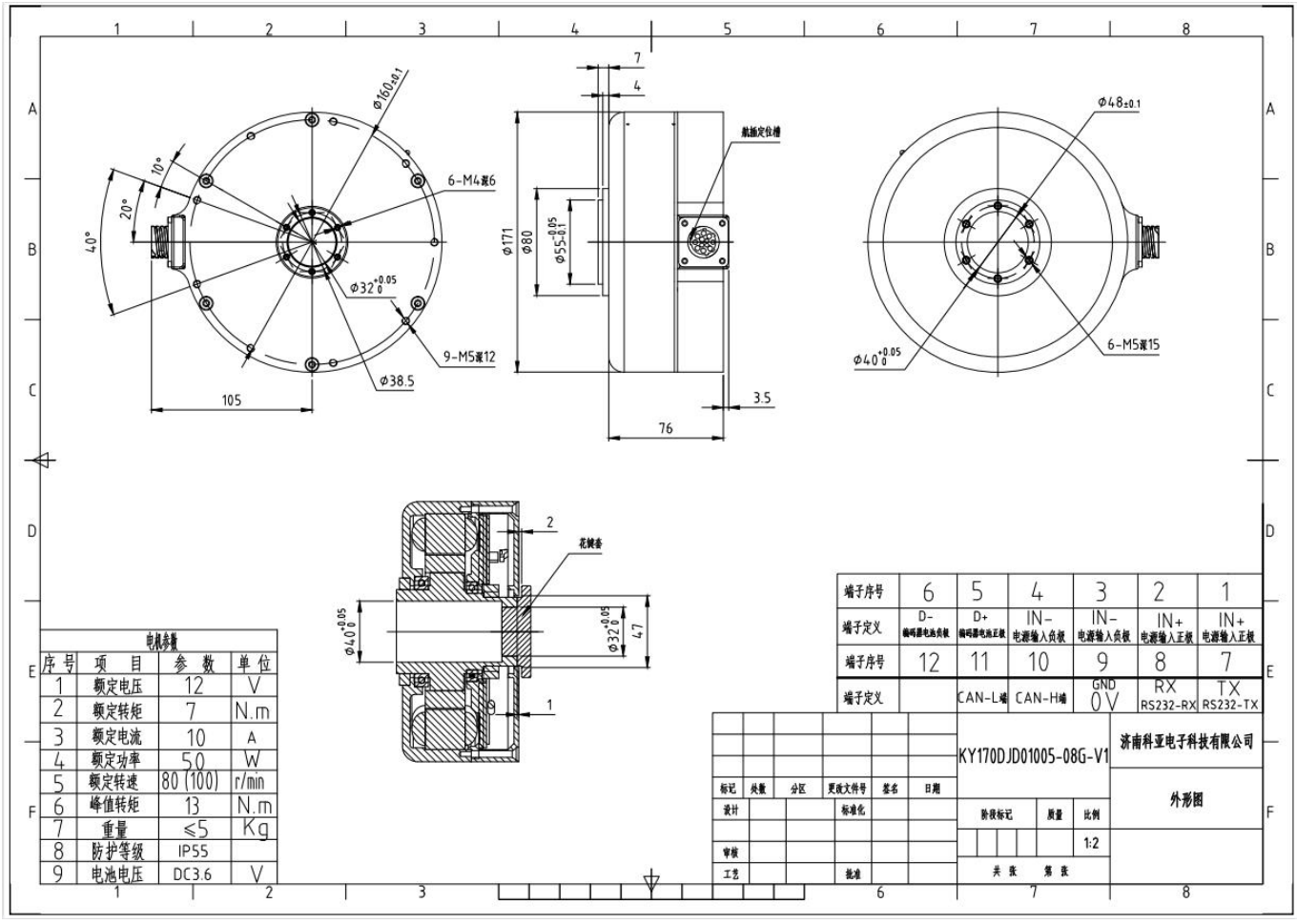
Auto Steering Motor

(Model: KY170DJD01005-08G-V1)

User's Manual (V1.1)

I. Overview

1.1 KY170DJD01005-08G-V1 Motor Parameters



1.2 Specification

- Continuous current 10A, maximum peak current 20A
- Rated DC supply voltage 12VDC (the power supply ranges 9 ~ 18VDC); Rated working power: 24VDC (the power supply ranges 9 ~ 28VDC, limit voltage 36V for hardware);

1.3 Operation Condition

1.3.1 Power supply

- Rated working power: 12VDC (the power supply ranges 9 ~ 18VDC); Rated working power: 24VDC (the power supply ranges 9 ~ 28VDC); Battery-powered is recommended.
- It can provide instantaneous current overload capability of 2 times continuous current.

1.3.2 operating requirements:

- Operators must wear electrostatic wristbands;
- The workbench must take anti-static measures;
- It is strictly forbidden to plug and unplug the aviation plug with power on.

1.3.3 Working environment:

- Operating temperature: -25~55°C (based on ambient temperature);

Storage temperature: -35~65°C (based on ambient temperature);

- Humidity: 5%--90%RH, condensation (25°C)
- Protection level: IP55, front of the motor is exposed to rain.
- Insulation performance: input to the chassis DC600V, leakage current 0.07mA.The insulation resistance is 20 MΩ or more.
- Three-proof requirements: meet the requirements of three defenses (dust, moisture, salt spray).
- Vibration requirements: Frequency 5Hz ~ 25Hz, amplitude 3mm, 0.09g.
  Frequency 25Hz~200Hz, amplitude 1.47mm, 116g.
  30min at each direction (Horizontal, vertical, and longitudinal directions)
- Cooling method: natural cooling

II. Functional Technical Indicators

2.1 Main Functions

- Working mode: Speed mode, Torque mode, Absolute position mode, Relative position mode
- Feedback:17-bit multi-turn absolute encoder.
- Control: RS232, CAN2.0,CANopen;
- LED indicator for faults;
- To realize motor speed control and data reading via RS232;
- Internal temperature monitoring and protection inside the driver;
- Overcurrent and overload protection
- Overvoltage and undervoltage protection
- Locked-rotor and over speed protection
- Motor short circuit protection

2.2. Working Mode Configuration Table

Work Modes	Control Signals	Feedback Part
Speed mode	RS232,CAN,CANopen	Absolute encoder
Position mode	RS232,CAN,CANopen	Absolute encoder

2.3. Technical Parameters

Parameters	Label	Parameter value	Unit
Voltage	U	9-28	VDC
Max continuous current	I_c	10	A
Max peak current	I_{max}	20	A
PWM switching frequency	f_{pwm}	10	kHz
Output encoder power supply	$+5V_{out}$	5	VDC
	I_{cc}	100	mA

Under voltage	Vu	7 (adjustable)	V
Over voltage	Vo	32(adjustable)	V
Operating temperature	Industrial grade (standard product)	-25 ~ +55	℃
	Military grade	-40 ~ +65	
Storage temperature	Industrial grade (standard product)	-35 ~ +65	℃
	Military grade	-55 ~ +85	

III. Port Description

3.1. Interface Definition

Port	Definition	Description	Recommend wire
1、2	IN+	Power input +	1.5mm²
	IN+	Power input +	1.5mm²
3、4	IN-	Power input -	1.5mm²
	IN-	Power input -	1.5mm²
5	D+	Encoder battery+	0.5 mm²
6	D-	Encoder battery-	0.5 mm²
7	TX	RS232—TX	0.5 mm²
8	RX	RS232—RX	0.5 mm²
9	GND	0 V	0.5 mm²
10	CAN-H	CAN-H	0.5 mm²
11	CAN-L	CAN-L	0.5 mm²
12	NG	Idle	

(1) TX, RX, GND:

RS232 interface, to achieve command control, as well as parameter settings, operating state commissioning, etc.

(2) CAN-H, CAN-L: CAN interface,to achieve command control.

(3) D+, D- encoder battery: an external battery is required when the power-off position is memorized (use 3.6V battery specially for encoders) After installing the battery, the motor needs to be powered off and restarted to take effect. If the battery is subsequently replaced or the battery is powered off for multiple turns, the position will be cleared.

(4) IN+ 、IN- ：

Since the power cord required for vehicles is long, the voltage drop will increase due to line loss when the current is large. It is recommended to select the wire specifications according to below table.

Cable length (m)	Cable diameter (mm²)	Allowable Continuous Amps
1-3	2.5	＜17A
3-4.5	4	＜25A

Note: When the motor “undervoltage alarm”, there may be the following reasons:

(1) The battery is aging, and the internal resistance of the battery will increase after a long time of use, thereby reducing the battery's discharge capacity.

(2) The steering hydraulic pump is aging, the flow valve is blocked, etc., which causes the steering resistance to increase and the motor current to increase.

(3) The cable diameter is too thin, the voltage drop is large, and when the torque is large, the voltage is pulled down, causing the driver to detect undervoltage.

3.2. Serial Port Connection

Using high-speed standard serial cable, DB9 plug meets the label definition:

Driver label	RS232 cable pin No.
TX	2
RX	3
GND	5

IV. Operating Instructions

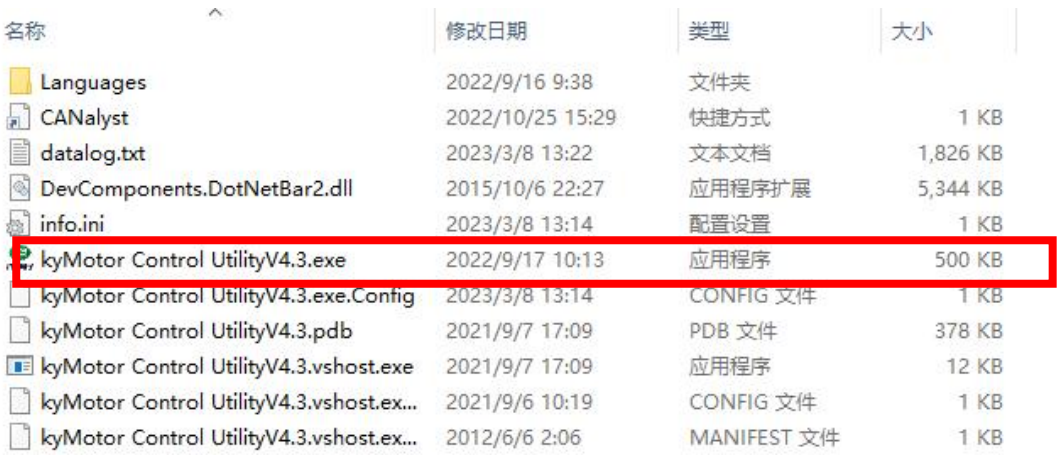
4.1. The auto steering motor software

4.1.1 Configuration instructions

- Servo controller parameters can be set by the software.
- The software communicates with the motor driver through RS232, and the baud rate is 115200bit.
- The software is developed under the .NET environment. XP systems need to have .NET 4.0 installed.

