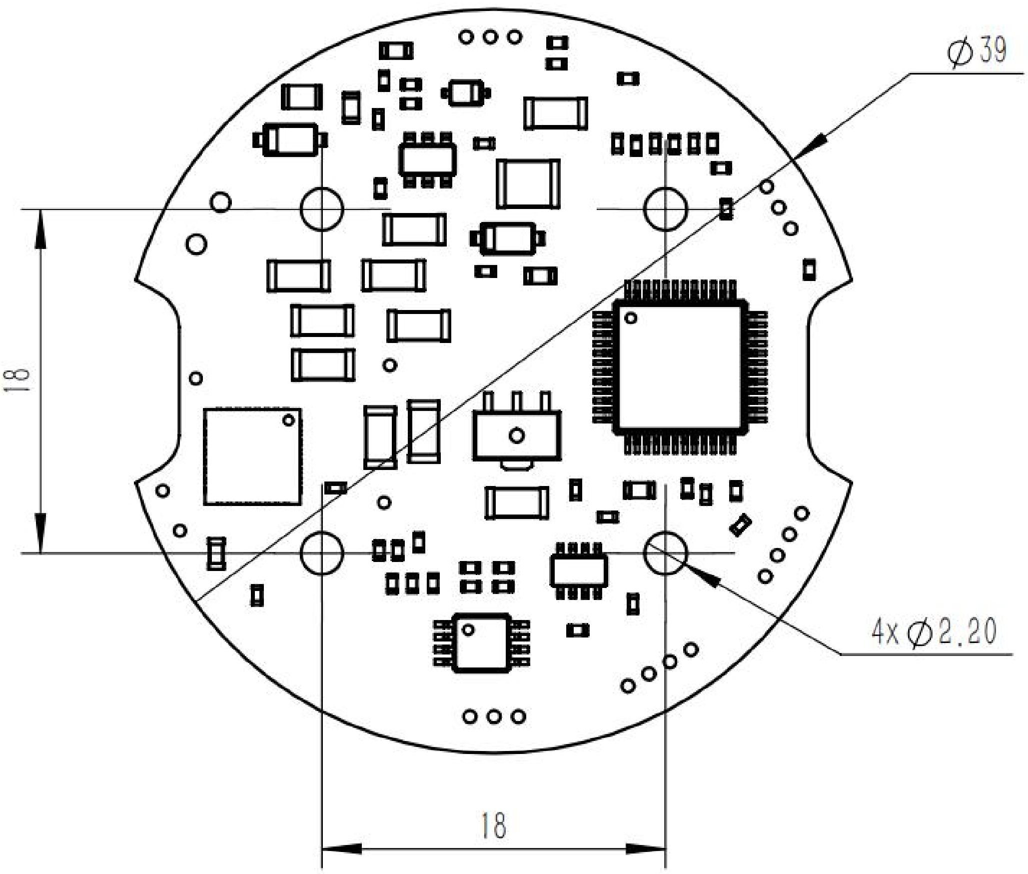
SensoredFoc24V-GD32

Drive instruction manual

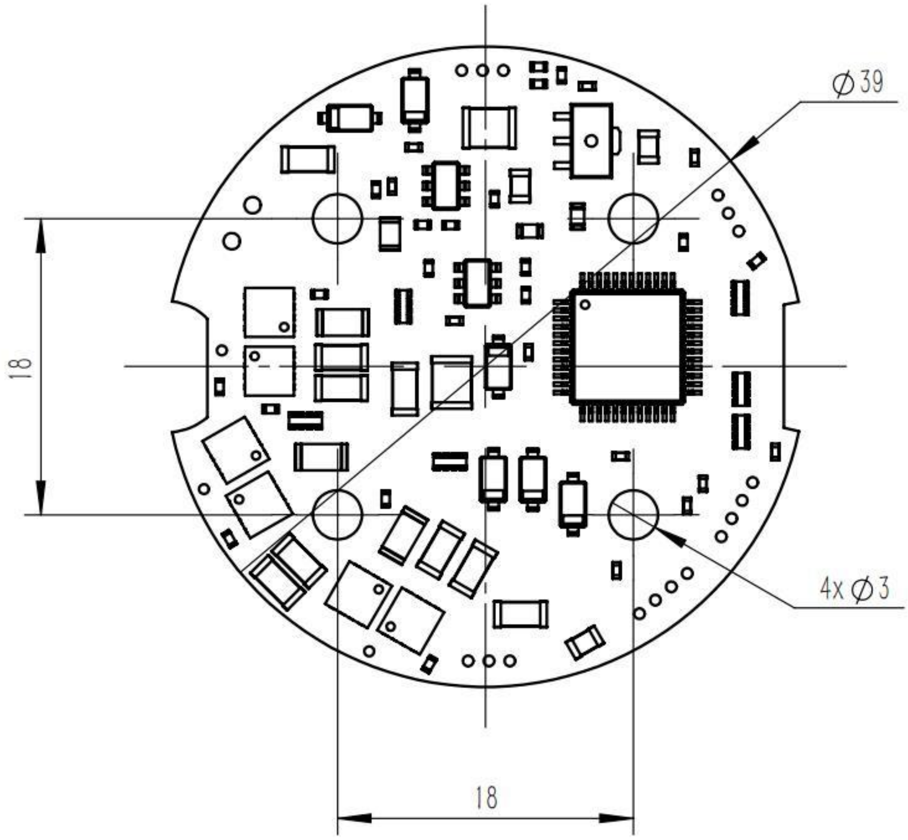
Catalog

1、Overview

1、Drive size parameter

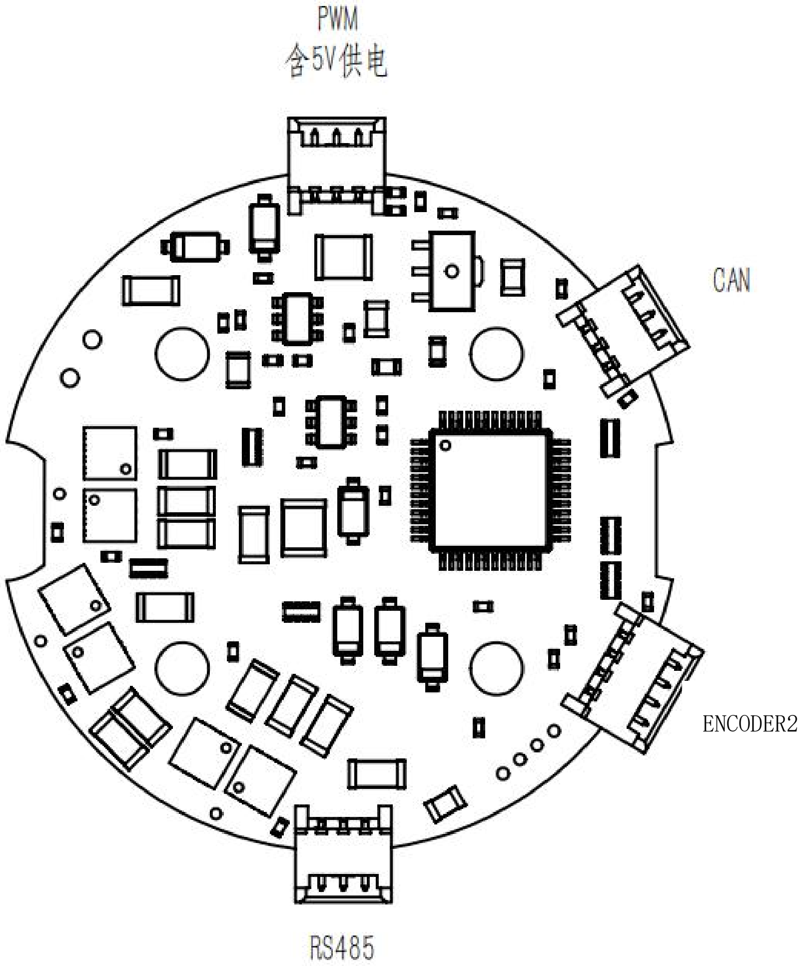


Standard Edition



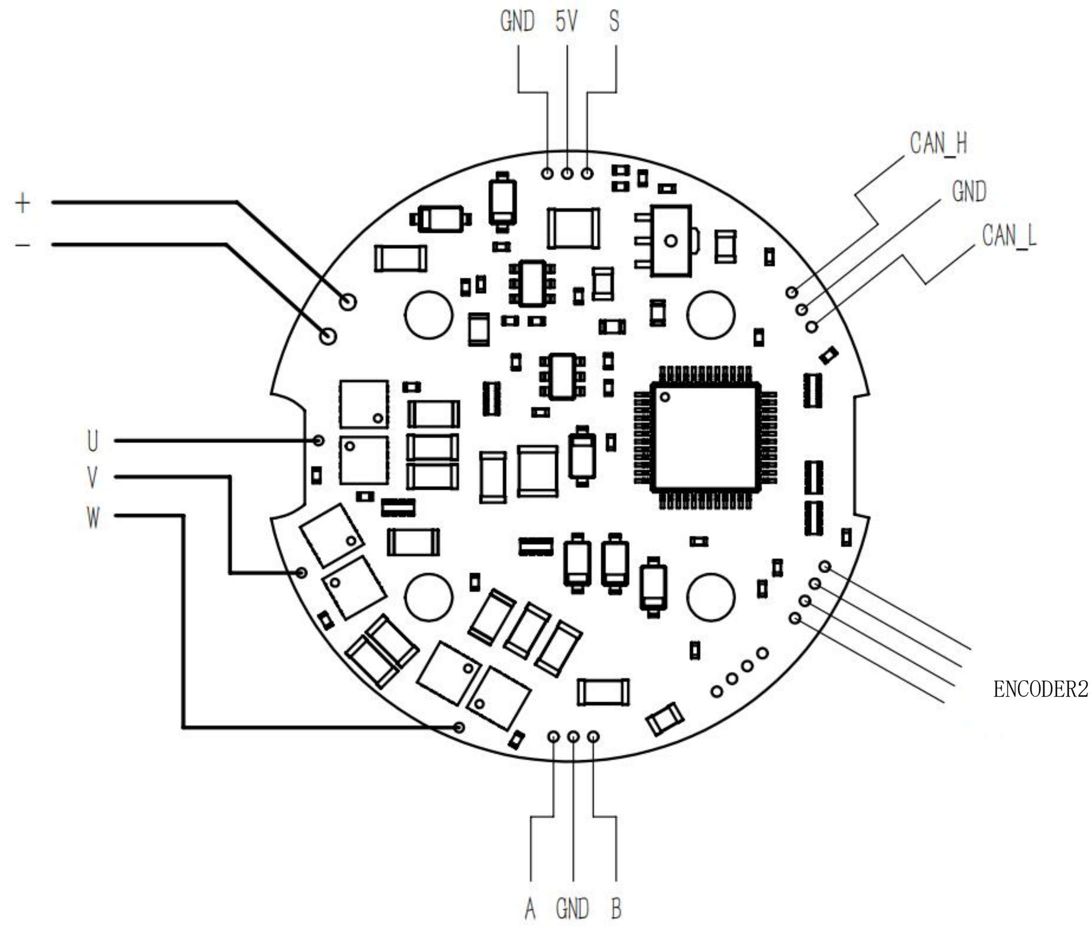
High power version

2、Drive interface definition



* The interface definition of the standard version is consistent with that of the high-power version.

3. Driver pin definition



* The pin definition of the standard version is the same as that of the high-power version

4. Electrical parameters

Parameter	Numerical value	Explain
Rated voltage	0-26V	
Rated current	（2A）3A	
Peak voltage	30V	
Peak current	（4A）6A	* Operating current setting greater than (2A) 3A requires additional heat sink
Single-lap position accuracy	14 bits	
Master control chip	GD32	
Interface parameters	Explain	

PWM	Control the motor rotation according to the configuration mode (position/speed/torque), with 5V power supply
ENCODER2	The second encoder interface is used to set up the absolute position
RS485	Driver parameter adjustment interface, firmware upgrade interface
CAN	Bus control interface, compatible with MIT control protocol

