SERVO MOTOR CONTROL PROTOCOL V2.0

Disclaimer

Thank you for using the RMD series motor drive system. Before use, please read this statement carefully. Once used, it will be regarded as acceptance of all contents of this statement .Please use the motor which strictly abide by the manual, product description and relevant laws, regulations, policies, installation guidelines. In the process of using the product, the user promises to be responsible for his behavior. Due to improper use, installation, modification caused by any loss, SUZHOU MICRO ACTUATOR TECHNOLOGY CO LTD(MyActuator)will not bear legal responsibility.

MyActuator is the trademark of SUZHOU MICRO ACTUATOR TECHNOLOGY CO LTD and its related companies. The product names, brands, etc. appearing in this article are all trademarks or registered trademarks of our company.

All copyright of products and handbooks are reserved by MyActuator.Reproduction in any form shall not be allowed without permission. Regarding the disclaimer the final interpretation right, all belongs to Myactuator.

1. Can bus parameters and single motor command data frame format

Bus Interface:CAN BUS
Baud rate: 1Mbps
The format of the message used to send control commands and motor replies to a single motor is as follows
Identifier：0x140 + ID(1~32)
Frame format：DATA
Frame type: standard frame
DLC：8byte

2. Single motor Command list

CAN control commands supported by RMD motor drive as following table:

SN	COMMAND NAME	COMMAND DATA
1.	Read Position loop KP data command	0x30
2.	Read Position loop Ki data command	0x31
3.	Read Speed loop KP data command	0x32
4.	Read Speed loop Ki data command	0x33
5.	Read Current loop KP data command	0x34
6.	Read Current loop Ki data command	0x35
7.	Write Position loop KP data to RAM command	0x36
8.	Write Position loop Ki data to RAM command	0x37
9.	Write Speed loop KP data to RAM command	0x38
10.	Write Speed loop Ki data to RAM command	0x39
11.	Write Current loop KP data to RAM command	0x3A
12.	Write Current loop Ki data to RAM command	0x3B
13.	Write Position loop KP data to ROM command	0x3C
14.	Write Position loop Ki data to ROM command	0x3D
15.	Write Speed loop KP data to ROM command	0x3E
16.	Write Speed loop Ki data to ROM command	0x3F
17.	Write Current loop KP data to ROM command	0x40
18.	Write Current loop Ki data to ROM command	0x41
19.	Read acceleration data command	0x42
20.	Write acceleration data to RAM command	0x43
21.	Read multiturn encoder position command	0x60
22.	Read multiturn encoder original position command	0x61
23.	Read multiturn encoder offset command	0x62
24.	Write multiturn encoder values to ROM as motor zero command	0x63
25.	Write multiturn encoder current position to ROM as motor zero command	0x64
26.	Read encoder data command	0x90
27.	Write encoder values to ROM as motor zero command	0x91
28.	Write current position to ROM as motor zero command	0x19
29.	Read multiturn turns angle command	0x92

30.	Read single circle angle command	0x94
31.	Read motor status 1 and error flag commands	0x9A
32.	Read motor status 2	0x9C
33.	Read motor status 3	0x9D
34.	Motor off command	0x80
35.	Motor stop command	0x81
36.	Motor running command	0x88
37.	Torque closed-loop command	0xA1
38.	Speed closed-loop command	0xA2
39.	Position closed-loop command 1	0xA3
40.	Position closed-loop command 2	0xA4
41.	Position closed-loop command 3	0xA5
42.	Position closed-loop command 4	0xA6
43.	Multiturn incremental position control command	0xA7
44.	Multiturn incremental position control command	0xA8
45.	read running mode	0x70
46.	read power value	0x71
47.	read Battery voltage	0x72
48.	TF command	0x73
49.	System reset command	0x76
50.	Brake opening command	0x77
51.	Brake close command	0x78
52.	CAN ID setting and reading	0x79

3. Single motor Command description

3.1 Read Position loop KP parameter command（one frame）

The host sends the command to read the current Position loop KP parameters.