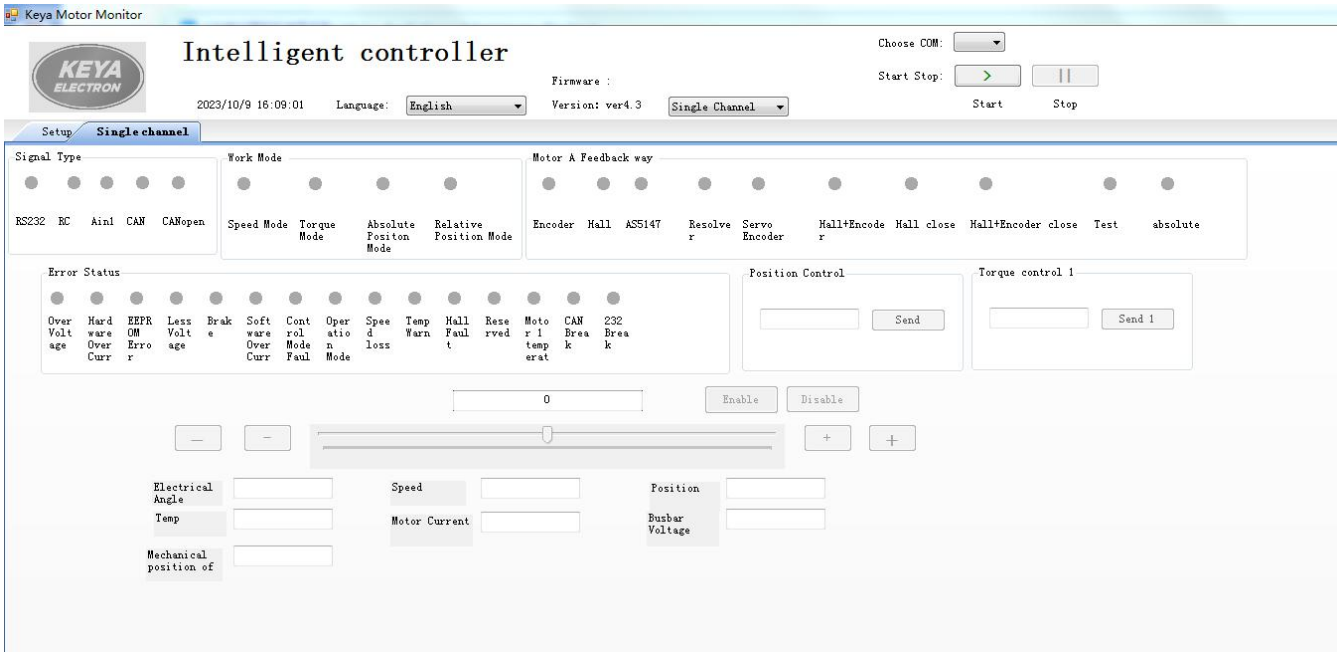
4.1.2 The software instructions

4.1.2.1 Double click the program “kyMotor Control Utility”



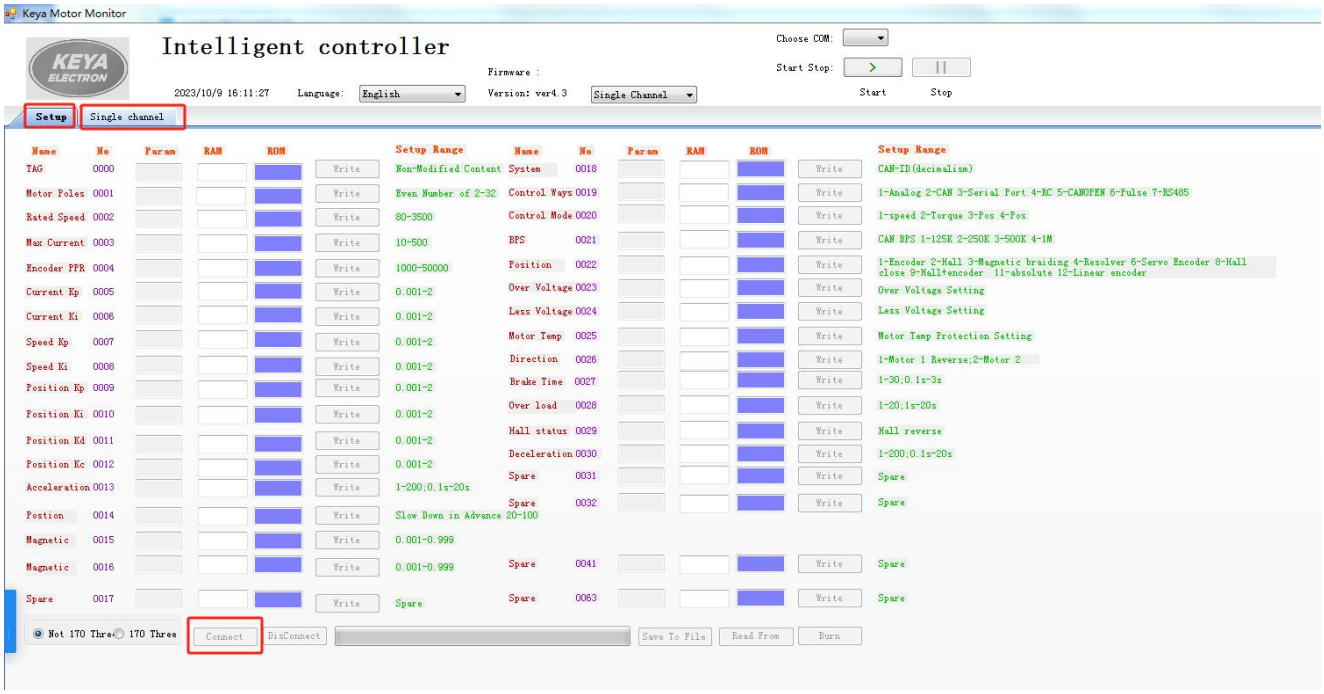
名称	修改日期	类型	大小
Languages	2022/9/16 9:38	文件夹	
CANalyst	2022/10/25 15:29	快捷方式	1 KB
datalog.txt	2023/3/8 13:22	文本文档	1,826 KB
DevComponents.DotNetBar2.dll	2015/10/6 22:27	应用程序扩展	5,344 KB
info.ini	2023/3/8 13:14	配置设置	1 KB
kyMotor Control UtilityV4.3.exe	2022/9/17 10:13	应用程序	500 KB
kyMotor Control UtilityV4.3.exe.Config	2023/3/8 13:14	CONFIG 文件	1 KB
kyMotor Control UtilityV4.3.pdb	2021/9/7 17:09	PDB 文件	378 KB
kyMotor Control UtilityV4.3.vshost.exe	2021/9/7 17:09	应用程序	12 KB
kyMotor Control UtilityV4.3.vshost.ex...	2021/9/6 10:19	CONFIG 文件	1 KB
kyMotor Control UtilityV4.3.vshost.ex...	2012/6/6 2:06	MANIFEST 文件	1 KB

4.1.2.2 Click “Start” button. If the communication is successful, the interface will read the control parameters, while the LED on the upper left will flash green, indicating that the parameters are communicating successfully.

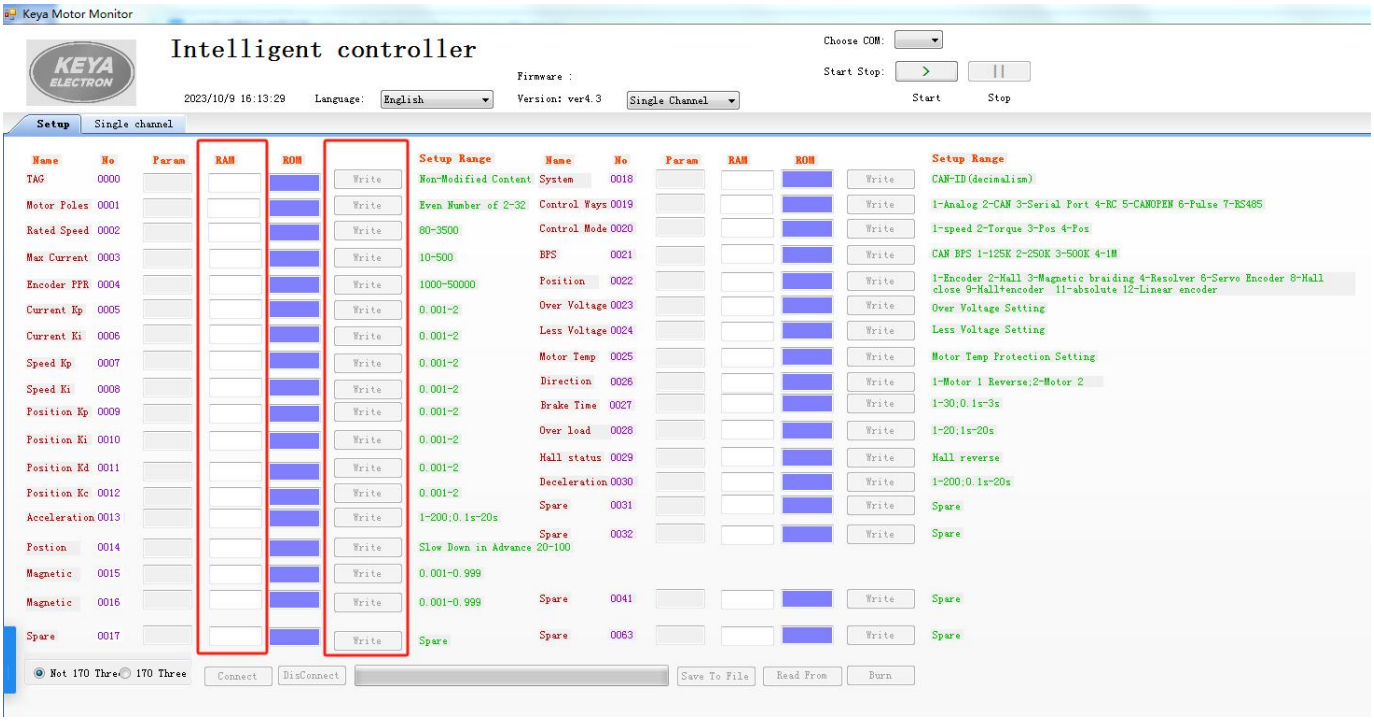


4.1.2.3 Open the “Setup” interface and click the “Connect” button in the lower left corner to set up a connection between the software and the motor driver.