* Parameters in parentheses are standard version parameters

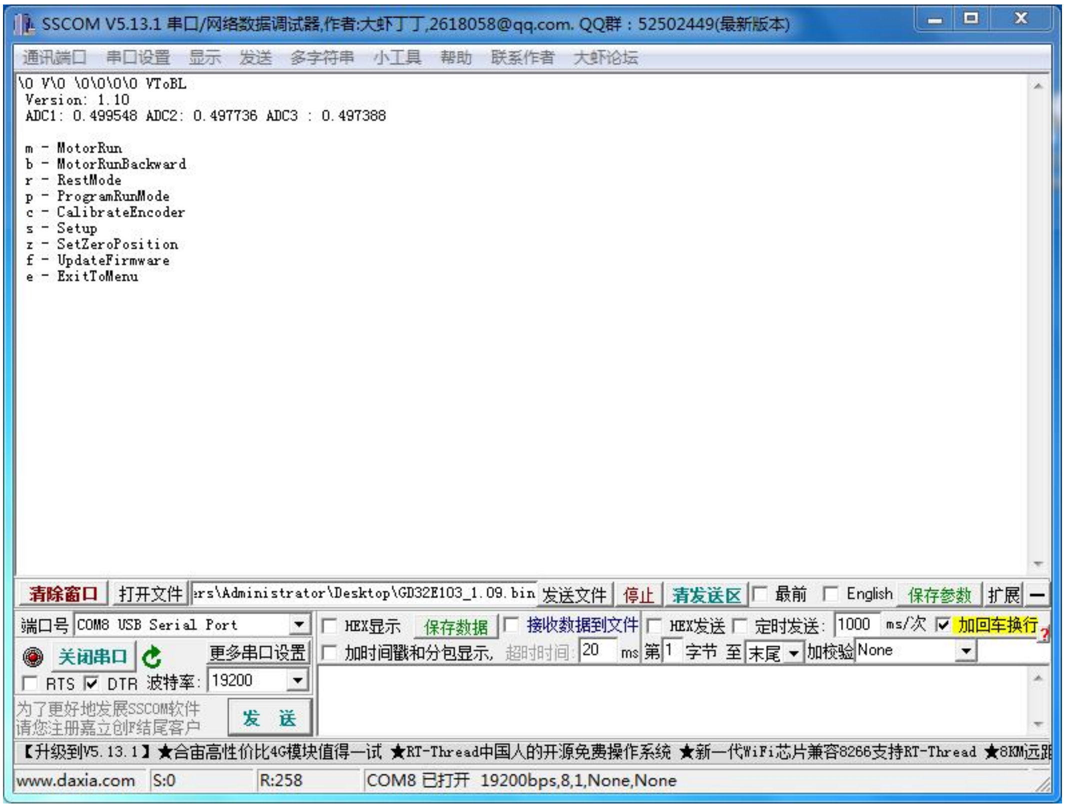
2、Configuration instructions via RS485

RS485 baud rate fixed 19200, 8-bit data, no check, 1stop bit

All commands end with 0x0D 0x0a (carriage return-line feed)

All data written through the RS485 interface are stored in the chip and will not be lost in case of power failure.

1、Introduction to the first-level directory



m - Motor Run Mode

Forward operation of motor

Speed Mode-32 RPM

Moment Mode-1 NM

Position Mode-Locked at the current position

b - Motor Run Backward Mode

The motor rotates in the reverse direction

Speed Mode- (-32 RPM)