Data field	Description	Data
DATA[0]	command byte	0x30
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply(one frame)

The drive reply data contains the Kp parameter of the position loop, which is converted using the Q format (Q24)

eg,kp=0.25,Position loop after conversion kp, anglePidKp=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x30
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Position loop Kp parameters low byte 1	DATA[4] = *(uint8_t *)(&anglePidKp)
DATA[5]	Position loop Kp parameters byte 2	DATA[5] = *((uint8_t *)(&anglePidKp)+1)
DATA[6]	Position loop Kp parameters byte 3	DATA[6] = *((uint8_t *)(&anglePidKp)+2)
DATA[7]	Position loop Kp parameters byte 4	DATA[7] = *((uint8_t *)(&anglePidKp)+3)

3.2 Read Position loop Ki parameter command（one frame）

The host sends the command to read the current Position loop Ki parameters.

Data field	Description	Data
DATA[0]	command byte	0x31
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00

DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply(one frame)

The drive reply data contains the Ki parameter of the position loop,which is converted using the Q format (Q24)

eg，ki=0.25,Position loop after conversion ki,anglePidKi=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x31
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Position loop Ki parameters low byte 1	DATA[4] = *(uint8_t *)(&anglePidKi)
DATA[5]	Position loop Ki parameters byte 2	DATA[5] = *((uint8_t *)(&anglePidKi)+1)
DATA[6]	Position loop Ki parameters byte 3	DATA[6] = *((uint8_t *)(&anglePidKi)+2)
DATA[7]	Position loop Ki parameters byte 4	DATA[7] = *((uint8_t *)(&anglePidKi)+3)

3.3 Read Speed loop Kp parameter command（one frame）

The host sends the command to read the current Speed loop KP parameters.

Data field	Description	Data
DATA[0]	command byte	0x32
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply(one frame)

The drive reply data contains the Kp parameter of the speed loop,which is converted using the Q format (Q24)

eg,kp=0.25,speed loop after conversion kp,speedPidKp=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x32
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	speed loop Kp parameters low byte 1	DATA[4] = *(uint8_t *)(&speedPidKp)
DATA[5]	speed loop Kp parameters byte 2	DATA[5] = *((uint8_t *)(&speedPidKp)+1)
DATA[6]	speed loop Kp parameters byte 3	DATA[6] = *((uint8_t *)(&speedPidKp)+2)
DATA[7]	speed loop Kp parameters byte 4	DATA[7] = *((uint8_t *)(&speedPidKp)+3)

3.4 Read Speed loop Ki parameter command（one frame）

The host sends the command to read the current Speed loop Ki parameters.

Data field	Description	Data
DATA[0]	command byte	0x33
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply(one frame)

The drive reply data contains the Ki parameter of the speed loop,which is converted using the Q format (Q24)

eg，ki=0.25,speed loop after conversion ki,speedPidKi=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x33
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	speed loop Ki parameters low byte 1	DATA[4] = *(uint8_t *)(&speedPidKi)
DATA[5]	speed loop Ki parameters byte 2	DATA[5] = *((uint8_t *)(&speedPidKi)+1)
DATA[6]	speed loop Ki parameters byte 3	DATA[6] = *((uint8_t *)(&speedPidKi)+2)

DATA[7]	speed loop Ki parameters byte 4	DATA[7] = *((uint8_t *)(&speedPidKi)+3)

3.5 Read Current loop Kp parameter command（one frame）

The host sends the command to read the Current loop Kp parameters.

Data field	Description	Data
DATA[0]	command byte	0x34
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply(one frame)

The drive reply data contains the Kp parameter of the current loop,which is converted using the Q format (Q24)

eg,kp=0.25,current loop after conversion kp,torquePidKp=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x34
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	current loop Kp parameters low byte 1	DATA[4] = *(uint8_t *)(&torquePidKp)
DATA[5]	current loop Kp parameters byte 2	DATA[5] = *((uint8_t *)(&torquePidKp)+1)
DATA[6]	current loop Kp parameters byte 3	DATA[6] = *((uint8_t *)(&torquePidKp)+2)
DATA[7]	current loop Kp parameters byte 4	DATA[7] = *((uint8_t *)(&torquePidKp)+3)

3.6 Read Current loop Ki parameter command（one frame）

The host sends the command to read the current Current loop Ki parameters.

