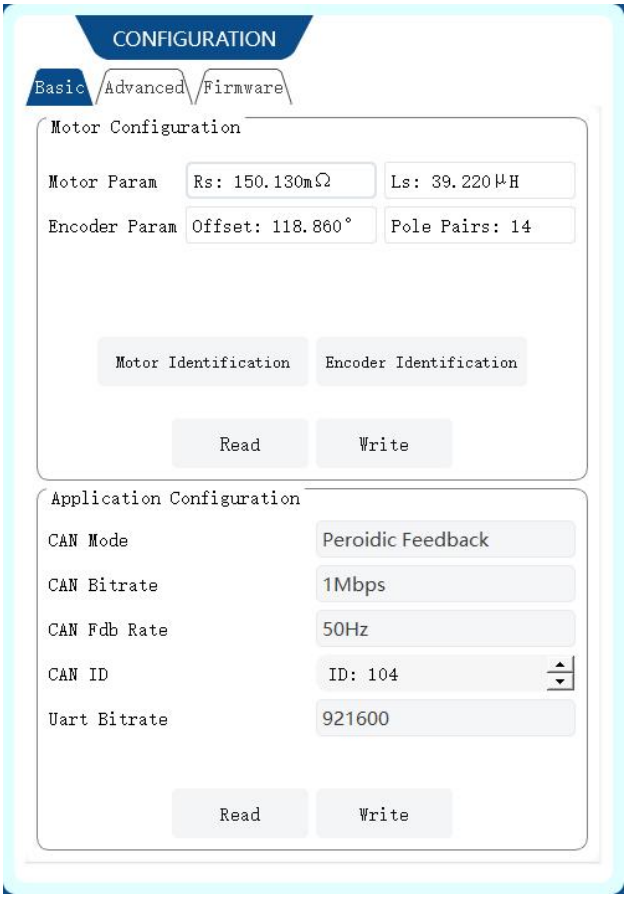
F. Connection

G. Stop

E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.1 Configuration

3.1.1.1 Basic Settings



The Basic Settings interface allows users to configure motor parameters and communication parameters; The Motor Settings section supports motor parameter identification, encoder parameter calibration, and parameter reading and writing. For specific motor calibration operations, see section 3.2. The Application Settings section supports settings for CAN port mode, CAN bus rate, CAN feedback rate, CAN ID, and serial port baud rate, which users can match according to actual conditions.

The CAN port mode is divided into two types: periodic feedback and query-reply. Periodic reporting will provide messages at regular intervals based on the CAN feedback rate; the query-reply mode only sends back message reports when the motor receives the correct control commands.

`Motor Identification`: Perform motor parameter identification, which can be observed in the position curve to see if the motor is moving. After identification, it will automatically stop and update the motor parameters to the upper computer.

E-mail:info@cubemars.com
https://www.cubemars.com/

⚠： The entire process of motor parameter identification should be kept under no-load conditions; otherwise, it may lead to inaccurate identification parameters or motor damage. The identification process will start with a sharp, short sound, then the motor starting to rotate with a louder noise.

`Encoder Identification`：Perform encoder parameter identification, which can be observed in the position curve to see if the motor is moving. After identification, it will automatically stop and update the encoder parameters to the upper computer.

⚠： The entire process of motor parameter identification should be kept under no-load conditions; otherwise, it may lead to inaccurate identification parameters or motor damage. The identification process will start with a sharp, short sound. Performing this process consecutively multiple times can cause the motor temperature to rise sharply.

`Read`：Read the motor parameters and encoder parameters from the driver board to the upper computer.

`Write`：Write the motor parameters and encoder parameters from the upper computer to the driver board.

3.1.1.2 Advanced Settings

⚠： Warning. Note that customers who are using this product for the first time should not arbitrarily modify the parameters in the advanced settings, otherwise, the motor may run abnormally.

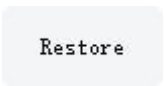
E-mail:info@cubemars.com
https://www.cubemars.com/

The Advanced Settings interface allows users to customize more parameters according to their needs. In addition to the parameter reading and writing functions, it also provides parameter restoration, loading, and exporting functions.

Read Parameters: Click to read the parameters from the driver board to the upper computer.

Click to write the current upper computer parameters to the driver board.

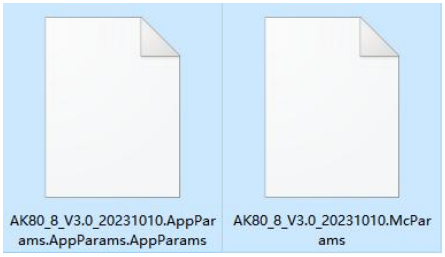
⚠: Before rewriting motor parameters, be sure to read the parameters first; otherwise, other motor parameters will be incorrect. If this happens, please download the APP parameters for the corresponding motor from the official website, and then write the default parameters of the motor through "Load from file"



Click to restore the upper computer parameters to the default values for the corresponding model.

E-mail:info@cubemars.com
https://www.cubemars.com/

`Load from file` Load the parameter file into the upper computer; click and select the parameter file for the corresponding motor model, which comes in two formats: .AppParams and .McParams.

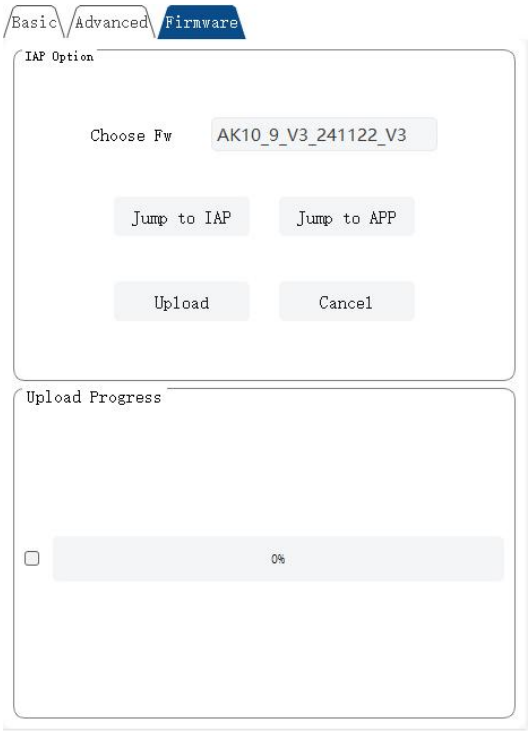


`Save to file` Export parameters： Click to export parameters, select the save file, and you will obtain a custom parameter file.

⚠: Please strictly use the specified voltage, current, power, and temperature. Our company will not assume any legal responsibility for personal injury or irreversible damage to the driver board and motor caused by the improper operation of this product.

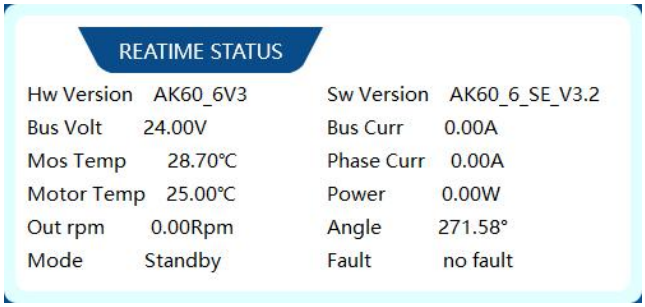
E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.1.3 Firmware Upgrade



The Firmware Upgrade interface includes three sections: Firmware Information, Upgrade Options, and Upgrade Progress. Customers can upgrade firmware based on their own needs. For specific steps on upgrading the firmware, see section 3.4.

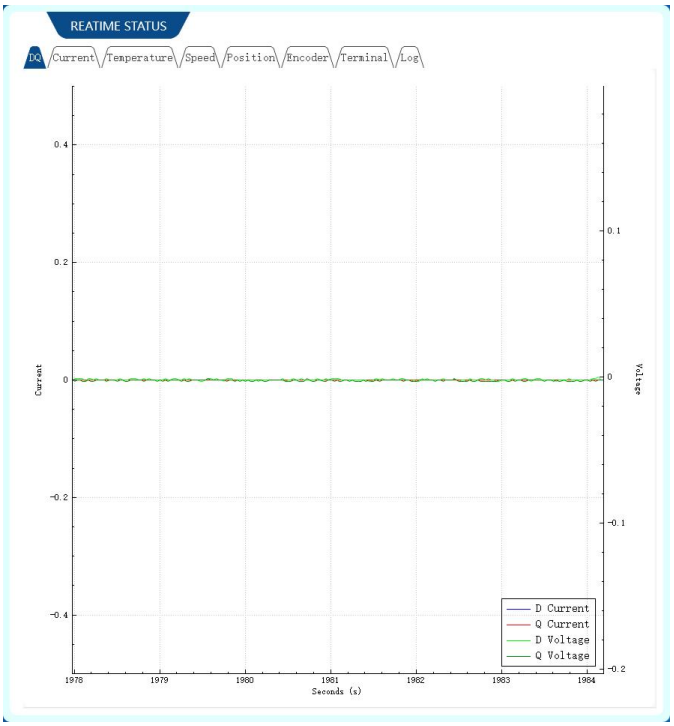
3.1.2 Real-time Status



The Real-time Status interface displays the current software and hardware versions, motor firmware version, motor parameter status, operating mode, and fault status.