Moment Mode-1 NM

Position Mode-Locked at the current position

r - Rest Mode

The motor stops

p - Program Run Mode

Set the operating mode

c - Calibrate Encoder

Calibrate motor encoder 1

s - Setup

Drive parameter settings

z - Set Zero Position

Set zero position of motor position

f －UpdateFirmware

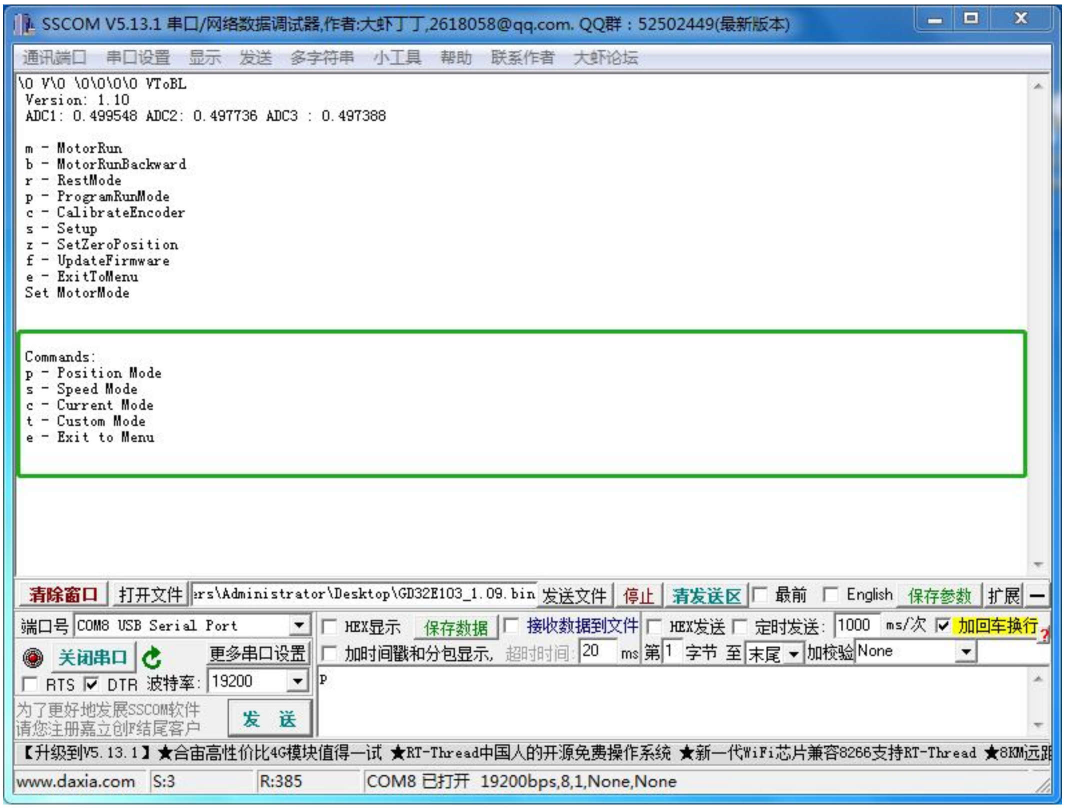
Upgrade the system firmware

e - Exit to Menu

Exit the menu

2、Introduction to the secondary directory

1）p - Program Run Mode



p - Position Mode

Location mode

s - Speed Mode

Speed mode

c - Current Mode

Moment mode

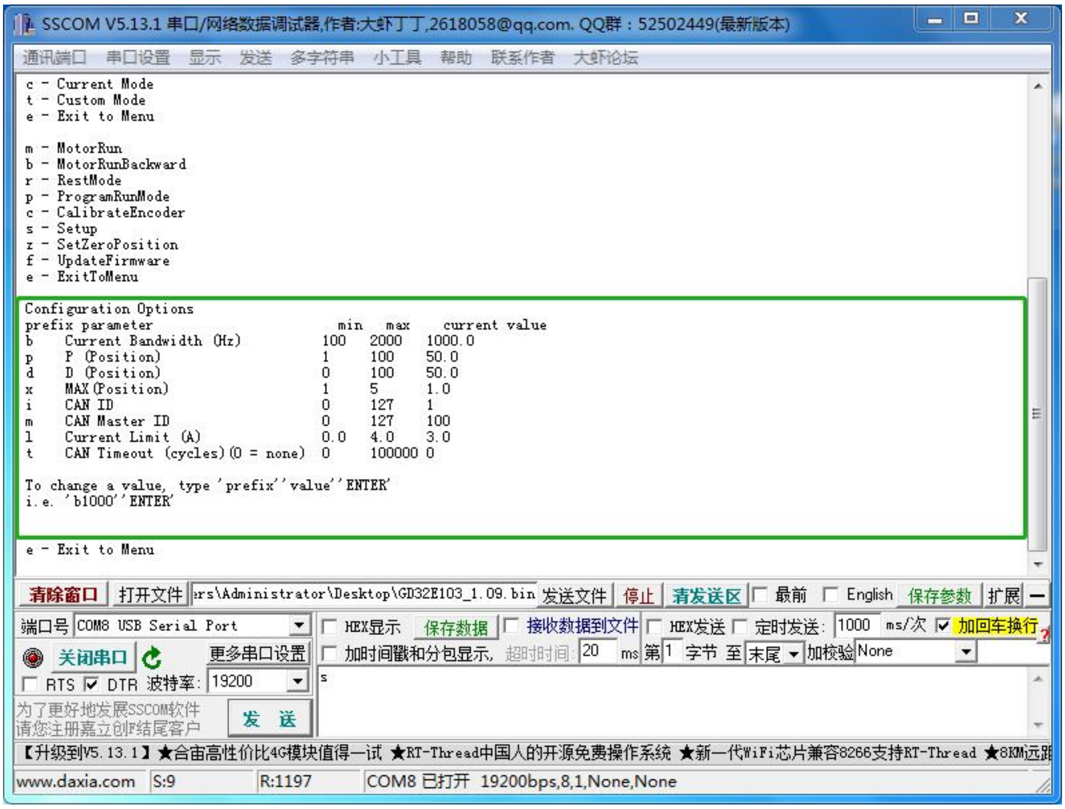
t - Custom Mode

*** Custom mode (not available for retail)**

e - Exit to Menu

Return to the previous menu

2）s - Setup



b - Current Bandwidth (Hz)

The specified current loop bandwidth range is 100-2000 in Hz. The larger this value is, the higher the dynamic response of the motor will be.

p - P (Position)

The P parameter when PWM controls the position loop, the greater the P, the greater the motor hardness

d - D (Position)

D parameter when PWM controls the position loop, the larger D is, the larger the motor damping is

*** Note that the above two items are invalid for can control, only for PWM control.**

x - MAX(Position) Pmax

Can controlled maximum number of turns

The maximum number of turns of can control is + -1 when Pmax = 1 without end encoder.