Data field	Description	Data
DATA[0]	command byte	0x35
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00

DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply(one frame)

The drive reply data contains the Ki parameter of the current loop,which is converted using the Q format (Q24)

eg，ki=0.25,current loop after conversion ki,torquePidKi=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x35
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	current loop Ki parameters low byte 1	DATA[4] = *(uint8_t *)(&torquePidKi)
DATA[5]	current loop Ki parameters byte 2	DATA[5] = *((uint8_t *)(&torquePidKi)+1)
DATA[6]	current loop Ki parameters byte 3	DATA[6] = *((uint8_t *)(&torquePidKi)+2)
DATA[7]	current loop Ki parameters byte 4	DATA[7] = *((uint8_t *)(&torquePidKi)+3)

3.7 Write Position loop Kp parameter to RAM command (one frame)

The host sends the command to write the Kp parameters of position loop to the RAM, and the write parameters are invalid after the power off,and the Q format (Q24) is used for conversion.

eg，Kp=0.25,position loop after conversion Kp,anglePidKp=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x36
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Position loop Kp parameters low byte 1	DATA[4] = *(uint8_t *)(&anglePidKp)
DATA[5]	Position loop Kp parameters byte 2	DATA[5] = *((uint8_t *)(&anglePidKp)+1)
DATA[6]	Position loop Kp parameters byte 3	DATA[6] = *((uint8_t *)(&anglePidKp)+2)
DATA[7]	Position loop Kp parameters byte 4	DATA[7] = *((uint8_t *)(&anglePidKp)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the reply command is the same as the received command.

3.8 Write Position loop Ki parameter to RAM command (one frame)

The host sends the command to write the Ki parameters of position loop to the RAM, and the write parameters are invalid after the power off,and the Q format (Q24) is used for conversion.
eg，Ki=0.25,position loop after conversion Ki,anglePidKi=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x37
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Position loop Ki parameters low byte 1	DATA[4] = *(uint8_t *)(&anglePidKi)
DATA[5]	Position loop Ki parameters byte 2	DATA[5] = *((uint8_t *)(&anglePidKi)+1)
DATA[6]	Position loop Ki parameters byte 3	DATA[6] = *((uint8_t *)(&anglePidKi)+2)
DATA[7]	Position loop Ki parameters byte 4	DATA[7] = *((uint8_t *)(&anglePidKi)+3)

Drive reply(one frame)
The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.9 Write Speed loop Kp parameter to RAM command (one frame)

The host sends the command to write the Kp parameters of speed loop to the RAM, and the write parameters are invalid after the power off,and the Q format (Q24) is used for conversion.
eg，Kp=0.25,speed loop after conversion kp,speedPidKp=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x38
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	speed loop Kp parameters low byte 1	DATA[4] = *(uint8_t *)(&speedPidKp)
DATA[5]	speed loop Kp parameters byte 2	DATA[5] = *((uint8_t *)(&speedPidKp)+1)
DATA[6]	speed loop Kp parameters byte 3	DATA[6] = *((uint8_t *)(&speedPidKp)+2)
DATA[7]	speed loop Kp parameters byte 4	DATA[7] = *((uint8_t *)(&speedPidKp)+3)

Drive reply(one frame)
The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.10 Write Speed loop Ki parameter to RAM command (one frame)

The host sends the command to write the Ki parameters of speed loop to the RAM, and the write parameters are invalid after the power is turned off,and the Q format (Q24) is used for conversion.
eg，Ki=0.25,speed loop after conversion Ki,speedPidKi=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x39
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	speed loop Ki parameters low byte 1	DATA[4] = *(uint8_t *)(&speedPidKi)
DATA[5]	speed loop Ki parameters byte 2	DATA[5] = *((uint8_t *)(&speedPidKi)+1)
DATA[6]	speed loop Ki parameters byte 3	DATA[6] = *((uint8_t *)(&speedPidKi)+2)
DATA[7]	speed loop Ki parameters byte 4	DATA[7] = *((uint8_t *)(&speedPidKi)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.11 Write Current loop Kp parameter to RAM command (one frame)