E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.3 Real-time Data



This page supports viewing real-time data feedback and debugging interface, as well as plotting graphs. The data includes:

-DQ: DQ voltage represents the motor voltage converted onto the D-axis and Q-axis, and DQ current represents the motor current converted onto the D-axis and Q-axis.

- Current: Current in indicates the motor's input current, Current motor indicates the motor's phase line current, and Duty cycle indicates the motor's duty ratio.
- Temperature: Temperature MOSFET indicates the temperature of the motor's MOS transistor, and Temperature Motor indicates the motor's temperature.
- Speed: ERPM indicates the electrical rotational speed of the motor's rotor.
- Position: Position indicates the position of the motor's rotor.
- Encoder Angle: Encoder angle indicates the angle of the encoder.

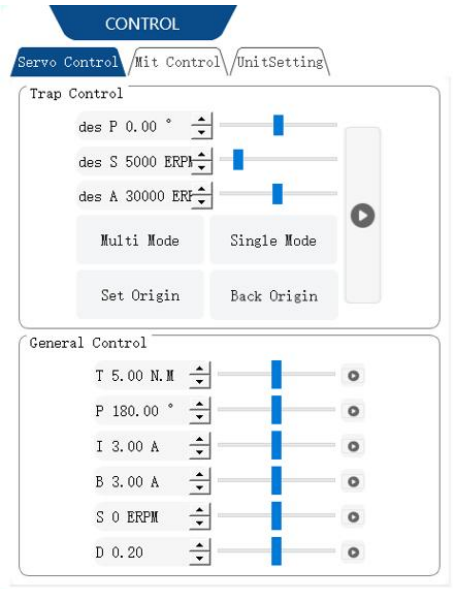
3.1.4 Chinese-English Switch

The upper computer currently supports switching between Chinese and English languages. Click on "中" or "EN" in the red box in the upper right corner of the interface to switch languages.

E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.5 Control

3.1.5.1 Servo Control



Multi Mode：In the position-velocity loop mode, the motor can be set to positions ranging from -36000° to 36000°;

Single Mode： In the position-speed loop mode, the motor can be set to positions ranging from 0 to 359°;

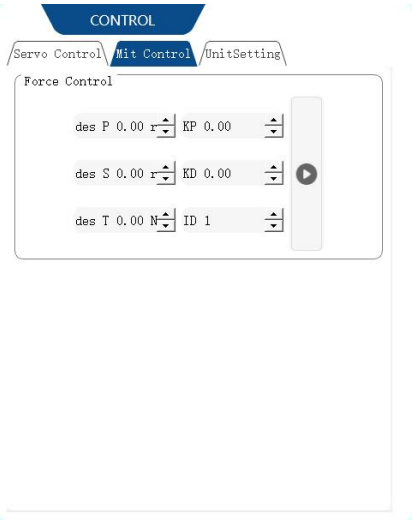
Set Origin：The motor will set the current rotor position as the zero position;

Back Origin：The motor rotates to the zero position. (The motor will operate at the maximum speed, pay attention to safety)

 ：The corresponding mode start button.

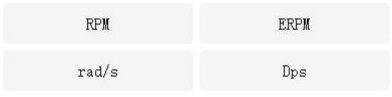
E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.5.2 Force Control (Mit) Control



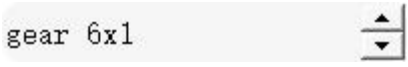
Force control mode (motion control mode, MIT MODE) offers three different control modes: position, speed, and torque loops, which are determined by the data sent. For specific operation methods, see section 3.3.2.

3.1.5.3 Unit Settings



:Each button corresponds to the unit that needs to be converted, supporting the conversion display of rotational speed (RPM, ERPM, rad/s, Dps).

:Enter the number of motor poles, as shown in the left figure, 14 is the number of pole pairs.



:Enter the motor reduction ratio.

E-mail:info@cubemars.com
https://www.cubemars.com/

3.1.6 Connection

The Connection module can configure serial communication, as shown in the left figure. First, click "Refresh", then select the communication baud rate, choose the correct port, and finally "Connect" of the motor. The display changes from "Not connected" to "Connected to COMX" indicating a successful serial connection. As shown in the right figure, click "Disconnect" to disconnect the communication.

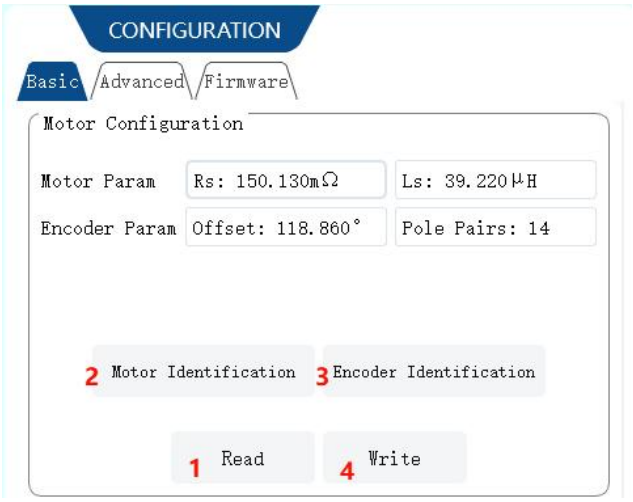
3.1.7 Stop

The experimental stop button, when clicked, will stop the motor's working.

3.2 Driver Board Calibration

When you have reinstalled the driver board on the motor, changed the wiring sequence of the motor's three-phase lines, or updated the firmware, recalibration is necessary (it was calibrated when it left the factory for the first time, no need to calibrate again). After calibration, the motor can be used.

3.2.1 Calibration Steps



E-mail:info@cubemars.com
https://www.cubemars.com/

STEP0:

Ensure that the motor's power supply is stable, the connectors are properly connected, and successfully connected to the upper computer, then enter the system settings page.

STEP1:

Click "Read" and wait for the connection interface to display.

;

STEP2:

Click "Motor Identification", after a brief beep 

; the motor will start rotating. Wait for about 10 seconds before the motor stops rotating. 

When the display appears, it indicates that the motor parameter identification process is complete.

STEP3:

Click "Encoder Identification", the motor will rotate slowly. Wait for about 45 seconds, and when the connection interface display is complete 

the encoder parameter identification is finished.

STEP4:

Click "Write", and when the connection interface display is complete 

the calibration is finished.

⚠：The entire process of motor parameter and encoder parameter identification should be kept under no-load conditions; otherwise, it may lead to inaccurate identification parameters or motor damage. The encoder parameter identification process generates heat, and performing this process consecutively multiple times can cause the motor temperature to rise sharply.

3.3 Control Demonstration

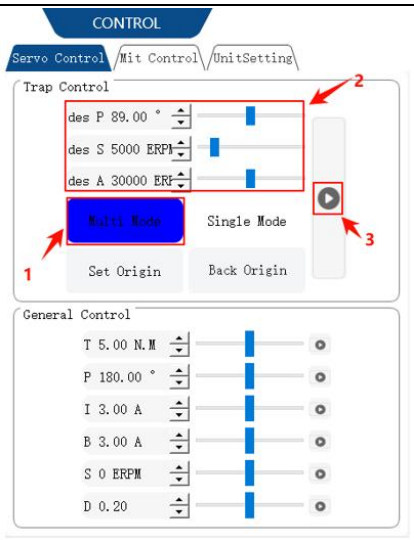
After confirming that the motor power supply and load are correctly installed, the wiring is correct, and the driver board calibration is complete, you can start using the motor. Below are explanations for the use of servo control and force control modes.

3.3.1 Servo Mode

3.3.1.1 Multi-Position-Velocity Loop Mode

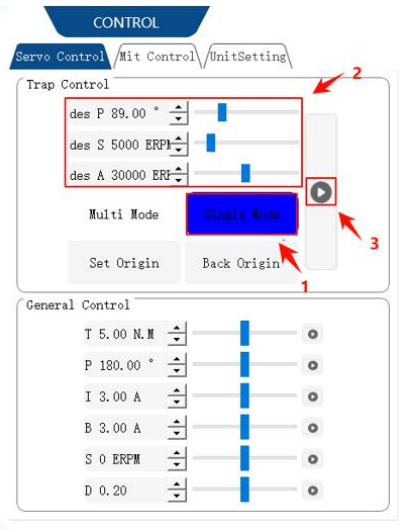
Ensure that the motor input power is stable, the connectors are properly connected, and successfully connected to the upper computer. In the "Servo Control" interface, click on "Multi Mode," enter the desired position (at this time, the position is ±100 turns, i.e., -36000° -36000°), desired speed, and acceleration. The motor will move at the desired speed until it reaches the desired position.