Note: When reading the configurations, it must be done after the motor starts normally, and the configuration can’t be read at the moment the motor is powered on, otherwise the configuration parameters will be reset to the initial value! ! !



4.1.2.4 The RAM in the red box can be modified. The left side of the red box is for the driver parameters, and the right side of the red box is the data in the E^2ROM. In normal cases, the data in same row are consistent (equal). As the software data is continuously scanned, when modifying the data, modify it quickly and click the “Write” button correspondingly.



4.1.2.5 For example, the maximum current needs to be changed, the E^2 ROM data is “20”, need change it to "15". Change “RAM” of 0003 to 15, and click “Write” button quickly. Confirm that “15” is no longer changing. Same steps for other parameters, multiple parameters can be modified at the same time.

4.1.2.6 Click the "Burn" button at the bottom right. Program the data in RAM to E^2ROM. Note: The “Burn” process takes 3 seconds or so.

4.1.2.7 The “Burn” button turns red, indicating that data is being programmed. Please wait and observe the data. Until it prompts “Burned successfully”. Then the data (RAM, ROM, Param) in blue blocks are consistent, indicating that the ROM data is stored into the driver.

4.1.2.8 Until now, the modification of the control parameters is completed. Click the "Disconnect" button and close the software.

4.1.2.9 Electrify the motor driver again. (Note: No matter you modify the configuration or not, you must power off the motor and restart it.)

4.1.2.10 When burning the configuration for multiple motors, you can click “save to file” and then “read from” buttons to download to another motor.

4.1.3 The software instructions Parameter function description

0000 Identifier. when the system is connected, recognize the software communication or serial port control. (No need to modify)

0001 The poles number of motor.

0002 Rated motor speed (The factory setting is 100rpm)

0003 Maximum protection current of motor (Factory setting 20A)

0004 The number of encoder lines

0005 Kp parameter of current loop PI control of driver

0006 Ki parameter of current loop PI control of driver

0007 Kp parameter of speed loop PI control of driver

0008 Ki parameter of speed loop PI control of driver

0009-0012 PID parameter of position loop.

0013 Acceleration time. "50" means the acceleration time from 0rpm to rated speed is 5s.

0015 Zero position compensation of magnetic encoder

0016 Zero position compensation of resolver

0017 Standby

0018 System address of driver, or node number of control. (Decimal number)

This parameter is used in the CAN, CANOpen.

For example: the CAN bus ID: 0x06000000 + system address of driver

0019 Control signal selection

2--CAN 3-- RS232 5-- CANopen

0020 Work mode selection:

1--Speed control

2--Torque control

3--Absolute position control

4--Relative position control

0021 CAN bus baud rate selection (Factory setting 250K)

1--125k 2--250k 3-- 500k

0022 Position sensor selection (Standard setting is Absolute encoder)

3--Single-turn absolute encoder

11--Multiturn absolute encoder

0023 The value of Over voltage protection

0024 The value of Less voltage protection

0025 The value of temperature protection

0026 The motor rotation direction (speed mode, torque mode)

0--Forward 1--reverse

0027 The brake delay braking time. (The motor has no this function.)

“10” means 1 second after receiving the disable signal, the motor will lose power and hold the brake.

0028 Overload delay protection time.

“1” means protection will occur immediately after reaching the protection current for 1s.

“20” means time 20s. If the driver current reaches the protection current, the controller will do overload protection after 20s.

0030 Deceleration time, "50" means the deceleration time from rated speed to 0rpm is 5s;

0033 Zero point of steering motor

Modification is not allowed. Otherwise, the motor performance will be affected.

0034 CAN heartbeat upload time (Standard setting: 20ms)

0-2000 corresponds to 0-2000ms

Other parameters: spare

4.2 Indicator description

4.2.1 Status indicator (Blue light): Observe the status of the driver according to the blinking frequency of the indicator.

No. of flashes	Definition	Possible Reasons
0	Enable status	No fault
1	Disable	No fault,disable status
2	Over voltage	Input voltage is higher than 32V (configurable)
3	Hardware over current protection	Over current protection, it is caused by motor Instantaneous maximum current, short circuit or field tube damage
4	EEPROM error	Enter the Flash configuration program, please power off and restart the driver after the configuration.
5	Less voltage	Input voltage is lower than 7V(configurable)
6	Reserved	Reserved
7	Over current protection (the protection value in software)	The phase current reaches the software setting protection value, then stop output.
8	Control mode fault	Control mode selection error
9	Phase failure	The motor's phase cable disconnected, or the voltage between the motor and the driver doesn't match with each other
10	Stall protection	The actual speed exceeds 25% of the set value
11	Temperature alarm	The driver stop working when the temperature is above 85°C
12	Hall fault	Motor Hall is off or faulty
13	Current sensor fault	The internal current sensor is faulty
14	Spare	Spare
15	CAN disconnected	Under CAN mode, no CAN signal input
16	RS232 disconnected	Under RS232 mode, no 232 signal input

4.2.2 Fault indicator (Red light):

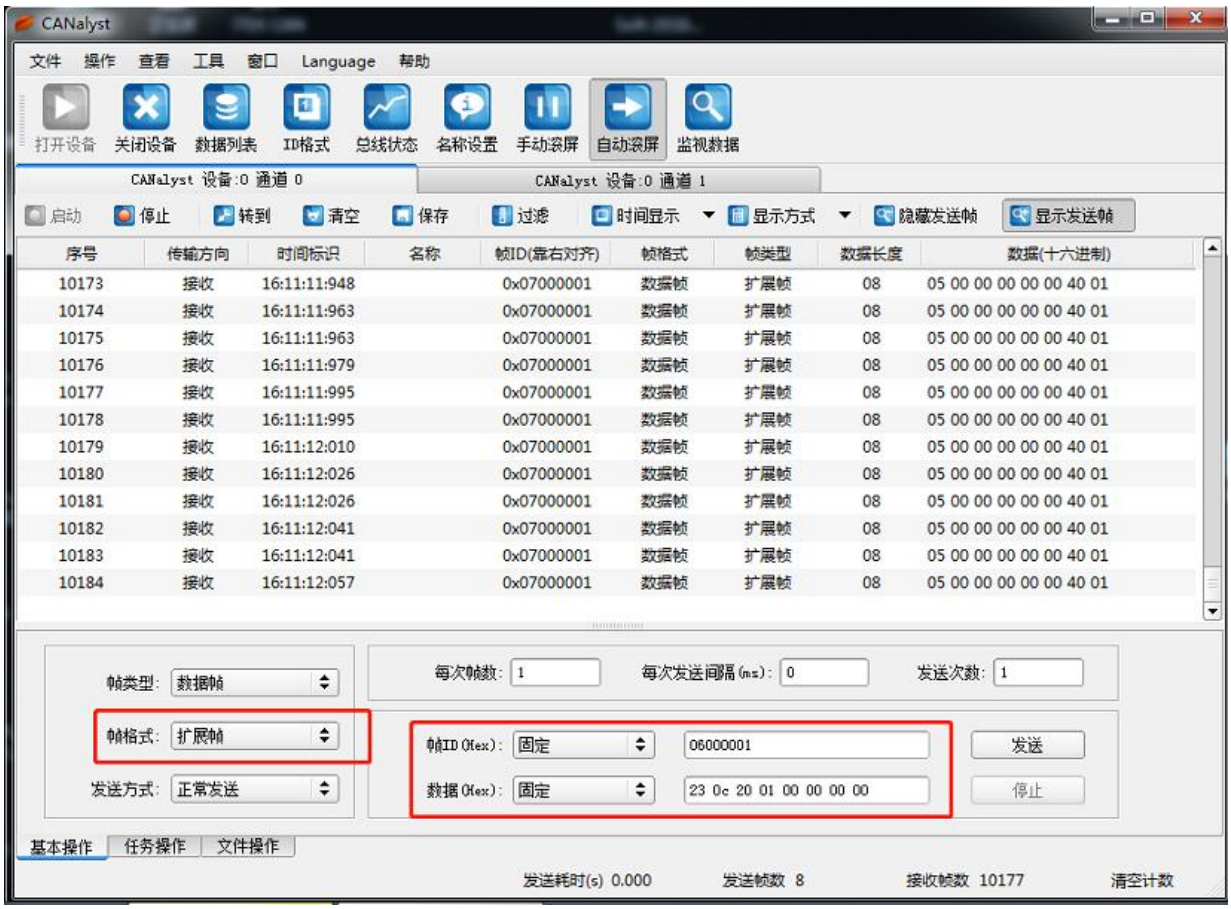
Under any control mode, the red indicator light will always be on in case of any failure.

4.3 CAN2.0 command instructions

4.3.1 General configuration

- CAN bus protocol baud rate 250Kbps

- Frame format: extended frames,hexadecimal
- The watchdog detects the line-off period of 1000ms (speed command is sent continuously, the interval must not exceed 1000ms)
- According to the CANOpen format, the data adopts query mode.
- According to the CANOpen format, there is a fixed heartbeat and send related data (Frequency is 1Hz).
- The ID is 4 bytes and the instruction is 8 bytes.
- Send ID: 0x06000000+ driver address (ID can be configured by the software, factory default is 1)
- Feedback ID: 0x05800000+ driver address
- Heartbeat ID:0x07000000+ driver address
- The returned query data are all hexadecimal, which need to be converted into decimal data in sequence.