The host sends the command to write the Kp parameters of current loop to the RAM, and write parameters are invalid after the power off,and the Q format (Q24) is used for conversion.
eg，kp=0.25,current loop after conversion kp,torquePidKp=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x3A
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	current loop Kp parameters low byte 1	DATA[4] = *(uint8_t *)(&torquePidKp)
DATA[5]	current loop Kp parameters byte 2	DATA[5] = *((uint8_t *)(&torquePidKp)+1)
DATA[6]	current loop Kp parameters byte 3	DATA[6] = *((uint8_t *)(&torquePidKp)+2)
DATA[7]	current loop Kp parameters byte 4	DATA[7] = *((uint8_t *)(&torquePidKp)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.12 Write Current loop Ki parameter to RAM command (one frame)

The host sends the command to write the Ki parameters of current loop to the RAM, and the write parameters are invalid after the power is turned off,and the Q format (Q24) is used for conversion.
eg，Ki=0.25,current loop after conversion Ki,torquePidKi=0.25*16777216=4194304;

DATA[0]	command byte	0x3B
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	current loop Ki parameters low byte 1	DATA[4] = *(uint8_t *)(&torquePidKi)
DATA[5]	current loop Ki parameters byte 2	DATA[5] = *((uint8_t *)(&torquePidKi)+1)
DATA[6]	current loop Ki parameters byte 3	DATA[6] = *((uint8_t *)(&torquePidKi)+2)
DATA[7]	current loop Ki parameters byte 4	DATA[7] = *((uint8_t *)(&torquePidKi)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.13 Write Position loop Kp parameter to ROM command (one frame)

The host sends the command to write the Kp parameters of position loop to the ROM, and the write parameters are valid after the power off,and the Q format (Q24) is used for conversion.
eg，kp=0.25,position loop after conversion kp,anglePidKp=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x3C
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	position loop Kp parameters low byte 1	DATA[4] = *(uint8_t *)(&anglePidKp)
DATA[5]	position loop Kp parameters byte 2	DATA[5] = *((uint8_t *)(&anglePidKp)+1)
DATA[6]	position loop Kp parameters byte 3	DATA[6] = *((uint8_t *)(&anglePidKp)+2)
DATA[7]	position loop Kp parameters byte 4	DATA[7] = *((uint8_t *)(&anglePidKp)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.14 Write Position loop Ki parameter to ROM command (one frame)

The host sends the command to write the Ki parameters of position loop to the ROM, and the write parameters are valid after the power off,and the Q format (Q24) is used for conversion.
eg，ki=0.25,position loop after conversion Ki,anglePidKi=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x3D
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	position loop Ki parameters low byte 1	DATA[4] = *(uint8_t *)(&anglePidKi)
DATA[5]	position loop Ki parameters byte 2	DATA[5] = *((uint8_t *)(&anglePidKi)+1)
DATA[6]	position loop Ki parameters byte 3	DATA[6] = *((uint8_t *)(&anglePidKi)+2)
DATA[7]	position loop Ki parameters byte 4	DATA[7] = *((uint8_t *)(&anglePidKi)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.15 Write Speed loop Kp parameter to ROM command (one frame)

The host sends the command to write the Kp parameters of speed loop to the ROM, and the write parameters are valid after the power off,and the Q format (Q24) is used for conversion.

eg，kp=0.25,speed loop after conversion kp,speedPidKp=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x3E
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	speed loop Kp parameters low byte 1	DATA[4] = *(uint8_t *)(&speedPidKp)
DATA[5]	speed loop Kp parameters byte 2	DATA[5] = *((uint8_t *)(&speedPidKp)+1)
DATA[6]	speed loop Kp parameters byte 3	DATA[6] = *((uint8_t *)(&speedPidKp)+2)
DATA[7]	speed loop Kp parameters byte 4	DATA[7] = *((uint8_t *)(&speedPidKp)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.16 Write Speed loop Ki parameter to ROM command (one frame)

The host sends the command to write the Ki parameters of speed loop to the ROM, and the write parameters are valid after the power off,and the Q format (Q24) is used for conversion.

eg，ki=0.25,speed loop after conversion Ki,speedPidKi=0.25*16777216=4194304;