E-mail:info@cubemars.com
https://www.cubemars.com/



3.3.1.2 Single-Position-Velocity-Loop Mode Mode

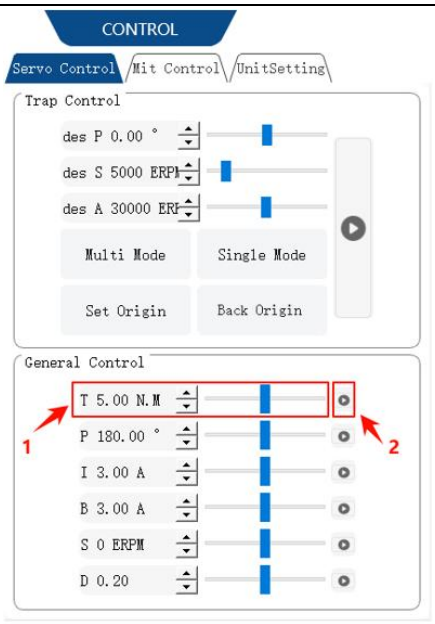
Ensure that the motor input power is stable, the connectors are properly connected, and successfully connected to the upper computer. In the "Servo Control" interface, click on "Single Mode," enter the desired position (at this time, the position is only one turn, i.e., 0° -359°), desired speed, and acceleration. The motor will move at the desired speed until it reaches the desired position.



3.3.1.3 Braking Mode

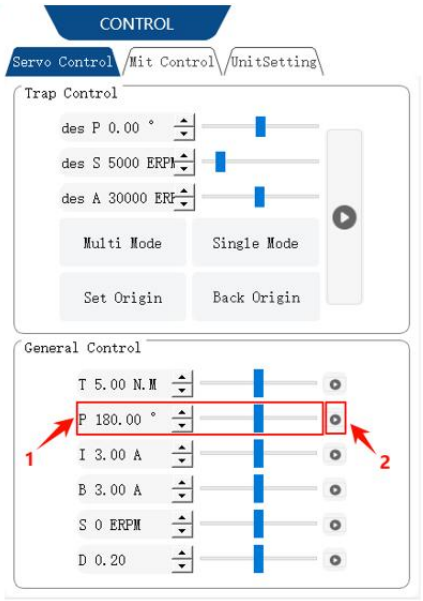
Ensure that the motor input power is stable, the connectors are properly connected, and successfully connected to the upper computer. In the "Servo Control" interface, enter the desired torque T, and the motor will brake with the desired torque.

E-mail:info@cubemars.com
https://www.cubemars.com/



3.3.1.4 Position Loop Mode

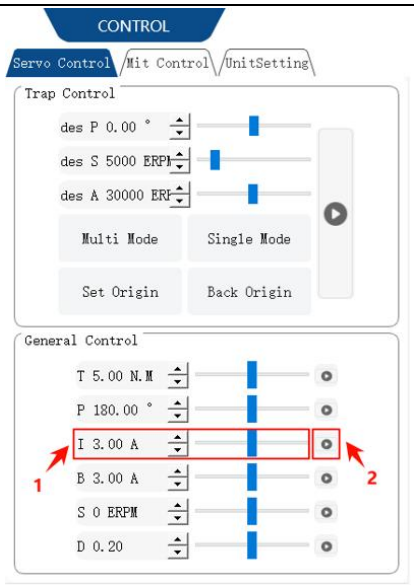
Ensure that the motor input power is stable, the connectors are properly connected, and successfully connected to the upper computer. In the "Servo Control" interface, enter the desired position P, and the motor will reach the desired position with maximum speed and acceleration.



3.3.1.5 Current Loop Mode

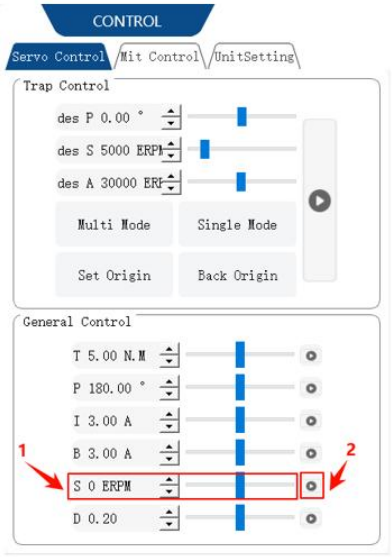
Ensure that the motor input power is stable, the connectors are properly connected, and successfully connected to the upper computer. In the "Servo Control" interface, enter the desired current I, and the motor will operate at the desired current.

E-mail:info@cubemars.com
https://www.cubemars.com/



3.3.1.6 Velocity Loop Mode

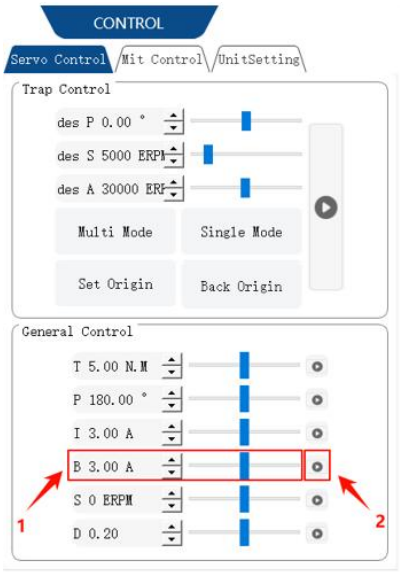
Ensure that the motor input power is stable, the connectors are properly connected, and the motor is in servo mode. After successfully connecting to the upper computer, in the "Servo Control" interface, enter the desired speed S (±50000 ERPM), and the motor will operate at the desired speed (with the default maximum acceleration).



3.3.1.7 Braking Loop Mode

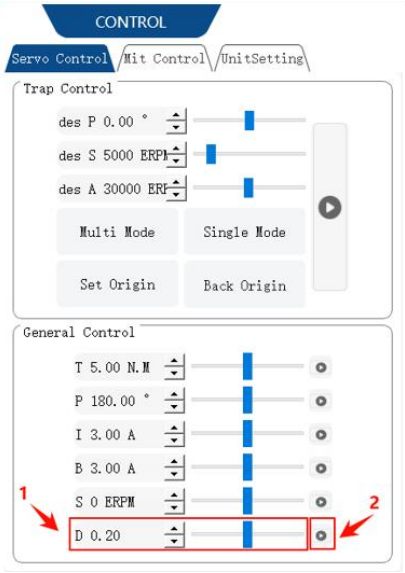
Ensure that the motor input power is stable, the connectors are properly connected, and successfully connected to the upper computer. In the "Servo Control" interface, enter the desired braking current B, and the motor will brake with the desired current.

E-mail:info@cubemars.com
https://www.cubemars.com/



3.3.1.8 Duty Cycle Mode

Ensure that the motor input power is stable, the connectors are properly connected, and the motor is in servo mode. After successfully connecting to the upper computer, in the "Servo Control" interface, enter the desired duty cycle (default 0.005-0.95), and the motor will operate at the desired duty cycle.

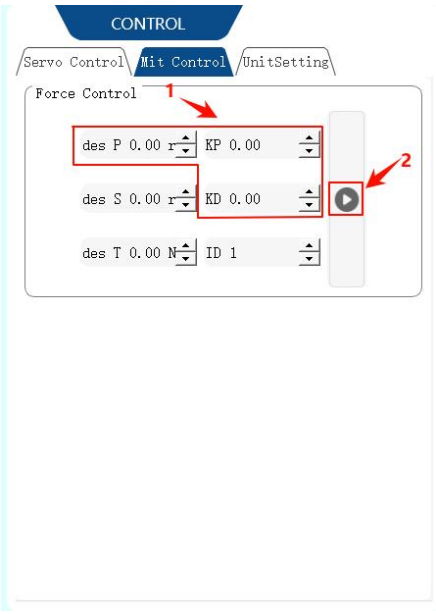


3.3.2 Force Control Mode(Mit Control)

3.3.2.1Force Control Position Mode

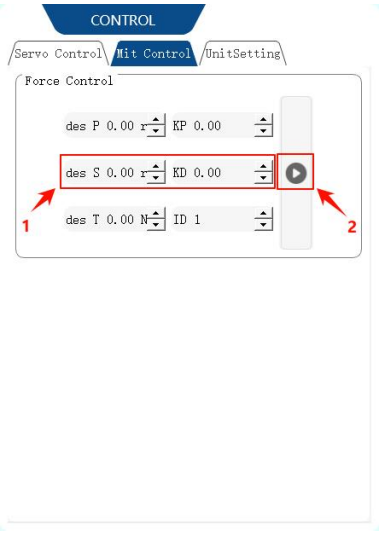
Ensure that the motor input power is stable, the connectors are properly connected, and successfully connected to the upper computer. In the "MIT Control" interface, enter the corresponding "CAN ID," desired position, and KP, KD, then click "Start," and the motor will move to the desired position. (The motor operates with a specified position and Kp, Kd values, turning

towards the specified position)