4.3.2 Instruction direction

Enable = 0x230D200100000000

Disable = 0x230C200100000000

Speed = 0x2300200100000000

Torque = 0x2301200100000000

Position = 0x2302200100000000

Speed: -10000 -- 10000 corresponds to negative rated speed -- rated speed

Torque: -10000 -- 10000 corresponds to negative rated torque -- rated torque

Position: -2147483648 --- +2147483647 （10000/circle)

Enable: 23 0D 20 01 00 00 00 00

Return ID: 0x05800000 + driver set address

Return Data 60 0D 20 00 00 00 00 00

Disable: 23 0C 20 01 00 00 00 00

Return ID: 0x05800000 + driver set address

Return Data 60 0C 20 00 00 00 00 00

Speed control: 23 00 20 01 DATA_H(h) DATA_H(1) DATA_L(h) DATA_L(1)

Return ID: 0x05800000 + driver set address

Return Data 60 00 20 00 00 00 00 00

Torque control: 23 01 20 01 DATA_H（h） DATA_H（l） DATA_L（h） DATA_L（l）

Return ID: 0x05800000 + driver set address

Return Data 60 01 20 00 00 00 00 00

Position control: 23 02 20 01 DATA_H（h） DATA_H（l） DATA_L（h） DATA_L（l）

Return ID: 0x05800000 + driver set address

Return Data 60 02 20 00 00 00 00 00

Motor current query (A): 40 00 21 01 00 00 00 00

Return ID: 0x05800000 + driver set address

Return Data 60 00 21 01 00 00 00 DATA

Fault query: 40 12 21 01 00 00 00 00

Return ID: 0x05800000 + driver set address

Retun Data 60 12 21 01 DATA-H DATA-L 00 00

Motor speed query: 40 03 21 01 00 00 00 00

Return ID: 0x05800000 + driver set address

Return Data 60 03 21 01 DATA_H（h） DATA_H（l） DATA_L（h） DATA_L（l ）

Rotor mechanical position query(0-9999): 40 04 21 01 00 00 00 00

Return ID: 0x05800000 + driver set address

Return Data 60 04 21 01 00 00 DATA-H DATA-L

Encoder count value query (four bytes): 40 04 21 02 00 00 00 00

Return ID: 0x05800000 + driver set address

Return Data 60 04 21 02 DATA_H（h） DATA_H（l） DATA_L（h） DATA_L（l）

Power supply voltage query (V): 40 0D 21 02 00 00 00 00

Return ID: 0x05800000 + driver set address

Return Data 60 0D 21 02 00 00 00 DATA

Temperature query(℃): 40 0F 21 01 00 00 00 00

Return ID: 0x05800000 + driver set address

Return Data 60 0F 21 01 00 00 DATA 00

Heartbeat return command:

Return ID: 0x07000000 + driver address

Return data:

ELE-H ELE-L speed_H speed_L DATA_H DATA_L Err_H Err_L

Description:

ELE-H ELE-L is Multi-turn position count value (360/turn)

speed_H speed_L is Motor speed (rpm/min; int type)

DATA_H, DATA_L is the working current (A; int type)

Err_H, Err_L is failure code, corresponding to the number of flashes of the indicator.

4.3.3 Example of CAN control

Note: When sending speed control commands and torque control commands, they need to be sent continuously, and the time interval must not exceed 1000ms, otherwise it will be judged as CAN disconnects and reports an error, you need to resend the enable command when starting again.

4.3.3.1 Speed Control:

(Speed command value ‰)x(preset max speed)= actual speed

The system address in the software is set as 1 (Parameter 0018 is set to 1)

The “Control Ways” in the software is set as CAN. (Parameter 0019 is set to 2)

The “Control Mode” in the software is set as speed control (Parameter 0020 is set to 1)

The control command ID: 0x06000001 (extended ID)

Speed setpoint -10000-+10000 stands for -1500rpm-+1500rpm
（0xD8F0）(0x2710）

Eg: If you want the motor to run at 50 rpm(The rated speed in software is 100 rpm)

Enable command: 23 0D 20 01 00 00 00 00

Speed command: 23 00 20 01 00 00 13 88 （0x1388 = 5000）

Eg: If you want the motor to run at -50rpm（The rated speed in software is 100rpm）

Enable: 23 0D 20 01 00 00 00 00

Speed command: 23 00 20 01 FF FF D8 F0 （0xD8F0 = -10000）

4.3.3.2 Torque control:

The limiting current value = (given command value ‰）x (The preset rated current) x 80%

The software sets the system address to 1 (Parameter 0018 is set to 1)

The “Control Ways” in the software is set as CAN (Parameter 0019 is set to 2)

The “Control Mode” in the software is set as torque control (Parameter 0020 is set to 2)

The control command ID：0x06000001 (extended ID)

Command value -10000 — 10000 stands for negative rated torque -- positive rated torque
（0xD8F0） （0x2710）

Eg 1: Rated torque 100% output

Enable:23 0D 20 01 00 00 00 00

Command: 23 01 20 01 00 00 27 10 （0x2710 = 10000)

Under this condition, when the phase current reaches the 80% of rated current, the motor will stop output after 5s.

Eg 2: Rated torque 80% output

Enable 23 0D 20 01 00 00 00 00

Command: 23 01 20 01 FF FF E0 C0 （0xFFFFE0C0 = -8000）

Under this condition, when the phase current reaches the 80%x80% of rated current, the motor will stop output after 5s.

4.3.3.3 Position Control (10000/circle)

Position command -50000—+50000 stands for 5 circles in anticlockwise—5 circles in clockwise
（0xFFFF 3CB0)（0xC350 ）

The “Control Ways” in the software is CAN control (Parameter 0019 is set to 2)

The “Control Mode” in the software is set as Absolute position control (Parameter 0020 is set to 3)

Or The “Control Mode” is set as Relative position control (Parameter 0020 is set to 4)

The software sets the system address to 1 (Parameter 0018 is set to 1)

The control command ID：0x0600 0001 (extended ID）

Data sending sequence:

(a) Enable: 23 0D 20 01 00 00 00 00

(b)Position control: 23 02 20 01 DATA_H（H） DATA_H（L） DATA_L（H）DATA_L（L ）

Eg1: Make the motor rotate 1.8 circles in clockwise

(a) Enable: 23 0D 20 01 00 00 00 00

(b) Position control command: 23 02 20 01 FF FF B9 B0

Eg2: Make the motor rotate 72 degrees counterclockwise

（72 *(10000/360) = 2000 = 0x7D0）

(a) Enable: 23 0D 20 01 00 00 00 00

(b) Position control command: 23 02 20 01 00 00 07 D0

4.3.3.4 Mechanical zero calibration procedures

①. Straighten the wheel without connecting the battery. Set parameter 16 to 0.0 in the software; set parameter 17 to 0.0.

②. Adjust the mechanical position, power on the driver again, and read a certain value of the encoder position value (0-9999). (The software displays interface "position")

③. After converting the position value, set parameter 16 of the software, read a certain value of the current position value (0-9999), and write the corresponding parameter 16 (0-9.999) of the driver.

For example: feedback 8667, parameter 16 = 8.667;

④. After connecting the multi-turn encoder, cut off the power and restart it.

4.4 Serial port command instructions

4.4.1 General Configuration

The serial port communication port of the driver is set as follows:

115200bits/s

8-bit data

1 start bit

1 stop bit

No parity

HEX format

4.4.2 Control format

E0	data0	00	00	00	00	00	00
				H		L	
				Hexadecimal（Ten-thousandth ratio）			

E0 : Control command

data0: 01 ------ Enable

00 ------Disable

- Speed mode:

Speed mode: -10000 ---- +10000 corresponds to negative rated speed --- positive rated speed

Enable: E0 01 00 00 00 00 00 00

Send forward speed control 10%: E0 01 00 00 00 00 03 E8

Send reverse speed control 10%: E0 01 00 00 FF FF FC 18

Disable: E0 00 00 00 00 00 00 00

- Torque mode:

Torque mode:-10000 ---- +10000 corresponds to negative rated torque--- positive rated torque

Enable: E0 01 00 00 00 00 00 00

Send torque command 80%：E0 01 00 00 00 1F 40 (0x1f40 = 8000)

Send torque command -60%：E0 01 00 FF FF E8 90 (0xFFFE890 = -6000)

Disable: E0 00 00 00 00 00 00 00

- Position mode: -2147483648----- +2147483647 (10000/circle)

Make the motor run 5 circles in anticlockwise

Enable: E0 01 00 00 00 00 00 00

Position command: E0 01 00 00 00 00 c3 50 （0x0000C350 = 50000）

Make the motor run 3 circles in clockwise

Enable: E0 01 00 00 00 00 00 00

Position command: E0 01 00 00 FF FF 8A D0 （0xFFFF8AD0 = -30000）

4.4.3 Query format

ED	Data1	00	00	00	00	00	00

ED: Means query command

data1: 00 control status

01--------electric angle (internal data)

02--------Speed (RPM)

03--------Current (A)

04--------The rotor mechanical position (0-9999)

05--------Voltage (V)

06--------Temperature (℃)

07--------Fault code

08--------Position （10000/circle）

09--------Program version

0A--------Encoder fault query

- Query the current motor speed（rpm）

Send: ED 02 00 00 00 00 00 00

Feedback: ED 02 00 64 00 00 00 00 00 00 00 00

I.e. the current motor speed is 100rpm

- Query the current motor Amps（A）

Send: ED 03 00 00 00 00 00 00

Feedback: ED 03 00 08 00 00 00 00 00 00 00 00

I.e. the current motor Amps is 8A

- Query the rotor mechanical position (0-9999)

Send: ED 04 00 00 00 00 00 00

Feedback:ED 04 1B 1F 00 00 00 00 00 00 00 00

I.e. the rotor mechanical position is 6943（10000/circle）

- Query the input voltage of driver（V）

Send: ED 05 00 00 00 00 00 00

Feedback: ED 05 0C 00 00 00 00 00 00 00 00

I.e. the input voltage of driver is 12V

- Query the controller temperature（℃）

Send: ED 06 00 00 00 00 00 00

Feedback: ED 06 00 1A 00 00 00 00 00 00 00 00

I.e. the controller temperature is 26℃

- Query the cumulative position value (10000/circle) (data can be positive or negative)

Send: ED 08 00 00 00 00 00 00

Feedback: ED 08 00 01 86 A2 00 00 00 00 00 00

I.e. the current position is 100002

- Query the driver status

Send: ED 00 00 00 00 00 00 00

Feedback: ED 00 31 20 00 00 00 00 00 00 00 00

I.e. the driver status is 3：RS232; 1: Encoder; 2: Torque mode

Eg:

Send: ED 00 00 00 00 00 00 00 00 00 00 00

Feedback: ED 00 28 40 00 00

"2" control mode: 1.Analog signal 2.CAN 3.RS232 4.RC

"8" feedback way: 1.Encoder 2.Hall 3.AS5147 4.Rotary transformer 6. motor encoder
7.Hall + Encoder 8.Hall close loop 9.Hall+encoder close loop 10.test
11.Absolute value

"4" Working mode: 1.Speed mode 2.Torque mode 3.Absolute position mode 4.Relative position mode

Therefore, the current status of driver is 2-CAN, 8-Hall close loop, 4-Relative position mode

- Query the fault code

Send: ED 07 00 00 00 00 00 00

Feedback: ED 07 00 10 00 00 00 00 00 00 00 00

I.e. the current fault code is 5，it is less voltage.