Data field	Description	Data

DATA[0]	command byte	0x3F
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	speed loop Ki parameters low byte 1	DATA[4] = *(uint8_t *)(&speedPidKi)
DATA[5]	speed loop Ki parameters byte 2	DATA[5] = *((uint8_t *)(&speedPidKi)+1)
DATA[6]	speed loop Ki parameters byte 3	DATA[6] = *((uint8_t *)(&speedPidKi)+2)
DATA[7]	speed loop Ki parameters byte 4	DATA[7] = *((uint8_t *)(&speedPidKi)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.17 Write Current loop Kp parameter to ROM command (one frame)

The host sends the command to write the Kp parameters of current loop to the ROM, and the write parameters are valid after the power off,and the Q format (Q24) is used for conversion.

eg，kp=0.25,current loop after conversion kp,torquePidKp=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	NULL	0x40
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	current loop Kp parameters low byte 1	DATA[4] = *(uint8_t *)(&torquePidKp)
DATA[5]	current loop Kp parameters byte 2	DATA[5] = *((uint8_t *)(&torquePidKp)+1)
DATA[6]	current loop Kp parameters byte 3	DATA[6] = *((uint8_t *)(&torquePidKp)+2)
DATA[7]	current loop Kp parameters byte 4	DATA[7] = *((uint8_t *)(&torquePidKp)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.18 Write Current loop Ki parameter to ROM command (one frame)

The host sends the command to write the Ki parameters of current loop to the ROM, and the write parameters are valid after the power is turned off,and the Q format (Q24) is used for conversion.

eg，ki=0.25,current loop after conversion Ki ,torquePidKi=0.25*16777216=4194304;

Data field	Description	Data
DATA[0]	command byte	0x41

DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	current loop Ki parameters low byte 1	DATA[4] = *(uint8_t *)(&torquePidKi)
DATA[5]	current loop Ki parameters byte 2	DATA[5] = *((uint8_t *)(&torquePidKi)+1)
DATA[6]	current loop Ki parameters byte 3	DATA[6] = *((uint8_t *)(&torquePidKi)+2)
DATA[7]	current loop Ki parameters byte 4	DATA[7] = *((uint8_t *)(&torquePidKi)+3)

Drive reply(one frame)

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.19 Read acceleration data command (one frame)

The host send the command to read motor acceleration data

Data field	Description	Data
DATA[0]	command byte	0x42
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply（one frame）

The driver reply data include acceleration data, data type :int32_t, unit:1dps/s,Parameter range 0-10000.

Data field	Description	Data
DATA[0]	command byte	0x42
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Acceleration low byte 1	DATA[4] = *(uint8_t *)(&Accel)
DATA[5]	Acceleration byte 2	DATA[5] = *((uint8_t *)(&Accel)+1)
DATA[6]	Acceleration byte 3	DATA[6] = *((uint8_t *)(&Accel)+2)
DATA[7]	Acceleration byte 4	DATA[7] = *((uint8_t *)(&Accel)+3)

3.20 Write acceleration data to RAM command (one frame)

The host sends the command to write the acceleration to the RAM, and the write parameters are invalid after the power is turned off. Acceleration data Accel is int32_t type, unit 1dps/s,Parameter range 0-10000.

Data field	Description	Data
DATA[0]	command byte	0x43
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Acceleration low byte 1	DATA[4] = *(uint8_t *)(&Accel)
DATA[5]	Acceleration byte 2	DATA[5] = *((uint8_t *)(&Accel)+1)
DATA[6]	Acceleration byte 3	DATA[6] = *((uint8_t *)(&Accel)+2)
DATA[7]	Acceleration byte 4	DATA[7] = *((uint8_t *)(&Accel)+3)

Drive reply（one frame）

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.21 Read multiturn encoder position data command(one frame)

The host sends this command to read the encoder multi-turn position.

Data field	Description	Data
DATA[0]	command byte	0x60
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command. The frame data contains the following parameters.

Encoder multiturn position (int64_t type, multiturn encoder value range ,valid data is 6 bytes),which is the encoder original position minus the encoder multiturn zero offset value,the seventh byte represents positive and negative, 0 is positive, and 1 is negative.