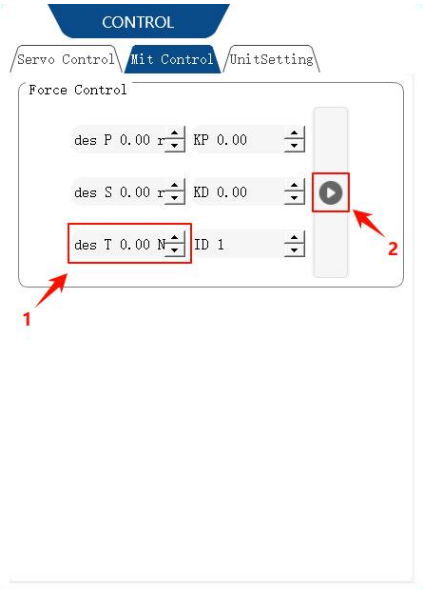
3.3.2.2 Force Control Velocity Mode

Ensure that the motor input power is stable, the connectors are properly connected, enter the desired speed S and KD, then click "Start," and the motor will operate at the specified speed. (The motor operates with a specified speed and Kd value, rotating at the specified speed)

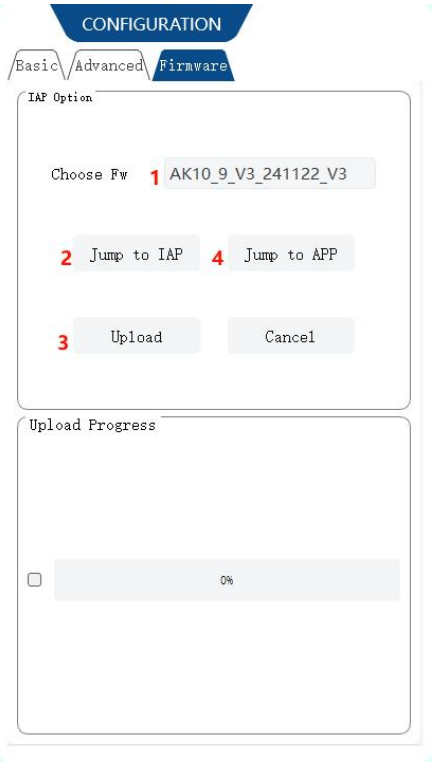


3.3.2.3 Force Control Torque Mode

Ensure that the motor input power is stable, the connectors are properly connected, enter the desired torque T, then click "Start," and the motor will operate at the specified torque. (The motor operates with a specified torque value, rotating at the specified torque value)



3.4Firmware Update



1. Select the corresponding firmware from the drop-down list.
2. Click "Jump to IAP."
3. Click "Upload" and wait for the upgrade progress bar to reach 100%.
4. Click "Jump to App," wait for 5 seconds, the motor enters the operating mode, and the firmware update is complete.

E-mail:info@cubemars.com
https://www.cubemars.com/

4. Driver Board Communication Protocol and Description

4.1Servo Mode Control Modes and Description

There are 6 control modes in the servo mode, each corresponding to a specific ID number.

Duty Cycle Mode: A specified duty cycle voltage is given to the motor, similar to square wave drive form;
Current Loop Mode: A specified Iq current is given to the motor, and since the motor output torque = iq*KT, it can be used as a torque loop (KT data can be referenced on the official website);
Current Brake Mode: A specified braking current is given to the motor to hold it in the current position (pay attention to motor temperature when using);
Velocity Mode: A specified operating speed is given to the motor (acceleration is default to the maximum value);
Position Mode: A specified position is given to the motor, and the motor will move to the specified position (speed and acceleration are default to the maximum value);
Position-Velocity Loop Mode: A specified position, speed, and acceleration are given to the motor.

Servo mode driver board data transmission definition

Identifier: Control Mode ID + Driver ID　　Frame Type: Extended Frame
Frame Format: DATA　　DLC: 8 bytes
The servo motor protocol is a CAN protocol, using an extended frame format as shown below:

Can ID bits	[28]-[8]	[7]-[0]
Field name (Function Definition)	Control mode ID （Control ModeID）	Drive ID （Driver ID）

Control mode has{0,1,2,3,4,5,6}　7 characteristic values corresponding to 7 control modes.

Duty Cycle Mode: 0
Current Loop Mode: 1
Current Brake Mode: 2
Velocity Mode: 3
Position Mode: 4
Set Origin Mode: 5
Position-Speed Loop Mode: 6

The following provides examples of controlling the motor in various modes.

E-mail:info@cubemars.com
https://www.cubemars.com/

Below are library call, function and macro definitions for various instances.

```
typedef enum {
	CAN_PACKET_SET_DUTY = 0,		// Duty Cycle Mode
	CAN_PACKET_SET_CURRENT,		// Current Loop Mode
	CAN_PACKET_SET_CURRENT_BRAKE,	//  Current Brake Mode
	CAN_PACKET_SET_RPM,		//  Speed Mode
	CAN_PACKET_SET_POS,		//  Position Mode
	CAN_PACKET_SET_ORIGIN_HERE,	// Set origin position mode (zero mode)
	CAN_PACKET_SET_POS_SPD,		// Position-Velocity Loop Mode
} CAN_PACKET_ID;

void comm_can_transmit_eid(uint32_t id, const uint8_t *data, uint8_t len) {
	uint8_t i=0;
	if (len > 8) {
		len = 8;
	}
	CanTxMsg TxMessage;
	TxMessage.StdId = 0;
	TxMessage.IDE = CAN_ID_EXT;
	TxMessage.ExtId = id;
	TxMessage.RTR = CAN_RTR_DATA;
	TxMessage.DLC = len;
	for(i=0;i<len;i++)
		TxMessage.Data[i]=data[i];
	CAN_Transmit(CHASSIS_CAN, &TxMessage);	//CAN port trasmissionTxMessage Data Can port sends TxMesage Data
}


void buffer_append_int32(uint8_t* buffer, int32_t number, int32_t *index) {
	buffer[(*index)++] = number >> 24;
	buffer[(*index)++] = number >> 16;
	buffer[(*index)++] = number >> 8;
	buffer[(*index)++] = number;
}

void buffer_append_int16(uint8_t* buffer, int16_t number, int16_t *index) {
	buffer[(*index)++] = number >> 8;
	buffer[(*index)++] = number;
}
```

4.1.1 Duty Cycle Mode

Duty Cycle Mode Data Transmission Definition

E-mail:info@cubemars.com
https://www.cubemars.com/

Data Bit	Data[0]	Data[1]	Data[2]	Data[3]
Range	0~0xff	0~0xff	0~0xff	0~0xff
Corresponding Variable	Duty Cycle Bit 25-32	Duty Cycle Bit 17-24	Duty Cycle Bit 9-16	Duty Cycle Bit 1-8

```
void comm_can_set_duty(uint8_t controller_id, float duty) {
    int32_t send_index = 0;
    uint8_t buffer[4];
    buffer_append_int32(buffer, (int32_t)(duty * 100000.0), &send_index);
    comm_can_transmit_eid(controller_id |
                ((uint32_t)CAN_PACKET_SET_DUTY << 8), buffer, send_index);
}
```

4.1.2 Current Loop Mode

Current Loop Mode Data Transmission Definition

Data Bit	Data[0]	Data[1]	Data[2]	Data[3]
Range	0~0xff	0~0xff	0~0xff	0~0xff
Corresponding Variable	Current Bit 25-32	Current Bit 17-24	Current Bit 9-16	Current Bit 1-8

The current value is of int32 type, and the values -60000 to 60000 represent -60 to 60 A.

Current Loop Mode Send Example Routine

```
void comm_can_set_current(uint8_t controller_id, float current) {
    int32_t send_index = 0;
    uint8_t buffer[4];
    buffer_append_int32(buffer, (int32_t)(current * 1000.0), &send_index);
    comm_can_transmit_eid(controller_id |
                ((uint32_t)CAN_PACKET_SET_CURRENT << 8), buffer, send_index);
}
```

4.1.3 Current Brake Mode

Current Brake Mode Data Transmission Definition

Data Bit	Data[0]	Data[1]	Data[2]	Data[3]
Range	0~0xff	0~0xff	0~0xff	0~0xff
Corresponding Variable	Brake Current Bit 25-32	Brake Current Bit 17-24	Brake Current Bit 9-16	Brake Current Bit 1-8

E-mail:info@cubemars.com
https://www.cubemars.com/

The brake current value is of int32 type, and the values 0 to 60000 represent 0 to 60 A.