- Fault code analysis

At first, convert the hexadecimal to binary, then check the "1" position from the right to the left, which corresponds to the indicator blinking frequency.

Eg: The feedback data is 03 01

0 3 0 1

0000 0011 0000 0001 ; corresponds to 1, 9, 10 three faults.

Refer to Indicator Description table for details.

- Encoder fault query

Send: ED 0A 00 00 00 00 00 00

Feedback: ED 0A A0 00 00 00 00 00 00 00 00

I.e. the fault is: error bit, no battery;

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Error bit	Warning bit	No battery	Battery less voltage	Too strong magnetic field	Too weak magnetic field	Over temperature	Over speed

4.4.4 Serial heartbeat data

After successful communication, the status of driver will be transferred automatically. Start bit 0xEE, the data definition is as below: (hexadecimal)

Dat0	Dat1	Dat2	Dat3	Dat4	Dat5	Dat6	Dat7	Dat8	Dat9	Dat10	Dat11	Dat12
EE	Electrical angle		Fault code		TEMP	VOLT	Speed		Position			

Note: The returned data are in hexadecimal, should be converted to decimal.

V. Fault protection and reset

5.1 Failure protection basis

5.1.1 Temperature alarm

When the temperature of the driver exceeds 85 °C, a temperature alarm occurs. When it is restored to 75 °C, the alarm is cleared automatically.

5.1.2 Overcurrent protection

When the phase current reaches the set current value, it stops for the set overload time. Reset it by clicking “Enable”.

5.1.3 Overvoltage and undervoltage protection

The system will make undervoltage protection when the power supply voltage is lower than the set undervoltage protection value. After the voltage recovers, the fault automatically disappears;

The system will make undervoltage protection when the power supply voltage is higher than the set undervoltage protection value. After the voltage recovers, the fault automatically disappears;

5.2 Communication faults information table(same as indicator state)

Fault bit	Definition	Cause of issue	Recovery mode
Bit0	Enable, Disable	0- Enable 1- Disable	
Bit1	Over voltage	Input voltage is higher than 32V (configurable)	self-reset
Bit2	Hardware over current protection	Over current protection, it is caused by motor Instantaneous maximum current,	Power off and reset

		short circuit or field tube damage	
Bit3	EEPROM error	Enter the Flash configuration program, please power off and restart the motor driver after the configuration.	Power off and reset
Bit4	Less voltage	Input voltage is lower than 7V (Configurable)	Disabling reset
Bit5	Reserved	Reserved	
Bit6	Over current protection (the protection value in software)	The phase current reaches the software setting protection value and stop output.	Disabling reset
Bit7	Control mode failure	Control mode selection error	Power off and reset
Bit8	Phase failure	The motor's phase cable disconnected, or the voltage between the motor and the driver doesn't match with each other	Power off and reset
Bit9	Stall protection	The actual speed exceeds 25% of the set value;	Disabling reset
Bit10	Temperature alarm	The driver stops output when the temperature is above 85°C	self-reset
Bit11	Hall failure	Motor Hall is off or faulty	Disabling reset
Bit12	Current sensor failure	The internal current sensor is faulty	Power off and reset
Bit13	Spare	Spare	
Bit14	CAN disconnected	Under CAN mode, no CAN signal input	Enable reset
Bit15	RS232 disconnected	Under RS232 mode, no 232 signal input	Enable reset

VI. CANopen User Manual

6.1. Introduction to Communication Protocols

CANopen is one of the most famous and successful open fieldbus standards, which has been widely recognized and widely used in Europe and the United States. In 1992, CiA (CANinAutomation) was established in Germany and began to develop CANopen, an application layer protocol of automatic CAN. Since then, members of the Association have developed a series of CANopen products, which are widely used in machinery manufacturing, pharmaceutical, food processing and other fields.

DAS series servos are standard CAN slave devices, strictly follow CANopen2.0A/B protocol, and any host computer supporting this protocol CAN communicate with them. DAS series servos use a strictly defined list of objects, called the object dictionary. This object dictionary is designed based on the CANopen international standard, and all objects have clear functional definitions. Some Objects, such as speed and position, can be modified by the external controller, while some Objects can only be modified by the drive itself, such as state and error messages. These objects are hexadecimal numbers. For example, the CANopen address in working mode is 0x60400010, as shown in Table 1-1.

Table 1-1 List of sample object dictionaries

Complete CANopen address composition			Attribute	Meaning
Index	Sub-index	Bits (data length)		
0x6040	00	0x10	RW	Device state control byte
0x6060	00	0x08	RW	Operating modes
0x6041	00	0x10	MW	Device state byte

Objects have the following attributes:

(1) RW (read and write): Objects can be read or written;

(2) RO (read only): An object can only be read;

(3) WO (write only): Can only be written;

(4) M (mappable): Object mappable, similar to indirection;

(5) S (storable): Objects can be stored in the Flash-ROM area and will not be lost during power failure.

6.2. Hardware Description

CAN communication protocol mainly describes the way of information transfer between devices. The definition of CAN layer is consistent with the OSI model of open system interconnection. Each layer communicates with the same layer on another device. The specification of CAN defines the lowest two layers of the model, the data link layer and the physical layer. There are no strict regulations on the physical layer of CAN bus, which CAN use a variety of physical media, such as twisted pair fiber, etc. The most commonly used is the twisted pair signal, which is transmitted by differential voltage (common bus transceiver). The two signal lines are called CAN H and CAN L, which are both about 2.5V at static state, and the state is represented as logic 1. CAN H is higher than CAN L to represent the logic 0, which is called the explicit position. At this time, the voltage value is usually CAN H=3.5V and CAN L=1.5V, and the explicit position takes precedence in competition.

Caution

- The CAN L and CAN H pins of all slave stations CAN be directly connected, using serial connection mode, not star-shaped connection mode;
- The master station and the last slave station need to be connected to a terminal resistor of 120 ohm, the driver is external, not internal.
- Table 1-2 lists the maximum distance that each baud rate can communicate with.

Table 1-2 Maximum distance of each baud rate for communication

Speed of communication (bit/s)	Braking distance (m)
1M	25
500K	100
250K	250

125K	500
50K	600

6.3 Software Description

6.3.1 EDS Description

EDS (electronic data sheet) file is the identification file or similar code of the slave station connected to PLC, through which to identify the type of the slave station (is 401, 402, 403 similar, or belongs to the 402 of the device). This file contains all the information of the slave station, such as the manufacturer, serial number, software version, supported baud rate types, the OD that can be mapped and the attributes of each OD, etc., similar to the GSD file of Profibus. Therefore, before hardware configuration, we first need to import the EDS file from the station into the upper configuration software (please contact the business personnel when purchasing the EDS file).

6.3.2 SDO Description

SDO is mainly used to transfer low-priority objects between devices. Typically, it is used to configure and manage slave devices, such as modifying PID parameters of current loop, speed loop, position loop, PDO configuration parameters, etc. Such data transmission is the same as MODBUS, that is, after the master station sends data, the slave station needs to return data response. This communication mode is only suitable for parameter setting, not suitable for data transmission which requires high real-time performance. The communication mode of SDO is divided into upload and download. The upper computer can read and write the OD inside the servo according to the special SDO read and write instruction.

In CANopen protocol, the contents of the Object dictionary can be modified through Service Data Object (SDO). The following describes the structure of the SDO command and the guidelines it follows.

The basic structure of SDO is as follows: Client→Server/Server→Client

Data transmission format

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x600+Node_ID	8	Send command byte	Object index		Object sub-index	Max 4-byte data			

Data return format

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x580+Node_ID	8	Receive command byte	Object index		Object sub-index	0			

Caution:

When the SDO message reads node parameters, the command word is 0x40.

If the received data is 1 byte, the received command word is 0x4F. If the received data is 2 bytes, the received command word is 0x4B. If the received data is 4 bytes, the received command word is 0x43.

If there is an error in receiving data, the command word is 0x80.

Read node command Data_0	Data_1. 2	Data_3	Data_4	Data_5	Data_6	Data_7
0x40	Indexes	Sub index	00	00	00	00

Send SDO packets when modifying parameters

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x600+Node_ID	8	Send command byte	Object index		Object sub index	Max 4 bytes			

Receive SDO when changing parameters

Identifier	DLC	Data							
		0	1	2	3	4	5	6	7
0x580+Node_ID	8	Receive command byte	Object index		Object sub index	0			

Caution:

If the data to be sent is 1 byte while writing node parameter , the command word is 0x2F. If the data to be sent is two bytes, the command word is 0x2B. If the data to be sent is 4 bytes, the command word is 0x23. The SDO message is successfully sent, and the received command word is 0x60. If sending SDO packet fails, the received command word is 0x80.

For multi-byte indexes and data, low bytes are always be in front.