Data field	Description	Data
DATA[0]	command byte	0x60
DATA[1]	Encoder position low byte1	DATA[0] = *(uint8_t *)(&encoder)
DATA[2]	Encoder position byte2	DATA[1] = *((uint8_t *)(&encoder)+1)
DATA[3]	Encoder position byte3	DATA[2] = *(uint8_t *)(&encoder)+2）
DATA[4]	Encoder position byte4	DATA[3] = *((uint8_t *)(&encoder)+3)
DATA[5]	Encoder position byte5	DATA[4] = *((uint8_t *)(&encoder)+4)
DATA[6]	Encoder position byte6	DATA[5] = *((uint8_t *)(&encoder)+5)
DATA[7]	Encoder position positive or negative flag bit	1 minus , 0 plus

3.22 Read multiturn encoder original position data command(one frame)

The host sends this command to read the encoder multi-turn position.

Data field	Description	Data
DATA[0]	command byte	0x61
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command. The frame data contains the following parameters.

Encoder multiturn original position encoderRaw(int64_t type, value range ,valid data is 6 bytes),the seventh byte represents positive and negative, 0 is positive, and 1 is negative.

Data field	Description	Data
DATA[0]	command byte	0x61
DATA[1]	Encoder original position byte1	DATA[0] = *(uint8_t *)(&encoderRaw)
DATA[2]	Encoder original position byte2	DATA[1] = *((uint8_t *)(&encoderRaw)+1)
DATA[3]	Encoder original position byte3	DATA[2] = *((uint8_t *)(&encoderRaw)+2）
DATA[4]	Encoder original position byte4	DATA[3] = *((uint8_t *)(&encoderRaw)+3)
DATA[5]	Encoder original position byte5	DATA[4] = *((uint8_t *)(&encoderRaw)+4)

DATA[6]	Encoder original position byte6	DATA[5] = *((uint8_t *)(&encoderRaw)+5)
DATA[7]	Encoder original position positive or negative flag bit	1 negative , 0 positive

3.23 Read multiturn encoder zero offset data command(one frame)

The host sends this command to read the encoder multi-turn zero offset value.

Data field	Description	Data
DATA[0]	command byte	0x62
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command. The frame data contains the following parameters.

encoder multi-turn zero offset encoderOffset（int64_t type,value range,valid data is 6 bytes）,the seventh byte represents positive and negative, 0 is positive, and 1 is negative.

Data field	Description	Data
DATA[0]	command byte	0x62
DATA[1]	Encoder offset byte1	DATA[0] = *(uint8_t *)(&encoderOffset)
DATA[2]	Encoder offset byte2	DATA[1] = *((uint8_t *)(&encoderOffset)+1)
DATA[3]	Encoder offset byte3	DATA[2] = *((uint8_t *)(&encoderOffset)+2）
DATA[4]	Encoder offset byte4	DATA[3] = *((uint8_t *)(&encoderOffset)+3)
DATA[5]	Encoder offset byte5	DATA[4] = *((uint8_t *)(&encoderOffset)+4)
DATA[6]	Encoder offset byte6	DATA[5] = *((uint8_t *)(&encoderOffset)+5)
DATA[7]	Encoder offset positive or negative flag bit	1 negative , 0 positive

3.24 Write encoder multi-turn value to ROM as motor zero command(one frame)

The host sends this command to set the encoder zero offset, where the encoder multi-turn value encoder Offset that needs to be written is int64_t type,(value range,valid data is 6 bytes)，the seventh byte represents positive and negative, 0 is positive, and 1 is negative.

Data field	Description	Data
DATA[0]	command byte	0x63
DATA[1]	Encoder offset low byte1	DATA[0] = *(uint8_t *)(&encoderOffset)
DATA[2]	Encoder offset byte2	DATA[1] = *((uint8_t *)(&encoderOffset)+1)
DATA[3]	Encoder offset byte3	DATA[2] = *((uint8_t *)(&encoderOffset)+2）
DATA[4]	Encoder offset byte4	DATA[3] = *((uint8_t *)(&encoderOffset)+3)
DATA[5]	Encoder offset byte5	DATA[4] = *((uint8_t *)(&encoderOffset)+4)
DATA[6]	Encoder offset byte6	DATA[5] = *((uint8_t *)(&encoderOffset)+5)
DATA[7]	Encoder offset positive or negative flag bit	1negative , 0 positive

Drive reply (one frame)

The motor responds to the host after receiving the command. The frame data contains the following parameters.