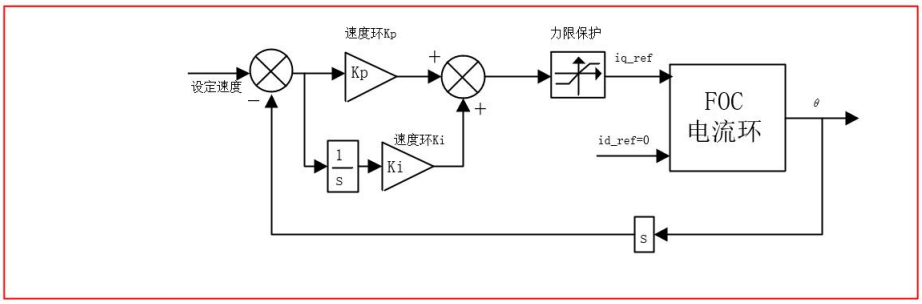
Current Brake Mode Send Example Routine

```
void comm_can_set_cb(uint8_t controller_id, float current) {
	int32_t send_index = 0;
	uint8_t buffer[4];
	buffer_append_int32(buffer, (int32_t)(current * 1000.0), &send_index);
	comm_can_transmit_eid(controller_id |
		((uint32_t)CAN_PACKET_SET_CURRENT_BRAKE << 8), buffer, send_index);
}
```


E-mail:info@cubemars.com
https://www.cubemars.com/

4.1.4 Velocity Loop Mode

Simplified Control Diagram for Velocity Loop



Velocity Loop Mode Data Transmission Definition

Data Bit	Data[0]	Data[1]	Data[2]	Data[3]
Range	0~0xff	0~0xff	0~0xff	0~0xff
Corresponding Variable	Speed Bit 25-32	Speed Bit 17-24	Speed Bit 9-16	Speed Bit 1-8

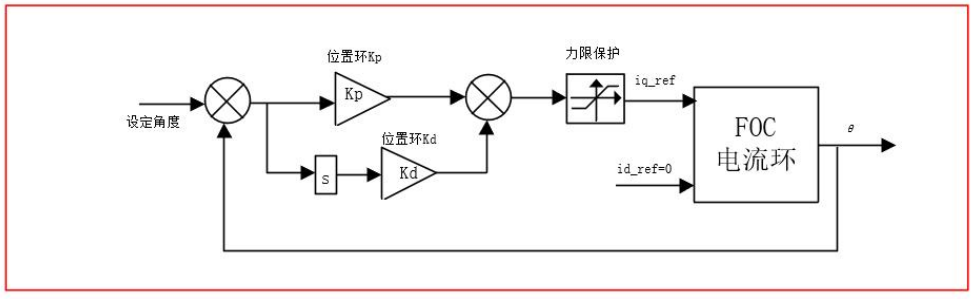
The speed value is of int32 type, and the range -100000 to 100000 represents -100000 to 100000 electrical RPM.

Velocity Loop Send Example Routine

```
void comm_can_set_rpm(uint8_t controller_id, float rpm) {
    int32_t send_index = 0;
    uint8_t buffer[4];
    buffer_append_int32(buffer, (int32_t)rpm, &send_index);
    comm_can_transmit_eid(controller_id |
                ((uint32_t)CAN_PACKET_SET_RPM << 8), buffer, send_index);
}
```

4.1.5 Position Loop Mode

Simplified Control Diagram for Position Loop



Position Loop Mode Data Transmission Definition

Data Bit	Data[0]	Data[1]	Data[2]	Data[3]
Range	0~0xff	0~0xff	0~0xff	0~0xff
Corresponding variable	Position Bit 25-32	Position Bit 17-24	Position Bit 9-16	Position Bit 1-8

E-mail:info@cubemars.com
https://www.cubemars.com/

The position value is of int32 type, and the range -360000000 to 360000000 represents positions from -36000° to 36000°.
Position Loop Send Example Routine

```
void comm_can_set_pos(uint8_t controller_id, float pos) {
    int32_t send_index = 0;
    uint8_t buffer[4];
    buffer_append_int32(buffer, (int32_t)(pos * 10000.0), &send_index);
    comm_can_transmit_eid(controller_id |
            ((uint32_t)CAN_PACKET_SET_POS << 8), buffer, send_index);
}
```

4.1.6 Setting Origin Mode

Data Bit	Data[0]
Range	0~0x01
Corresponding Variable	Set Command

The set command is of uint8_t type, 0 represents setting a temporary origin (erased upon power loss), and 1 represents setting a permanent origin (parameters are automatically saved).

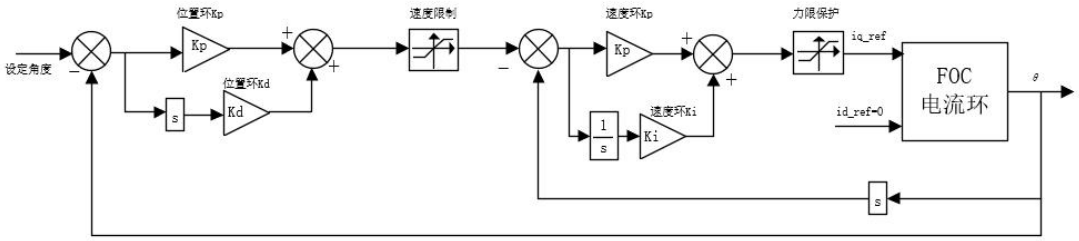
Position Loop Send Example Routine

```
void comm_can_set_origin(uint8_t controller_id, uint8_t set_origin_mode) {
    int32_t send_index = 1;
    uint8_t buffer;
    buffer=set_origin_mode;
    comm_can_transmit_eid(controller_id |
            ((uint32_t) CAN_PACKET_SET_ORIGIN_HERE << 8), buffer, send_index);
}
```


E-mail:info@cubemars.com
https://www.cubemars.com/

4.1.7 Position-Velocity Loop Mode

Simplified Diagram for Position-Speed Loop



Position-Velocity Loop Mode Data Transmission Definition

Data Bit	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
Range	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff
Corresponding Variable	Position 25-32	Position 17-24	Position 9-16	Position 1-8	Speed High Byte	Speed Low Byte	Acceleration High Byte	Acceleration Low Byte

The position value is of int32 type, and the range -360000000 to 360000000 corresponds to positions from -36000° to 36000

Tthe speed value is of int16 type, and the range -32768 to 32767 corresponds to -327680 to 327680 electrical RPM;

The acceleration value is of int16 type, and the range 0 to 32767 corresponds to 0 to 327670, with 1 unit equal to 10 electrical RPM/s^2 .

```
void comm_can_set_pos_spd(uint8_t controller_id, float pos,int16_t spd, int16_t RPA ) {
    int32_t send_index = 0;
    Int16_t send_index1 = 4;
    uint8_t buffer[8];
    buffer_append_int32(buffer, (int32_t)(pos * 10000.0), &send_index);
    buffer_append_int16(buffer,spd/10.0, & send_index1);
    buffer_append_int16(buffer,RPA/10.0, & send_index1);
    comm_can_transmit_eid(controller_id |
            ((uint32_t)CAN_PACKET_SET_POS_SPD << 8), buffer, send_index1);
}
```

4.2 Force Control Mode (MIT) Communication Protocol

The Force Control Mode has three control modes, all sharing a single Control ID number, with the specific control mode determined by the transmitted data. For detailed operation of each mode on the upper computer, see section 3.3.2.

E-mail:info@cubemars.com
https://www.cubemars.com/

Position Loop Mode: Operates the motor with a specified position and Kp, Kd values, the motor is going to rotate to the specified position.
Velocity Loop Mode: Operates the motor with a specified speed and Kd value, the motor is going to rotate at the specified speed.
Current Loop Mode: Operates the motor with a specified torque value, the motor is going to rotate at the specified torque value.