Write byte command Data_0	Data_1. 2	Data_3	Data_4	Data_5	Data_6	Data_7
0x2F	Indexes	Sub index	xx	00	00	00
0x2B	Indexes	Sub index	xx	xx	00	00
0x27	Indexes	Sub index	xx	xx	xx	00
0x23	Indexes	Sub index	xx	xx	xx	xx

6.3.3 PDO Description

PDO can transmit up to 8 bytes of data at a time, with no other protocol Settings (meaning the data content is predefined), and is mainly used to transmit data that needs to be exchanged at high frequencies. PDO transport broke the existing data transmission and concept, adopts new data exchange mode, both sides before transmission equipment in each area, sending and receiving equipment defined data in the data exchange direct send the data related to each other's data reception area, reduce the time of the question-and-answer inquiries, so as to greatly improve the efficiency of bus communication, This results in extremely high bus utilization.

6.3.3.1 PDO COB-ID Description

Cob-id is a unique mode of CANopen Communication Protocol. Its full name is Communication Object Identifier - Communication Object ID. These COB-IDS define the corresponding transport levels for PDO. The controller and the servo can define the same transmission level and its transmission content in their own software configuration, so that the controller and the servo adopt the same transmission level and transmission content, the data transmission is transparent, that is, both sides know the data content to be transmitted. Therefore, there is no need for the other party to reply whether the data is successfully transmitted when transmitting data.

The Default ID Allocation Table is based on the 11-bit CAN-ID defined by CANopen 2.0A (CANopen 2.0B Protocol COB-ID is 29 bits), including a 4-bit function code part and a 7-bit node-ID part, as shown in Figure 8-13.

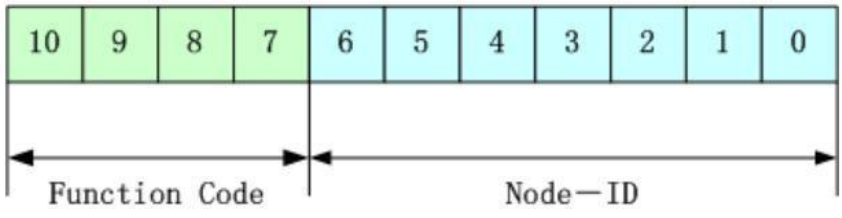


Figure 11-2 Default ID Description

Caution

Node-ID — The node-ID ranges from 1 to 127.

Function Code — Data transmission function code, defines transmission levels of various PDO, SDO, and management packets. A smaller function code has a higher priority.

Table 11-4 CANopen Pre-defined Master/Slave Connector Set CAN Identifier Assignment Table

Object	COB-ID
NMT Module Control	000H
SYNC	080H
TIME SSTAMP	100H
Object	COB-ID
Urgent	081H-0FFH
PDO1(send)	181H-1FFH
PDO1(receive)	201H-27FH
PDO2(send)	281H-2FFH
PDO2(receive)	301H-37FH
PDO3(receive)	381H-3FFH
PDO3(receive)	401H-47FH
PDO4(send)	481H-4FFH
PDO4(receive)	501H-57FH
SDO (send/server)	581H-5FFH
SDO (receive/client)	601H-67FH
NMT Error Control	701H-77FH

6.3.3.2 COB-ID

Sending PDO as opposed to servo refers to the data sent by the servo, which is received by the PLC. Sending the function code of PDO (COB-ID) is:

1. 0x180+servo station number
2. 0x280+servo station number
3. 0x380+servo station number
4. 0x480+servo station number

Relative to servo, receiving PDO refers to the data received by the servo, which is sent by PLC and sends the function code of PDO (COB-ID) is:

1. 0x200+servo station number
2. 0x300+servo station number
3. 0x400+servo station number
4. 0x500+servo station number

6.3.3.3 Type of PDO Transmission

PDO has two modes of transmission:

SYNC -- Transmission triggered by synchronous messages (transmission type: 0-240)

In this transmission mode, the controller must have the ability to send synchronous messages (messages with a frequency of up to 1KHZ). The servo sends synchronous messages after receiving them.

Aperiodic - transport is pre-triggered by remote frames or by object specific events specified in the device sub-protocol. In this mode, the servo drive sends the data in the PDO once it receives a synchronous message.

Cycle - Delivery is triggered every 1 to 240 SYNC messages. In this mode, the data in the PDO is sent once every n synchronization packets received by the servo drive.

Asynchronous (Transfer type: 254/255)

The secondary station sends packets after they are changed, regardless of whether the primary station asks for them or not. In addition, the interval between sending the same packet can be defined to prevent packets of high priority from occupying the bus all the time. (The lower the PDO value is, the higher the priority is.)

DAS series servo drivers support asynchronous transmission mode.

Caution

A PDO can specify a stop time, that is, define the minimum interval between two consecutive PDOS to avoid the problem that the data volume of high-priority information is so large that it always occupies the bus, and other low-priority data cannot compete with the bus. Disallow time from 16 bits without character

The integer is defined in 1ms.

6.3.3.4 Protection mode/Supervision type

The Supervision Type refers to the inspection mode selected by the master station to check the slave station during operation. Through these two ways to judge whether the slave station is faulty, slave station will be processed according to the faults.

1.The heartbeat packets

The secondary site periodically sends packets to the primary site according to Heartbeat Packet Generation Time. If the primary site does not receive the next heartbeat packet from the secondary site after a certain period (set on the primary site), the primary site fails to judge the secondary site.

Packet format -- (0x700+ Node NUMBER) + state

State -- 0: start, 4: stop, 5: running, 127: pre-operation

2. Node protection

The master station periodically sends packets to the slave station by monitoring time. If the slave station does not receive any node packets from the master station after the monitoring time * life factor expires, the slave station alarms!

Format of the primary site request packet --

(0x700+ Node ID) (No data is contained in this

packet) Format of the secondary site response

packet -- (0x700+ Node ID) + state:

State -- The data portion includes a trigger bit (bit7) that must be alternately 0 or 1 in each node protection response. The trigger bit is set to 0 on the first node protection request. Bits 0 to 6 (bit0 ~ 6) indicate node state. 0: initialization, 1: link down, 2: link up, 3: operation, 4: stop, 5: running, 127: pre-operation.

Standard CAN slave stations generally support only one node protection mode. DAS series servo drivers use heartbeat message detection.

6.3.3.5 Startup Process description

CANopen supports both extended boot-up and minimal boot-up during network initialization. This initialization process can be represented by a node state transition diagram, as shown in Figure 11-3

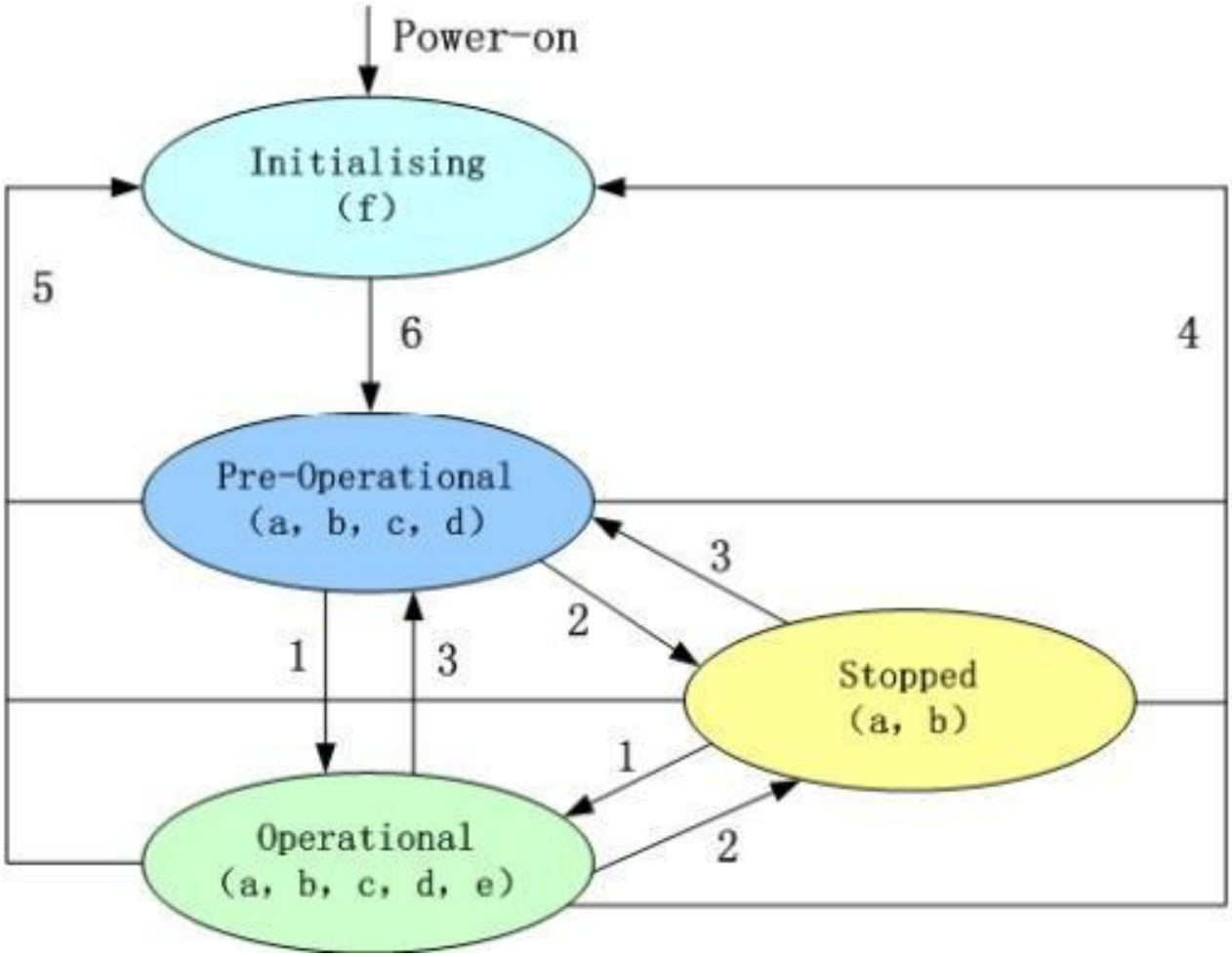


Figure 11-3 Node state conversion diagram

Table 11-5 CANopen network state

Code	Meaning
a	NMT
b	Node Guard
c	SDO
d	Emergency
e	PDO
F	Boot-up

Managing message Formats

COB-ID	DLC	Byte0	Byte1
0x000	02	CS	Station number

When node-id =0, all NMT slave devices are addressed. CS is a command word. Table 1-11 lists its values.