3.25 Write encoder current multiturn position to ROM as motor zero command(one frame)

Write the current encoder position of the motor as the initial position to the ROM
Notice:
1.This command needs to be re-powered to take effect
2.This command will write the zero position to the ROM of the drive. Multiple writes will affect the life of the chip. Frequent use is not recommended.

Data field	Description	Data
DATA[0]	command byte	0x64
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor reply to the host after receiving the command, and the data of encode roffset is the 0 offset value.

Data field	Description	Data
DATA[0]	command byte	0x64
DATA[1]	encoder offset low byte1	DATA[0] = *(uint8_t *)(&encoderOffset)
DATA[2]	encoder offset byte2	DATA[1] = *((uint8_t *)(&encoderOffset)+1)
DATA[3]	encoder offset byte3	DATA[2] = *((uint8_t *)(&encoderOffset)+2）
DATA[4]	encoder offset byte4	DATA[3] = *((uint8_t *)(&encoderOffset)+3)
DATA[5]	encoder offset byte5	DATA[4] = *((uint8_t *)(&encoderOffset)+4)
DATA[6]	encoder offset byte6	DATA[5] = *((uint8_t *)(&encoderOffset)+5)
DATA[7]	Encoder offset positive or negative flag bit	1negative , 0 positive

3.26 Read encoder data single-turn command(one frame)

The host sends this command to read the current position of the encoder. Note that the current command is used as a single-turn data reading command for direct drive motors.

Data field	Description	Data
DATA[0]	command byte	0x90
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command. The frame data contains the following parameters.

1.Encoder position encoder (int16_t type, the value range of 16bit encoder is 0~65535), which is the value of the original position of the encoder minus the zero offset of the encoder.

2.Encoder's original position encoderRaw (uint16_t type, the value range of 16bit encoder is 0~65535).

3.Encoder offset encoderOffset (uint16_t type, the value range of 16bit encoder is 0~65535), this point is regarded as the zero point of the motor angle.

Data field	Description	Data
DATA[0]	command byte	0x90
DATA[1]	NULL	0x00
DATA[2]	Encoder position low byte	DATA[2] = *(uint8_t *)(&encoder)

DATA[3]	Encoder position high byte	DATA[3] = *((uint8_t *)(&encoder)+1)
DATA[4]	Encoder original position low byte	DATA[4] = *(uint8_t *)(&encoderRaw)
DATA[5]	Encoder original position high byte	DATA[5] = *((uint8_t *)(&encoderRaw)+1)
DATA[6]	Encoder zero offset low byte	DATA[6] = *(uint8_t *)(&encoderOffset)
DATA[7]	Encoder zero offset high byte	DATA[7] = *((uint8_t *)(&encoderOffset)+1)

3.27 Write encoder value to ROM as motor zero command (1 frame)

The host sends this command to set the encoder zero offset, where the encoder value encoderOffset to be written is uint16_t type, and the value range of the 16bit encoder is 0~65535.

Data field	Description	Data
DATA[0]	command byte	0x91
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	Encoder zero offset low byte	DATA[6] = *(uint8_t *)(&encoderOffset)
DATA[7]	Encoder zero offset high byte	DATA[7] = *((uint8_t *)(&encoderOffset)+1)

Drive reply（one frame）

The motor responds to the host after receiving the command, the frame data is the same as the host sent.

3.28 Write the current position to the ROM as the motor zero command---single-turn command (1 frame)

Write the current encoder position of the motor as the initial position to the ROM

Notice:

1.This command needs to be re-powered to take effect

2.This command will write the zero point to the ROM of the drive. Multiple writes will reduce the life of the chip. Frequent use is not recommended.

Data field	Description	Data
DATA[0]	command byte	0x19
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00

DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply（one frame）

The motor responds to the host after receiving the command, the encoderOffset in the data is the zero offset value.

Data field	Description	Data
DATA[0]	command byte	0x19
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	Encoder zero offset low byte	DATA[6] = *(uint8_t *)(&encoderOffset)
DATA[7]	Encoder zero offset high byte	DATA[7] = *((uint8_t *)(&encoderOffset)+1)

3.29 Read multi turns angle command (one frame)

The host sends command to read the multi-turn angle of the motor.