The Force Control Mode uses the CAN protocol, adopting an extended frame format as shown below

Can ID bits	[28]-[8]	[7]-[0]
Field name	Control mode ID	Drive ID

The Control Mode ID for the Force Control Mode has a characteristic value of 8

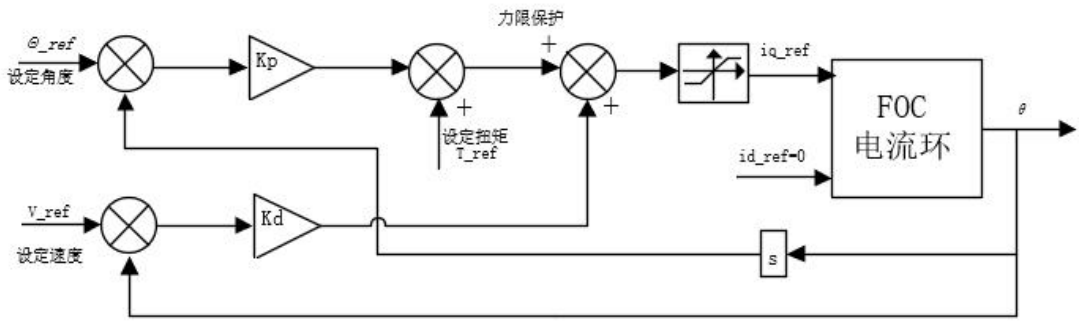
Force Control Mode Driver Board Data Transmission Definition

Identifier: Control Mode ID + Driver ID Frame Type: Extended Frame
Frame Format: DATA DLC: 8 bytes

Data Field	DATA[0]	DATA[1]		DATA[2]	DATA[3]
Data Bits	7-0	7-4	3-0	7-0	7-0
Data Content	High 8 bits of KP value	Low 4 bits of KP value	High 4 bits of KD value	Low 8 bits of KD value	High 8 bits of Position value

Data Field	DATA[4]	DATA[5]	DATA[6]		DATA[7]
Data Bits	7-0	7-0	7-4	3-0	0-7
Data Content	Low 8 bits of Position value	High 8 bits of Speed value	High 8 bits of Speed value	High 4 bits of Current value	Low 8 bits of Current value

Simplified Diagram for Force Control Mode



E-mail:info@cubemars.com
https://www.cubemars.com/

Parameter Ranges

Module Motor	AK10-9	AK60-6	AK70-9
KV	60	80	60
Motor position (rad)	-12.56f-12.56f		
Motor speed （rad/s）	-28.0f-28.0f	-60.0f-60.0f	-30.0f-30.0f
Motor torque （N.M）	-54.0f-54.0f	-12.0f-12.0f	-32.0f-32.0f
Kp range	0-500		
Kd range	0-5		

Force Control Mode Send Code Example

Send example code （parameters are based on AK10-9）

```
u8 SERVO_Can_Send_Msg(uint32_t ExtId,u8* msg,u8 len)
{
        u8 mbox;
        u16 i=0;
//          TxMessage.StdId=stdId;          // Standard indentifier
        TxMessage.ExtId=ExtId;          //  Set Extended Identifier
        TxMessage.IDE=CAN_Id_Extended;  // Standard frame
        TxMessage.RTR=CAN_RTR_Data;     // Data frame
        TxMessage.DLC=len;              //  Length of data to send
        for(i=0;i<len;i++)
        TxMessage.Data[i]=msg[i];
        mbox= CAN_Transmit(CANx, &TxMessage);
        i=0;
        while((CAN_TransmitStatus(CAN1,  mbox)==CAN_TxStatus_Failed)&&(i<0XFFF))i++;
    //Wait for transmission to complete
        if(i>=0XFFF)return 1;
        return 0;
}

float fmaxf(float x, float y){
    /// Returns maximum of x, y ///
    return (((x)>(y))?(x):(y));
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
    }

float fminf(float x, float y){
    /// Returns minimum of x, y ///
    return (((x)<(y))?(x):(y));
    }

float fmaxf3(float x, float y, float z){
    /// Returns maximum of x, y, z ///
    return (x > y ? (x > z ? x : z) : (y > z ? y : z));
    }

float fminf3(float x, float y, float z){
    /// Returns minimum of x, y, z ///
    return (x < y ? (x < z ? x : z) : (y < z ? y : z));
    }


float P_MIN =-12.56f;
float P_MAX =12.56f;
float V_MIN =-33.0f;
float V_MAX =33.0f;
float T_MIN =-65.0f;
float T_MAX =65.0f;
float Kp_MIN =0;
float Kp_MAX =500.0f;
float Kd_MIN =0;
float Kd_MAX =5.0f;
float Test_Pos=0.0f;


int p_int ;
int v_int ;
int kp_int ;
int kd_int ;
int t_int ;


void pack_cmd(uint8_t controller_id, float p_des, float v_des, float kp, float kd, float t_ff){
uint8_t buffer[8];

p_des = fminf(fmaxf(P_MIN, 0), P_MAX);
v_des = fminf(fmaxf(V_MIN, 0), V_MAX);
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
kp = fminf(fmaxf(Kp_MIN, kp), Kp_MAX);
kd = fminf(fmaxf(Kd_MIN, kd), Kd_MAX);
t_ff = fminf(fmaxf(T_MIN, t_ff), T_MAX);
/// convert floats to unsigned ints ///

  p_int = float_to_uint(p_des, P_MIN, P_MAX, 16);
  v_int = float_to_uint(v_des, V_MIN, V_MAX, 12);
  kp_int = float_to_uint(kp, Kp_MIN, Kp_MAX, 12);
  kd_int = float_to_uint(kd, Kd_MIN, Kd_MAX, 12);
  t_int = float_to_uint(t_ff, T_MIN, T_MAX, 12);

/// pack ints into the can buffer ///
buffer[0] = kp_int>>4;            //KP high byte
buffer[1] = ((kp_int&0xF)<<4)|( kd_int>>8);  // KP low 4 bits    KD high 4 bits
buffer[2] = kd_int&0xFF;        //Kd low byte
buffer[3] = p_int>>8;                // position high byte
buffer[4] = p_int&0xFF;  //position low byte
buffer[5] = v_int>>4;       // speed high byte
buffer[6] = ((v_int&0xF)<<4)|(t_int>>8);       //speed low 4 bits, torque high 4 bits
buffer[7] = t_int&0xff;     //torque low 8 bits
SERVO_Can_Send_Msg(controller_id |((uint32_t) CAN_PACKET_SET_mit << 8), buffer, 8);
        }
```

When sending packets, all numerical values must be converted to integer numbers using the following function before being transmitted to the motor.

```
        int float_to_uint(float x, float x_min, float x_max, unsigned int bits){
            /// Converts a float to an unsigned int, given range and number of bits ///
            float span = x_max - x_min;
            if(x < x_min) x = x_min;
            else if(x > x_max) x = x_max;
            return (int) ((x- x_min)*((float)((1<<bits)/span)));
        }
```


E-mail:info@cubemars.com
https://www.cubemars.com/

4.3 Motor Message Format

4.3.1 CAN Upload Message Protocol

The motor CAN message uses a timed upload mode, with an upload frequency that can be set from 1 to 500 Hz, and the upload byte is 8 bytes.

Data Bit	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
Range	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff	0~0xff
Corresponding Variable	High Byte of Position	Low Byte of Position	High Byte of Speed	Low Byte of Speed	High Byte of Current	Low Byte of Current	Motor Temperature	Error Code

The position is of int16 type, with a range of -32000 to 32000 representing a position of -3200° to 3200° ;
The speed is of int16 type, with a range of -32000 to 32000 representing an electrical speed of -320000 to 320000 rpm;
The current is of int16 type, with values -6000 to 6000 representing -60 to 60 A;
The temperature is of int8 type, with a range of -20 to 127 representing the driver board temperature of -20° C to 127° C;
The error code is of uint8 type, with 0 indicating no fault, 1 indicating motor over-temperature fault, 2 indicating over-current fault, 3 indicating over-voltage fault, 4 indicating under-voltage fault, 5 indicating encoder fault, 6 indicating MOSFET over-temperature fault, and 7 indicating motor lock-up.

Below is an example of message reception

```
void motor_receive(float* motor_pos,float*
motor_spd,float* cur,int_8* temp,int_8* error,rx_message)
  {
      int16_t pos_int = (rx_message)->Data[0] << 8 | (rx_message)->Data[1]);
      int16_t spd_int = (rx_message)->Data[2] << 8 | (rx_message)->Data[3]);
      int16_t cur_int = (rx_message)->Data[4] << 8 | (rx_message)->Data[5]);
    &motor_pos= (float)( pos_int * 0.1f); //Motor position
    &motor_spd= (float)( spd_int * 10.0f);//Motor speed
    &motor_cur= (float) ( cur_int * 0.01f);//Motor current
    &motor_temp= (rx_message)->Data[6] ;//Motor temperature
    &motor_error= (rx_message)->Data[7] ;//Motor error code
  }
```


E-mail:info@cubemars.com
https://www.cubemars.com/

4.3.2 Serial Port Message Protocol

The serial port's send and receive message protocol is as follows:

FrameHeader （0xAA）	Data Length (no frame header, frame tail, checksum)	Data Frame	Data Bit	Checksum High byte	Checksum Low byte	FrameTail （0xBB）

For the calculation of the checksum, refer to section 4.3.2.2.