Table 11-6 Table of CS values

Command	NMT Service
0x01	Start the node to start PDO transmission
0x02	Shut down the node and shut down the PDO transmission
0x80	The system enters the pre-operation state
0x81	Reset the node
0x82	Reset the communication

6.4. Object Dictionary Description

The CANopen dictionary shown in this section is subject to change. CANopen EDS files can be contacted by business people.

The object dictionary in the following table maps the EDS file of the DUAL-drive standard DAS series. It also contains runtime queries and runtime instructions.

Note: Motor B parameter is meaningless when using single channel driver!

COB-ID	Indexes	Sub index	Parameter name	Access attributes	Data type	Variant name	Remark
	0x1000	0	Type of device	RO	Uint32	Func.obj1000	
	0x1001	0	Error register	RO	Uint8	Func.obj1001	
	0x1003	0	Predefined error area	RW	Uint8	Func.highestSubIndex_obj1003	
		1		RO	Uint32	Func.obj1003[0]	
		2		RO	Uint32	Func.obj1003[1]	
		3		RO	Uint32	Func.obj1003[2]	
		4		RO	Uint32	Func.obj1003[3]	
	0x1005	0	Synchronization frame ID	RW	Uint32	SYNC COB-ID	
	0x1009	0	Hardware version	RO	Uint16	Func.obj1009	
	0x100A	0	Firmware version	RO	Uint16	Func.obj100A	
	0x1014	0	Urgent COB-ID	RW	Uint32	Emergency COB ID	
	0x1017	0	Heartbeat time	RWS	Uint32	Func.obj1014	
	0x1200	0	Slave Station SDO Parameters	RO	Uint8	Func.highestSubIndex_obj1200	
0x600 + Node ID			Slave Station Receiving ID	RO	Uint64	COB_ID_Client_to_Server_Receive_SDO	
0x580+_ Node ID			Slave Station Sending ID	RO	Uint64	COB_ID_Server_to_Client_Transmit_SDO	

6.4.1 RPDO Parameters

COB-ID	Indexes	Sub index	Parameter Name	Access attributes	Data type	Remark
0x600+id	0x1400	0	RPDO1 Parameters	RO	Uint8	Number of Entries
		1	COB-ID	RW	Uint32	0x200 + id
		2	Type of Transmission	RWS	Uint8	255
		3	Disallow Time	RWS	Uint16	0
		4	Compatibility (Reserved)	RW	Uint8	
		5	Processing Time	RWS	Uint16	1000
	0x1401		RPDO2 Parameters	RO	Uint8	Number of Entries
		1	COB-ID	RW	Uint32	0x300 + id
		2	Type of Transmission	RWS	Uint8	255
		3	Suspension Time	RWS	Uint16	0
		4	High compatibility	RW	Uint8	
		5	Processing Time	RWS	Uint16	1000
	0x1402	0	RPDO3 Parameters	RO	Uint8	Number of Entries
		1	COB-ID	RW	Uint32	0x400 + id
		2	Type of Transmission	RWS	Uint8	255
		3	Disallow Time	RWS	Uint16	0
		4	Compatibility (Reserved)	RW	Uint8	
		5	Processing Time	RWS	Uint16	1000
	0x1403		RPDO4 Parameters	RO	Uint8	Number of Entries
		1	COB-ID	RW	Uint32	0x500 + id
		2	Type of Transmission	RWS	Uint8	255
		3	Disallow Time	RWS	Uint16	0

		4	Compatibility (Reserved)	RW	Uint8	
		5	Processing Time	RWS	Uint16	1000

6.4.2 RPDO Mapping (factory setting)

COB-ID	Indexes	Sub index	Parameter Name	Access attributes	Data type	Remark
0x200 + id	0x1600	0	Number of Objects for RPDO1 Mapping	RW	Uint8	
		1	Map the first object	RWS	Uint32	**607A(A enabled)**
		2	Map the second object	RWS	Uint32	**6081(A data)**
		3	Map the third object	RWS	Uint32	
		4	Map the fourth object	RWS	Uint32	
0x300 + id	0x1601	0	Number of RPDO2 Mapped Object	RW	Uint8	
		1	Map the first object	RWS	Uint32	**6083(B enabled)**
		2	Map the second object	RWS	Uint32	**6084(B data)**
		3	Map the third object	RWS	Uint32	
		4	Map the fourth object	RWS	Uint32	
0x400 + id	0x1602	0	RPDO3 Mapping	RW	Uint8	
		1	Map the first object	RWS	Uint32	
		2	Map the second object	RWS	Uint32	
		3	Map the third object	RWS	Uint32	
		4	Map the fourth object	RWS	Uint32	
0x500 + id	0x1603	0	Number of RPDO4 Mapped Objects	RW	Uint8	
		1	Map the first object	RWS	Uint32	
		2	Map the second object	RWS	Uint32	
		3	Map the third object	RWS	Uint32	
		4	Map the fourth object	RWS	Uint32	

6.4.3 TRDO Parameters

COB-ID	Indexes	Subindex	Parameter Name	Data/read & write	Data type	Remark
0x180+id	0x1800	0	TPDO1 Parameters	RO	Uint8	Number of Entries
		1	COB-ID	RW	Uint32	180 + Nodes
		2	Type of Transmission	RWS	Uint8	255
		3	Disallow Time	RWS	Uint16	0
		4	Compatibility (Reserved)	RW	Uint8	
		5	Processing Time	RWS	Uint16	1000
0x280+id	0x1801		TPDO2 Parameters	RO	Uint8	Number of Entries
		1	COB-ID	RW	Uint32	280 + Nodes
		2	Type of Transmission	RWS	Uint8	255
		3	Disallow Time	RWS	Uint16	0
		4	Compatibility (Reserved)	RW	Uint8	
		5	Processing Time	RWS	Uint16	1011
0x380+id	0x1802	0	TPDO3 Parameters	RO	Uint8	Number of Entries
		1	COB-ID	RW	Uint32	380 + Nodes
		2	Type of Transmission	RWS	Uint8	255
		3	Disallow Time	RWS	Uint16	0
		4	Compatibility (Reserved)	RW	Uint8	
		5	Processing Time	RWS	Uint16	#1022
0x480+id	0x1803		TPDO4 Parameters	RO	Uint8	Number of Entries
		1	COB-ID	RW	Uint32	
		2	Type of Transmission	RWS	Uint8	
		3	Disallow Time	RWS	Uint16	
		4	Compatibility (Reserved)	RW	Uint8	
		5	Processing Time	RWS	Uint16	

6.4.4 TPDO Mapping

COB-ID	Indexes	Sub index	Parameter Name	Data/read & write	Data type	Remark
	0x1A00	0	TPDO1 Mapping	RW	Uint8	
		1	Map the first object	RWS	Int16	**2012(Rotation speed A)**
		2	Map the second object	RWS	Int16	**2010(Current A)**
		3	Map the third object	RWS	Int32	**2016(Position A)**
		4	Map the fourth object	RWS		
	0x1A01	0	TPDO2 Mapping	RW	Uint8	
		1	Map the first object	RWS	Int32	**2022(Rotation speed B)**
		2	Map the second object	RWS	Int8	**2020(Current B)**
		3	Map the third object	RWS	Int32	**2026(Position B)**
		4	Map the fourth object	RWS	Uint32	
	0x1A02	0	TPDO3 Mapping	RW	Uint8	
		1	Map the first object	RWS	Uint16	**2011(Fault A)**
		2	Map the second object	RWS	Uint16	**2021(Fault B)**
		3	Map the third object	RWS	Uint16	**2013(voltage)**
		4	Map the fourth object	RWS	Uint16	**203C(temperature)**
	0x1A03	0	TPDO4 Mapping	RW	Uint8	
		1	Map the first object	RWS	Uint32	**2003(control state). See P49 for more details**
		2	Map the second object	RWS	Uint32	**2004(control state)**
		3	Map the third object	RWS	Uint32	
		4	Map the fourth object	RWS	Uint32	

6.4.5 Function Code

	Indexes	Sub index	Parameter Name	Access attributes	Data type	Variant name	Remark
Monitored parameters:							
	0x2000	0	Substation node number	RO	Uint16	Func.Monitor.Slavenodes	
Motor A							
	0x2003	0	Motor A—control state	RO	Uint16	InStateGroup1	
	0x2010	0	Motor A—current effective value	RO	Uint16	MotorA_Current	A
	0x2011	0	Motor A—fault state bit	RO	Uint16	MotorA_Close	
	0x2012	0	Motor A—rotation speed	RO	int16	MotorA_Speed	rpm
	0x2013	0	A Driver DC bus voltage	RO	Uint16	MotorA_Voltage	V
None	0x2014	0	Motor A——Temp	RO	Uint16	MotorA_Temp	℃
	0x2015	0	Motor A——mechanical angle	RO	Uint16	MotorA_QepRewTeta	0-9999
	0x2016	0	Motor A——Position feedback	RO	int32	MotorA_Position	10,000
	0x2017	0	Motor A——electrical angle	RO	Uint16	MotorA_ElecTheta	Internal parameters
Motor B							
	0x2004	0	Motor B—control state	RO	Uint16	OutStateGroup1	
	0x2020	0	Motor B——current effective value	RO	Uint16	MotorB_Current	Unit A
	0x2021	0	Motor B——Fault state bit	RO	Uint16	MotorB_Close	Bit
	0x2022	0	Motor B——Speed	RO	int16	MotorB_Speed	rpm
	0x2023	0	B Driver DC bus voltage	RO	Uint16	MotorB_Voltage	V
None	0x2024	0	Motor B——Temp	RO	Uint16	MotorB_Temp	℃
	0x2025	0	Motor B——mechanical angle	RO	Uint16	Motor B_QepRewTeta	0-9999
	0x2026	0	Motor B—Position feedback	RO	int32	Motor B_Position	10,000
	0x2027	0	Motor B—Electrical angle	RO	Uint16	MotorB_ElecTheta	Internal parameters