The maximum number of turns for Pmax = 1 can control with end encoder is +-0.5 turns, and Pmax needs to be fixed to 1

i - CAN ID

Local can ID 0 -- 127

m - CAN Master ID

Host can ID 0--127

Can bus IDs are not repeatable, so be careful not to set the device IDs to the same values

l - Current Limit (A)

Maximum current 0 – 4A (standard version) 0-6A (high power version)

t - CAN Timeout (cycles)(0 = none)

Timeout setting of can communication in ms

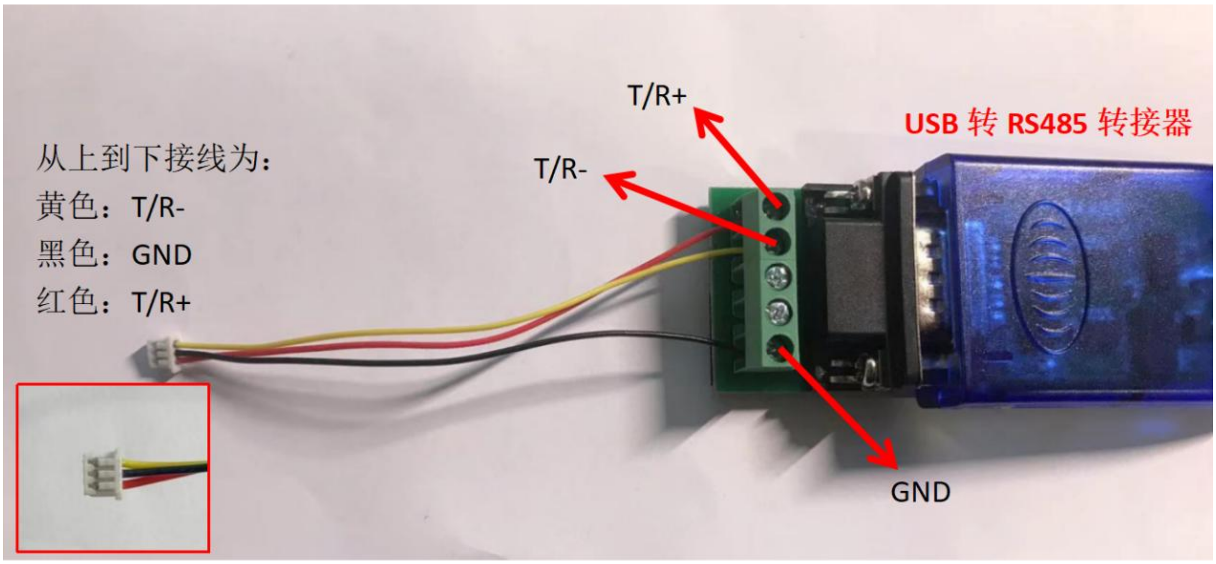
When the set time is exceeded, the equipment will automatically stop the motor. This function is used to prevent the motor from misoperation caused by communication failure.