Data Frame Definition

```
typedef enum {
    COMM_FW_VERSION = 65,
    COMM_JUMP_TO_BOOTLOADER=66,
    COMM_ERASE_NEW_APP=67,
    COMM_WRITE_NEW_APP_DATA=68,
    COMM_GET_VALUES=69,       //Get motor operating parameters
    COMM_SET_DUTY=70,         //The motor operates in duty cycle mode
    COMM_SET_CURRENT=71,    //The motor operates in current loop mode
    COMM_SET_CURRENT_BRAKE=72, //The motor operates in current bake mode
    COMM_SET_RPM=73,          //The motor operates in velocity loop mode
    COMM_SET_POS=74,          //The motor operates in position loop mode
    COMM_SET_HANDBRAKE=75,    //The motor operates om handbrake current loop mode
    COMM_SET_DETECT=76,       //The motor provides real-time feedback on the current position command
    COMM_ROTOR_POSITION=87,//The motor feedbacks the current position
    COMM_GET_VALUES_SETUP=16,//The motor requires instructions based on one or more parameters
    COMM_SET_POS_SPD=60,      // The motor operates in position-velocity loop mode
    COMM_SET_POS_MULTI=61,    //Set the motor movement to single circle motion mode
    COMM_SET_POS_SINGLE=62,   // Set the motor movement to multiple circles motion mode, range±100
    COMM_SET_POS_UNLIMITED=63, //Save
    COMM_SET_POS_ORIGIN=64, //Set the motor’s origin
} COMM_PACKET_ID;
```


E-mail:info@cubemars.com
https://www.cubemars.com/

4.3.2.1 Serial Port Message Protocol

Serial port command ： AA 01 45 18 61 BB // Command to get motor parameters, the motor will feedback its status once after receiving.

Example of motor's returned serial port command：AA 49 45 01 27 FC DF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 F2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FF FF FF FD 00 00 00 07 00 FF EF 54 F1 68 00 00 00 00 00 00 00 00 00 00 FF FF FF FC 86 D5 BB

// AA (frame header) + 49 (data length) + 45 (data frame) + MOS temperature (2 bytes) + motor temperature (2 bytes) + output current (4 bytes) + input current (4 bytes) + Id current (4 bytes) + Iq current (4 bytes) + motor throttle value (2 bytes) + motor speed (4 bytes) + input voltage (2 bytes) + reserved (24 bytes) + motor status code (1 byte) + motor external loop position value (4 bytes) + motor control ID number (1 byte) + temperature reserved (6 bytes) + Vd voltage value (4 bytes) + Vq voltage value (4 bytes) + CRC + BB (frame tail)

The formulas for converting some of the parameters sent by the motor are as follows:
MOS temperature = (float)buffer_get_int16(data, &ind) / 10.0
Motor temperature = (float)buffer_get_int16(data, &ind) / 10.0
Output current = (float)buffer_get_int32(data, &ind) / 100.0
Input current = (float)buffer_get_int32(data, &ind) / 100.0
Id current = (float)buffer_get_int32(data, &ind) / 100.0
Iq current = (float)buffer_get_int32(data, &ind) / 100.0
Motor throttle value = (float)buffer_get_int16(data, &ind) / 1000.0
Motor speed = (float)buffer_get_int32(data, &ind)
Input voltage = (float)buffer_get_int16(data, &ind) / 10.0
Motor external loop position = (float)buffer_get_int32(data, &ind) / 1000.0
Motor ID number = data
Motor Vd voltage = (float)buffer_get_int32(data, &ind) / 1000.0
Motor Vq voltage = (float)buffer_get_int32(data, &ind) / 1000.0

Motor Feedback Position Message Command
Serial port command: AA 02 4C 04 08 25 BB // After the motor receives, it sends the current position every 10ms

Example of motor's feedback position value transmission (the motor needs to be sent a feedback position message command in advance, and it will send the current position every 10ms after receiving):
Serial port command: AA 05 57 00 1A B6 64 6E CD BB
Pos = (float)buffer_get_int32(data, &ind) / 1000.0

E-mail:info@cubemars.com
https://www.cubemars.com/

Example of Motor Single or Multiple Parameter Acquisition Command

Serial port command: AA 05 13 00 00 00 01 FA A9 BB // Command to get motor temperature

Instruction note: This command can acquire a single or multiple motor parameters. The acquisition of parameters is determined by the 4-byte data segment. When the corresponding bit is 1, the motor will return the motor parameters of the corresponding bit; when it is 0, the field is excluded.

The motor parameters corresponding to the bits are as follows:

32-19 Bit	18 Bit	17 Bit	16 Bit	10-15 Bit	9 Bit	8 Bit	7 Bit
Reserved	Motor ID 1byte	Motor pos 4byte	Motor error 1byte	Reserved	Input voltage 2byte	Motor speed 4byte	Duty cycle 2byte
6 Bit	5 Bit	4 Bit	3 Bit	2 Bit	1 Bit		
Iq current 4byte	Id current 4byte	Input current 4byte	Output current 4byte	Motor temp 2byte	MOS temp 2byte		

After the motor receives this command, it will send the corresponding parameters.

Example: AA 07 13 00 00 00 01 01 21 DF BB BB // Feedback motor temperature

The formulas for converting some of the parameters sent by the motor are as follows:

MOS temperature = (float)buffer_get_int16(data, &ind) / 10.0

Motor temperature = (float)buffer_get_int16(data, &ind) / 10.0

Output current = (float)buffer_get_int32(data, &ind) / 100.0

Input current = (float)buffer_get_int32(data, &ind) / 100.0

Motor throttle value = (float)buffer_get_int16(data, &ind) / 1000.0

Motor speed = (float)buffer_get_int32(data, &ind)

Input voltage = (float)buffer_get_int16(data, &ind) / 10.0

Motor position = (float)buffer_get_int32(data, &ind) / 1000000.0

Motor ID number = data

Motor error status code

```
typedef enum {
    FAULT_CODE_NONE = 0,
    FAULT_CODE_OVER_VOLTAGE,// Over-voltage
    FAULT_CODE_UNDER_VOLTAGE,// Under-voltage
    FAULT_CODE_DRV,// Drive fault
    FAULT_CODE_ABS_OVER_CURRENT,// Motor over-current
    FAULT_CODE_OVER_TEMP_FET,// MOS over-temperature
    FAULT_CODE_OVER_TEMP_MOTOR,// Motor over-temperature
    FAULT_CODE_GATE_DRIVER_OVER_VOLTAGE,// Drive over voltage
    FAULT_CODE_GATE_DRIVER_UNDER_VOLTAGE,// Drive undervoltage
    FAULT_CODE_MCU_UNDER_VOLTAGE,// MCU under-voltage
    FAULT_CODE_BOOTING_FROM_WATCHDOG_RESET,// Under voltage
    FAULT_CODE_ENCODER_SPI,// SPI  Encoder fault
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
    FAULT_CODE_ENCODER_SINCOS_BELOW_MIN_AMPLITUDE,//Encoder limit exceeded
      FAULT_CODE_ENCODER_SINCOS_ABOVE_MAX_AMPLITUDE,//Encoder limit exceeded
    FAULT_CODE_FLASH_CORRUPTION,// FLASH Gault
    FAULT_CODE_HIGH_OFFSET_CURRENT_SENSOR_1,// Curretnt sampling channel 1 fault
    FAULT_CODE_HIGH_OFFSET_CURRENT_SENSOR_2,// Curretnt sampling channel 2 fault
    FAULT_CODE_HIGH_OFFSET_CURRENT_SENSOR_3,// Curretnt sampling channel 3 fault
    FAULT_CODE_UNBALANCED_CURRENTS,// Current imbalance

} mc_fault_code;
```