6.4.6 System Code

	Indexes	Sub index	Parameter Name	Access attributes	Data type	Variant name	Remark
Control data							
	0x607A		Enable Motor A Maintain control mode	RW	Uint32	Enable motor-A	0x030D2001(enabled) 0x030C2001(disabled)
	0x607A		Motor A Speed mode is enabled	RW	Uint32	Enable motor-A	0x030D2011(enabled) 0x030C2001(disabled)
	0x607A		Motor A absolute position mode is enabled	RW	Uint32	Enable motor-A	0x030D2031(enabled) 0x030C2001(disabled)
	0x607A		The relative position mode of motor A is enabled	RW	Uint32	Enable motor-A	0x030D2041(enabled) 0x030C2001(disabled)
	0x6081		Motor A action data	RW	int32	The data of motor-A	-10000~10000
	0x6099	01	Motor A speed limit (position mode)	RW	Uint32	Homing speed	0~10000
System parameters							
	0x203A		Control working condition	RO	Uint16	MotorState	
	0x203B		System failure state	RO	Uint16	MotorClose	Merging two-way faults
	0x203C		Driver temperature	RO	Uint16	Driver_Temp	
	0x203D		Driver software version	RO	int32	MotorVersion	

6.5 SDO Use Instruction

A enabled	0x030D2001	51191809(decimal)
A disabled	0x030C2001	51126273(decimal)
B enabled	0x030D2002	51191810(decimal)
B disabled	0x030C2002	51126274(decimal)

SDO test Driver ID：2

CANopen initiated: ID 0000 01 02

Motor A enabled

Send: 0x602 23 7A 60 00 01 20 0D 03 (0x030D2001, lower first, higher next)

Feedback: 0x582 60 7A 60 00 00 00 00 00

Motor A disabled

Send: 0x602 23 7A 60 00 01 20 0C 03 (0x030C2001, lower first, higher next)

Feedback: 0x582 60 7A 60 00 00 00 00 00

Motor A speed 50%

Send: 0x602 23 81 60 00 88 13 00 00 (5000=0x1388, lower first, higher next)

Feedback: 0x582 60 81 60 00 00 00 00 00

Index of motor running status:

Temperature: 0x203C Bus voltage: 0x2013 (lower first, higher next)

(Motor A: Speed 0x2012 Phase current 0x2010 Encoder count value (position) 0x2016 Fault 0x2011)

(Motor B: Speed 0x2022 Phase current 0x2020 Encoder count value (position) 0x2026 Fault 0x2021)

Read motor speed:

Send: 0x602 40 12 20 00 00 00 00 00

Feedback: 0x582 4B 12 20 00 96 00 00 00 (0x96 motor speed 150RPM)

Read driver temperature:

Send: 0x602 40 3C 20 00 00 00 00 00

Feedback: 0x582 4B 3C 20 00 21 00 00 00 (0x21 controller temperature 33°C)

Read driver bus voltage:

Send: 0x602 40 13 20 00 00 00 00 00

Feedback: 0x582 4B 13 20 00 30 00 00 00 (0x30 driver input voltage 48V)

Read encoder count value (position):

Send: 0x602 40 16 20 00 00 00 00 00

Feedback: 0x582 43 16 20 00 F1 D2 48 00 (0x0048D2F1 encoder count value 4772593)

Parsing: 0x0048D2F1 Encoder count value 4772593 (Int32, count from zero again after exceeding the range)

Note: The return value is in hexadecimal, and after converting to decimal, it is the pulse count value after the frequency multiplication of the actual encoder by 4.

(10000 (encoder resolution * 4 frequency multiplier) = 1circle)

Fault query:

Send: 0x602 40 11 20 00 00 00 00 00

Feedback: 0x582 4B 11 20 00 01 08 00 00

Note: The returned data is in hexadecimal except for the error code and should be converted into binary for reading.

Error Code Fault Analysis:

Convert it into binary, and then count from right to left which number 1 is in, which corresponds to the fault corresponding to the number of flashes of the status indicator light.

Example: The feedback data is: 4B 11 20 00 01 08 00 00

Channel A failure: 08 01 converted to binary: 100000000001

The fault of channel A is: 1 12 (disabled, Hall fault)

Note: When using CAN transceiver to send single frame data, the speed instruction must be sent within 1000ms after sending enable instruction; otherwise, the driver judges that CAN communication is off line for automatic protection and needs to send enable instruction again after protection to start.

The following figure shows a test data format using a CAN transceiver

VII PDO Usage examples

PDO Test Drive ID: 1

CANopen on: ID 0000 01 00

CANopen off: ID 0000 02 00

RPDO1: motor control command

- Motor A enable:

ID 0x201 01 20 0D 03 00 00 00 00 (0x030D2001, lower first, higher next)

- Motor A is disabled:

ID 0x201 01 20 0C 03 00 00 00 00 (0x030C2001, lower first, higher next)

- Motor A action data 10%:

ID 0x201 00 00 00 00 E8 03 00 00 (1000=0x03E8)

Speed value: 0x03E8=1000;

Target speed: 10% of the rated speed set by the driver, ten-thousandth ratio (1000/10000)

- Motor A enabled + speed 10%:

ID 0x201 01 20 0D 03 E8 03 00 00 (0x030D2001 enable + 0x03E8 speed 10%)

- Motor A forced position return to zero:

ID 0x201 00 00 0D 03 00 00 00 00 (0x030D0000)

Use SDO to set the mapping between TPDO and RPDO:

Set before CANopen is started, ID is 600+ID

1. T/RPDO mapping needs to add DLC after the sub-index (0x0010 two bytes or 0x0020 four bytes);
2. T/RPDO is not required;
3. All the low positions are in front and the high positions are next;

Example:

- Set the first object mapped by TPDO1 to 2012 (A speed)

Send: 0x601 23 00 1A 01 10 00 12 20

Feedback: 0x581 60 00 1A 01 00 00 00 00

index 0x1A00, sub-index 0x01,

Mapping object: speed 0x2012, 0x0010, data length 16 bits

- Set the second object mapped by TPDO1 to 2010 (A current)

Send: 0x601 23 00 1A 02 10 00 10 20

Feedback: 0x581 60 00 1A 02 00 00 00 00

index 0x1A00, sub-index 0x02,

Mapping object: current 0x2010, 0x0010, data length 16 bits;

- Set the third object mapped by TPDO1 to 2016 (position A)

Send: 0x601 23 00 1A 03 20 00 16 20

Feedback: 0x581 60 00 1A 03 00 00 00 00

Index 0x1A00, sub-index 0x03,

Mapping object: position 0x2016, 0x0020, data length 32 bits;

- Set the third object mapped by TPDO3 to the temperature of the drive 0x2013 (voltage)

Send: 0x601 23 02 1A 03 10 00 13 20

Feedback: 0x581 60 02 1A 03 00 00 00 00

index 0x1A02, sub-index 0x03,

Mapping object: temperature 0x2013, 0x0010, data length 16 bits;

- Set the fourth object mapped by TPDO3 to the temperature of the drive 0x203C (temperature)

Send: 0x601 23 02 1A 04 10 00 3C 20

Feedback: 0x581 60 02 1A 04 00 00 00 00

index 0x1A02, sub-index 0x04,

Mapping object: temperature 0x203C, 0x0010, data length 16 bits;

- Set the feedback time of TPDO: (TPDO1:0x1800, TPDO2:0x1801, TPDO3:0x1802, TPDO4:0x1803)

Example: TPDO1

Send: 0x601 2B 00 18 05 E8 03 00 00

Feedback: 0x581 60 00 18 05 00 00 00 00

index 0x1800, sub-index 0x05,

Feedback time: 1000ms, 0x03E8 (lower first, higher next);

RPDO mapping object; (RPDO1: 0x1600, RPDO2: 0x1601)

Example:

- Set the first object of RPDO1 mapping to (A enable)

Send: 0x601 23 00 16 01 20 00 7A 60

Feedback: 0x581 23 00 16 01 20 00 00 00

index 0x1A02, sub-index 0x04,

Mapping object: enable 0x607A, 0x0020, data length 32 bits;

- Set the second object of RPDO1 mapping to (A action data)

Send: 0x601 23 00 16 02 20 00 81 60

Feedback: 0x581 23 00 16 02 20 00 00 00

index 0x1A02, sub-index 0x04,

Mapping object: temperature 0x6081, 0x0020, data length 32 bits;

The driver maps the data uploaded by TPDO by default (1000ms):

(0x281 is B motor data, same format as 0x181 A motor data)

- TPDO1 upload data: 0x181 DC 05 01 00 92 99 2A 01

8-bit hexadecimal

0x05DC: The current motor speed is 1500RPM

0x01: current motor current 1A

0x012A9992: current running position 19569042

- TPDO2 upload data: 0x281 is motor B data, the same format as 0x181 motor A data.

- TPDO3 upload data: 0x381 01 08 01 00 30 00 1E 00

8-bit hexadecimal

0x0801: Current A fault (disabled + Hall fault) 0x0001: Current B fault (disabled)

0x0030: The current bus voltage is 48V 0x001E: The current drive temperature is 30°C

- TPDO4 upload data: 0x481 14 51 14 51

4 digits hexadecimal

0x5114: The current control status feedback channel A is enabled;

Control status feedback:
A - 2003; B - 2004; Bit 2 (Channel A 1 is enabled; Channel A 0 is disabled.) Bit 3 (Channel B 1 is enabled. Channel B 0 is disabled) Add TPDO4: feedback two data: Channel A status and Channel B status (same now, for independent parameter setting)


输入 4(TPDO 4)
0x514C
0x514C
State feedback (two routes enabled)
计算器
查看(V) 编辑(E) 帮助(H)
Route B
Route A
1010001010011100
0000 0000 0000 0000 0000 0000 0000 0000
63 47 32
0000 0000 0000 0000 0101 0001 0100 1100
31 15 0
十六进制
十进制
八进制
二进制
Mod
A
MC
MR
MS
M+
M-
(
)
B
CE
C
±
√
RoL
RoR
C
7
8
9
/
%