e - Exit to Menu

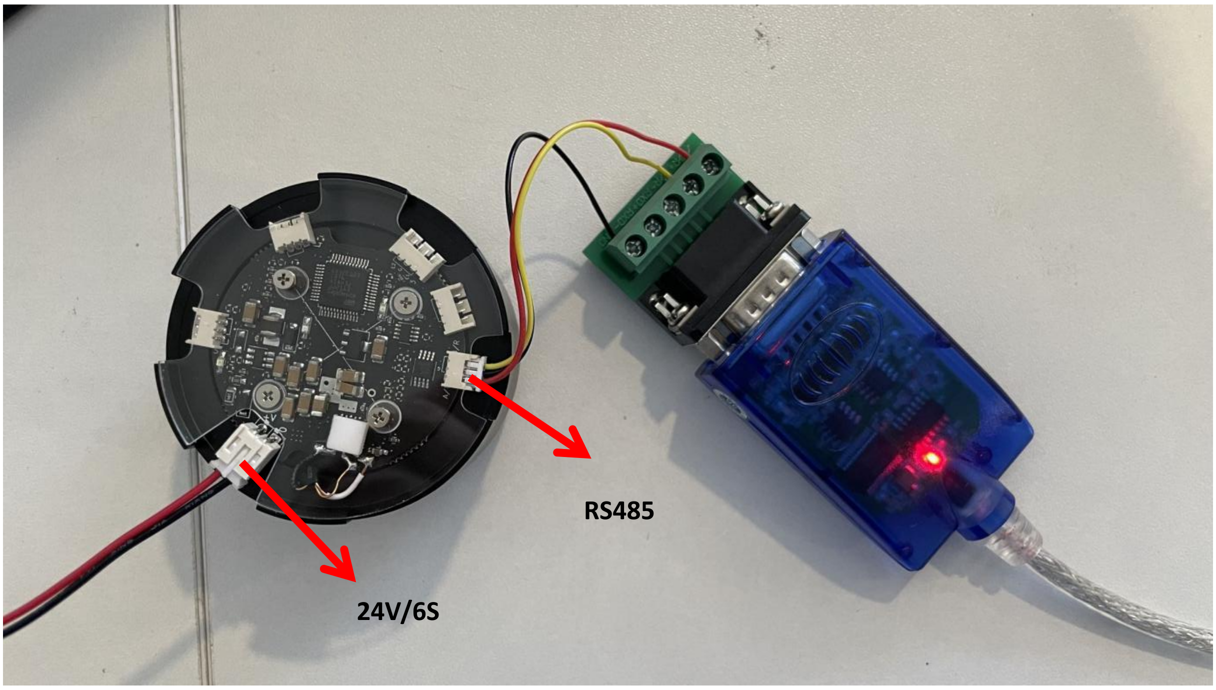
Return to the previous menu

3. Connection method of serial port assistant configuration driver

1) Prepare and connect the USB to RS485 converter



2) Link Driver



3) Power on 24V and configure CAN _ ID, current and other relevant parameters through serial port assistant

3、Introduction to drive mode

1、Introduction to PWM Control Mode

PWM signal frequency 50-1000Hz, pulse width 0.8ms-2ms, 1.45ms-1.55ms is the dead zone of the middle position, and the motor does not work

According to the motor operating mode

1) Constant torque control mode

Pulse width time = t unit ms, t > 1.55 forward operation, t < 1.45 reverse operation

Forward rotation:

Torque = (t - 1.55) * 2.5 units NM

Reverse:

Torque = (1.45 - t) * 2.5 units NM

2) Speed control mode

Pulse width time = t unit ms, t > 1.55 forward operation, t < 1.45 reverse operation

Forward rotation:

Speed = (t - 1.55) * 1000 units RPM

Reverse:

Speed = (1.45 – t) * 1000 units RPM

3) Position control mode

The position is always maintained and the motor works continuously.

Position = (t – 1.5) * 1.952 unit circle

The Position symbol represents the direction of motor deflection

When the PWM signal is active, the motor only responds to the PWM signal

When the PWM signal is invalid, the motor responds to the CAN command

2、Introduction of CAN bus control mode

CAN rate fixed 1Mbps

CAN slave address: slave addr is designated by the serial port, representing the local address.

CAN host address: master addr is specified by the serial port, representing the address of the control end

The CAN working mode is slave receiving-response. The slave does not actively send data to the bus. Only when it receives data, it responds to the master. The master controller actively sends data to all slave devices.

CAN data frame format: CAN standard frame with 11-bit ID

The received data frame length is 8 Byte, which is divided into four types of data frames.

1） Control the motor

0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xfc

Start the motor

0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xfd

Stop the motor

2) Set the motor position zero point

0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xfe

3）Switch the motor operation mode

0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xf9	Switch to torque mode
0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xfA	switch to speed mode

0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xfB Switches to position mode

4) Setting of motor operation parameters

Designated position, 16 digits, range 0-65536, representing the motor rotating to and holding the designated position within the range of -12.5 rad to 12.5 rad (-1 * PMax turns to + 1 * PMax turn

Calculation formula position = (p-32768)/32768 * 12.5 * 0.5 * Pmax unit rad

Byte0 specifies the upper 8 bits of the position p

Byte1 specifies the lower 8 bits of the position p

Specify speed, 12 digits, range 0-4096, specify motor speed

Calculation formula speed = (v-2048)/2048 * 65 units rad/s

Byte2 specifies the upper 8 bits of speed V

Byte3 (upper 4 bits) specifies the lower 4 bits of the velocity V

Set position ring PD control P value, 12 digits, range 0-4096

Calculation formula KP = set _ KP/4096 * 500

Byte3 (lower 4 bits) sets the upper 4 bits of Kp

Byte4 setting Kp lower 8 bits

Set position ring PD control D value, 12 digits, range 0-4096

Calculation formula: kd = set _ kd/4096 * 5

Byte5 Set Kd high 8 bits

Byte6 (upper 4 bits) sets the lower 4 bits of Kd

Specified torque, 12 digits, range 0-4096, constant torque for specified motor

Calculation formula torque = (t-2048)/2048 * 4 unit ampere

Byte6 (lower four digits) specifies torque T upper four digits

Byte7 Specified torque t Lower 8 bit

5) CAN response

CAN reply frame, length 6 Byte

Byte 0 Native ID

Byte 1 current position p upper 8 bits

Byte 2 current position p lower 8 bits

Byte 3 Current speed V 8 bits higher

Byte 4 (high 4 bits)   Current speed V low 4 bits

Byte 4 (Low 4 Digits)   Current Torque tHigh 4 Digit

Byte 5   Current torque tLower 8 bit

Calculation formula

Calculation formula:

Torque = (t-2048)/2048 * 4 unit ampere

Speed = (v-2048)/2048 * 65 units rad/s

Position = (p-32768)/32768 * 12.5 * 0.5 * Pmax unit rad

a. Constant torque control model

Pulse width time = t unit ms, t > 1.55 forward operation, t < 1.45 reverse operation

Forward rotation:

Torque = (t - 1.55) * 2.5 units NM

Reverse:

Torque = (1.45 - t) * 2.5 units NM

b. Speed control mode

Pulse width time = t unit ms, t > 1.55 forward operation, t < 1.45 reverse operation

Forward rotation:

Speed = (t - 1.55) * 1000 units RPM

Reverse:

Speed = (1.45 - t) * 1000 units RPM

c. Position control mode

The position is always maintained and the motor works continuously.

Position = (t - 1.5) * 1.952 unit circle

The Position symbol represents the direction of motor deflection

3、Introduction and Routine of CAN Bus Control Mode

```
uint16_t  float_to_uint(float v,float v_min,float v_max,uint32_t width)
{
```

```
    float temp;

    int32_t utemp;

    temp = ((v-v_min)/(v_max-v_min))*((float)width);

    utemp = (int32_t)temp;

    if(utemp < 0)

        utemp = 0;

    if(utemp > width)

        utemp = width;

    return utemp;

}

Void CanCmdSend(void)

{

    Switch(CanMode)

    {

    Case 0:  //Motor Run

        g_transmit_message.tx_data[0]==0xFF;

        g_transmit_message.tx_data[1]==0xFF;

        g_transmit_message.tx_data[2]==0xFF;

        g_transmit_message.tx_data[3]==0xFF;

        g_transmit_message.tx_data[4]==0xFF;

        g_transmit_message.tx_data[5]==0xFF;

        g_transmit_message.tx_data[6]==0xFF;

        g_transmit_message.tx_data[7]==0xFC;

        CanSend();

        Break;

    Case 1:   //Motor Rest

        g_transmit_message.tx_data[0]==0xFF;

        g_transmit_message.tx_data[1]==0xFF;

        g_transmit_message.tx_data[2]==0xFF;

        g_transmit_message.tx_data[3]==0xFF;

        g_transmit_message.tx_data[4]==0xFF;
```

```
        g_transmit_message.tx_data[5]==0xFF;
        g_transmit_message.tx_data[6]==0xFF;
        g_transmit_message.tx_data[7]==0xFD;
        CanSend();
        Break;
    Case 2:   //Set Zero Position(temp)
        g_transmit_message.tx_data[0]==0xFF;
        g_transmit_message.tx_data[1]==0xFF;
        g_transmit_message.tx_data[2]==0xFF;
        g_transmit_message.tx_data[3]==0xFF;
        g_transmit_message.tx_data[4]==0xFF;
        g_transmit_message.tx_data[5]==0xFF;
        g_transmit_message.tx_data[6]==0xFF;
        g_transmit_message.tx_data[7]==0xFE;
        CanSend();
        Break;
    Case 3:
        s_p_int = float_to_uint(f_position, P_MIN, P_MAX, 65535);
        s_v_int = float_to_uint(f_velocity, V_MIN, V_MAX, 4096);
        s_Kp_int = float_to_uint(f_kp, 0 , KP_MAX, 4096);
        s_Kd_int = float_to_uint(f_kd, 0 , KD_MAX, 4096);
       s_c_int = float_to_uint(f_current, -C_MAX, C_MAX, 4096);
        g_transmit_message.tx_data[0] = s_p_int>>8;
        g_transmit_message.tx_data[1] = s_p_int&0xFF;
        g_transmit_message.tx_data[2] = s_v_int>>4;;
        g_transmit_message.tx_data[3] = ((s_v_int&0xF)<<4) + (s_Kp_int >>8);
        g_transmit_message.tx_data[4] = s_Kp_int &0xFF;
        g_transmit_message.tx_data[5] = s_Kd_int>>4;
        g_transmit_message.tx_data[6] = ((s_Kd_int &0xF)<<4) + (s_c_int >>8);
        g_transmit_message.tx_data[7] = s_c_int&0xFF;;
        CanSend();
```

```
            Break;
        }
    }
```

4、The second encoder

The second encoder can realize that even if the end position is changed after power failure, the position before power failure can be found after power on again.

The second encoder has been calibrated at the factory. Do not disassemble the circuit board and magnetic ring of the second encoder.

*** If the second encoder accessory is disassembled, it must be returned to the factory for re-calibration.**

In the state with the second encoder, Pmax needs to be configured as 1.