4.3.2.2 Serial Port Checksum

```
unsigned short crc16(unsigned char *buf, unsigned int len) {
    unsigned int i;
    unsigned short cksum = 0;
    for (i = 0; i < len; i++) {
    cksum = crc16_tab[(((cksum >> 8) ^ *buf++) & 0xFF)] ^ (cksum << 8);
    }
    return cksum;
}
const unsigned short crc16_tab[] = { 0x0000, 0x1021, 0x2042, 0x3063, 0x4084,
0x50a5, 0x60c6, 0x70e7, 0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad,
0xe1ce, 0xf1ef, 0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7,
0x62d6, 0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485, 0xa56a,
0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d, 0x3653, 0x2672,
0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4, 0xb75b, 0xa77a, 0x9719,
0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc, 0x48c4, 0x58e5, 0x6886, 0x78a7,
0x0840, 0x1861, 0x2802, 0x3823, 0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948,
0x9969, 0xa90a, 0xb92b, 0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50,
0x3a33, 0x2a12, 0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b,
0xab1a, 0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49, 0x7e97,
0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70, 0xff9f, 0xefbe,
0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78, 0x9188, 0x81a9, 0xb1ca,
0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f, 0x1080, 0x00a1, 0x30c2, 0x20e3,
0x5004, 0x4025, 0x7046, 0x6067, 0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d,
0xd31c, 0xe37f, 0xf35e, 0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214,
0x6277, 0x7256, 0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c,
0xc50d, 0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c, 0x26d3,
0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634, 0xd94c, 0xc96d,
0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab, 0x5844, 0x4865, 0x7806,
0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3, 0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e,
0x8bf9, 0x9bd8, 0xabbb, 0xbb9a, 0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1,
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
0x1ad0, 0x2ab3, 0x3a92, 0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b,
0x9de8, 0x8dc9, 0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0,
0x0cc1, 0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0 };


//int16    Data bit organization
void buffer_append_int16(uint8_t* buffer, int16_t number, int32_t *index) {
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//uint16 Data bit organization
void buffer_append_uint16(uint8_t* buffer, uint16_t number, int32_t *index) {
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//int32 Data bit organization
void buffer_append_int32(uint8_t* buffer, int32_t number, int32_t *index) {
    buffer[(*index)++] = number >> 24;
    buffer[(*index)++] = number >> 16;
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//uint32Data bit organization
void buffer_append_uint32(uint8_t* buffer, uint32_t number, int32_t *index) {
    buffer[(*index)++] = number >> 24;
    buffer[(*index)++] = number >> 16;
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//int64 Data bit organization
void buffer_append_int64(uint8_t* buffer, int64_t number, int32_t *index) {
    buffer[(*index)++] = number >> 56;
    buffer[(*index)++] = number >> 48;
    buffer[(*index)++] = number >> 40;
    buffer[(*index)++] = number >> 32;
    buffer[(*index)++] = number >> 24;
    buffer[(*index)++] = number >> 16;
    buffer[(*index)++] = number >> 8;
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
    buffer[(*index)++] = number;
}

//uint64 Data bit organization
void buffer_append_uint64(uint8_t* buffer, uint64_t number, int32_t *index) {
    buffer[(*index)++] = number >> 56;
    buffer[(*index)++] = number >> 48;
    buffer[(*index)++] = number >> 40;
    buffer[(*index)++] = number >> 32;
    buffer[(*index)++] = number >> 24;
    buffer[(*index)++] = number >> 16;
    buffer[(*index)++] = number >> 8;
    buffer[(*index)++] = number;
}

//CRC Checksum
unsigned short crc16(unsigned char *buf, unsigned int len) {
    unsigned int i;
    unsigned short cksum = 0;
    for (i = 0; i < len; i++) {
        cksum = crc16_tab[(((cksum >> 8) ^ *buf++) & 0xFF)] ^ (cksum << 8);
    }
    return cksum;
}

  // Packet organization and transmission
void packet_send_packet(unsigned char *data, unsigned int len, int handler_num) {
    int b_ind = 0;
    unsigned short crc;
    if (len > PACKET_MAX_PL_LEN) {
        return;
    }
    if (len <= 256) {
        handler_states[handler_num].tx_buffer[b_ind++] = 2;
        handler_states[handler_num].tx_buffer[b_ind++] = len;
    } else {
        handler_states[handler_num].tx_buffer[b_ind++] = 3;
        handler_states[handler_num].tx_buffer[b_ind++] = len >> 8;
        handler_states[handler_num].tx_buffer[b_ind++] = len & 0xFF;
    }

    memcpy(handler_states[handler_num].tx_buffer + b_ind, data, len);
    b_ind += len;
```


E-mail:info@cubemars.com
https://www.cubemars.com/

```
    crc = crc16(data, len);
    handler_states[handler_num].tx_buffer[b_ind++] = (uint8_t)(crc >> 8);
    handler_states[handler_num].tx_buffer[b_ind++] = (uint8_t)(crc & 0xFF);
    handler_states[handler_num].tx_buffer[b_ind++] = 3;

    if (handler_states[handler_num].send_func) {
        handler_states[handler_num].send_func(handler_states[handler_num].tx_buffer,
b_ind);
    }
}
```


E-mail:info@cubemars.com
https://www.cubemars.com/

4.4 Control Command Examples

4. 4. 1 CAN Port Control Command Examples

Frame format: Extended frame - Data frame (using motor ID 0x68 as an example)

Mode	ID	DATA	Descriptions
Duty Cycle Mode	00 00 00 68	00 00 4E 20	0.2 Duty Cycle
	00 00 00 68	FF FF B1 E0	-0.2 Duty Cycle
Current Loop	00 00 01 68	FF FF F0 60	-4A IQ Current
	00 00 01 68	00 00 0F A0	4A IQ Current
Brake Current Mode	00 00 02 68	FF FF F0 60	-4A Brake current
	00 00 02 68	00 00 0F A0	4A Brake current
Velocity Loop	00 00 03 68	00 00 13 88	5000 ERPM Electrical Speed
	00 00 03 68	FF FF EC 78	-5000 ERPM Electrical Speed
Position Loop	00 00 04 68	00 5B 8D 80	Motor rotates to 600degrees
	00 00 04 68	FF A4 72 80	Motor rotates to -600degrees
Position-Velocity Loop	00 00 06 68	00 98 96 80 03 E8 03 E8	Motor rotates to 1000degrees 10000 ERPM electrical speed 10000 electrical acceleration
	00 00 06 68	FF 67 69 80 FC 18 FC 18	Motor rotates to -1000degrees -10000 ERPM electrical speed -10000 electrical acceleration
MIT Velocity Loop	00 00 08 68	00 06 66 7F FF 8F 57 FF	Kd set to 2, speed set to 6rad/s
	00 00 08 68	00 06 66 7F FF 70 97 FF	Kd set to 2, speed set to -6rad/s

E-mail:info@cubemars.com
https://www.cubemars.com/

MIT Position Loop	00 00 08 68	01 06 66 BD 70 7F F7 FF	Kp set to 2, Kd set to 2, motor rotates to 6rad
	00 00 08 68	01 06 66 42 8F 7F F7 FF	Kp set to 2, Kd set to 2, motor rotates to -6rad
MIT Torque Loop	00 00 08 68	00 00 00 7F FF 7F F8 3F	2A IQ Current
	00 00 08 68	00 00 00 7F FF 7F F8 7E	4A IQ Current

4.4.2 Serial Port Control Command Examples

Mode	Serial Port Command	Descriptions
Duty Cycle Mode	AA 05 46 00 00 4E 20 D6 4C BB	0.20 Duty Cycle
	AA 05 46 FF FF B1 E0 88 3F BB	-0.20 Duty Cycle
Brake Current Mode	AA 05 48 00 00 13 88 55 E5 BB	5A Brake Current
	AA 05 48 FF FF EC 78 3D C5 BB	-5A Brake Current
Velocity Loop	AA 05 49 00 00 03 E8 90 61 BB	1000 ERPM Electrical Speed
	AA 05 49 FF FF FC 18 F8 41 BB	-1000 ERPM Electrical Speed
Position Loop	AA 05 4A 0A BA 95 00 E1 4D BB	Motor rotates to 180 degrees
	AA 05 4A 05 5D 4A 80 84 93 BB	Motor rotates to 90 degrees
Position-Velocity Loop	AA 0D 3C 00 02 BF 20 00 00 13 88 00 00 75 30 18 1C BB	180 degrees, speed 5000ERPM, acceleration 30000/S
Current Loop	AA 05 47 00 00 13 88 30 1C BB	5 A IQ Current
	AA 05 47 FF FF EC 78 58 3C BB	- 5 A IQ Current
MIT Velocity Loop	AA 15 60 00 00 00 00 00 00 17 70 00 00 00 00 00 00 00 00 00 00 07 D0 93 DA BB	Kd set to 2, speed set to 6rad/s
	AA 15 60 00 00 00 00 FF FF E8 90 00 00 00 00 00 00 00 00 00 00 07 D0 87 5C BB	Kd set to2, speed set to -6rad/s

E-mail:info@cubemars.com
https://www.cubemars.com/

MIT Position Loop	AA 15 60 00 00 17 70 00 00 00 64 00 00 00 00 00 00 07 D0 00 00 07 D0 91 BC BB	Kp set to 2, Kd set to 2, motor rotates to 6rad
	AA 15 60 FF FF E8 90 00 00 00 64 00 00 00 00 00 00 07 D0 00 00 07 D0 C9 10 BB	Kp set to 2, Kd set to 2, motor rotates to -6rad
MIT Torque Loop	AA 15 60 00 00 00 00 00 00 00 64 00 00 07 D0 00 00 00 00 00 00 00 00 CB B7 BB	2A IQ Current
	AA 15 60 00 00 00 00 00 00 00 64 00 00 0F A0 00 00 00 00 00 00 00 00 2A 27 BB	4A IQ Current

E-mail:info@cubemars.com
https://www.cubemars.com/