Data field	Description	Data
DATA[0]	command byte	0x92
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters.

motor Angle，(int64_t type,value range,valid data is 6 bytes),the seventh byte represent positive and negative, 0 is positive and 1 is negative, unit 0.01°/LSB.

Data field	Description	Data

DATA[0]	command byte	0x92
DATA[1]	Angle low byte 1	DATA[1] = *(uint8_t *)(&motorAngle)
DATA[2]	Angle byte2	DATA[2] = *((uint8_t *)(& motorAngle)+1)
DATA[3]	Angle byte3	DATA[3] = *((uint8_t *)(& motorAngle)+2)
DATA[4]	Angle byte4	DATA[4] = *((uint8_t *)(& motorAngle)+3)
DATA[5]	Angle byte5	DATA[5] = *((uint8_t *)(& motorAngle)+4)
DATA[6]	Angle byte6	DATA[6] = *((uint8_t *)(& motorAngle)+5)
DATA[7]	Motor angle positive or negative flag bit	1 negative, 0 positive

3.30 Read single circle angle command (1 frame)

The host sends command to read the single circle angle of the motor.

Data field	Description	Data
DATA[0]	command byte	0x94
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters.

The motor single circle angle is int16_t type data. It starts from the encoder zero point and increases clockwise. When the zero point is reached again, the value returns to 0, the unit is 0.01°/LSB, and the value range is 0~35999.

Data field	Description	Data
DATA[0]	command byte	0x94
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	single angle low byte	DATA[6] = *(uint8_t *)(& circleAngle)

DATA[7]	single angle high byte	DATA[7] = *((uint8_t *)(& circleAngle)+1)

3.31 Read motor status 1 and error flag command (1 frame)

This command reads the current motor temperature, voltage and error status flag.

Data field	Description	Data
DATA[0]	command byte	0x9A
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters.

1.Motor temperature (int8_t type, unit 1℃/LSB)

2.voltage（uint16_t type，unit 0.1V/LSB）。

3.Error State (uint16_t type, each bit represents different motor state)

Data field	Description	Data
DATA[0]	command byte	0x9A
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	voltage low byte	DATA[3] = *(uint8_t *)(&voltage)
DATA[5]	voltage high byte	DATA[4] = *((uint8_t *)(& voltage)+1)
DATA[6]	Error State low byte 1	DATA[6] =*(uint8_t *)(&errorState)
DATA[7]	Error State byte 2	DATA[7] = *((uint8_t *)(& errorState)+1)

Memo：

System_errorState value state table 1 is as follows:

System_errorState value	Status description
0x0000	Hardware over-current
0x0002	Motor stalled
0x0004	Low voltage

0x0008	Over-voltage
0x0010	Over-current
0x0020	brake opening failed
0x0040	Bus current error
0x0080	Battery voltage error
0x0100	overspeed
0x0200	Position loop exceeded error
0x0400	VDD error
0x0800	DSP internal sensor temperature is overheated
0x1000	motor temperature is overheated
0x2000	Encoder calibration error

Table 1

CAN_errorState value state table 2 is as follows:

CAN_errorState value	Status description
0x00F0	PID parameter write ROM protection, non-safe operation
0x00F1	Encoder value is written into ROM protection, non-safe operation
0x00F2	Three-loop switching operation error, non-safe operation
0x00F3	Motor brake is not open
0x00F4	Motor write ROM protection, non-safe operation

Table 2

3.32 Read motor status 2 command (1 frame)

This command reads the current temperature, voltage, speed and encoder position of the motor.

Data field	Description	Data
DATA[0]	command byte	0x9C
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters.

1.Motor temperature（int8_t type，1℃/LSB）。

2.Motor torque current Iq（int16_t type，Range:-2048~2048,real torque current range:-33A~33A）。

3.Motor speed（int16_t type，1dps/LSB）。

4.Encoder position value（uint16_t type，16bit encoder value range:0~65535）。

Data field	**Description**	**Data**
DATA[0]	Command byte	0x9C
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Torque current low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&iq)+1)
DATA[4]	Speed low byte	DATA[4] = *(uint8_t *)(&speed)
DATA[5]	Speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	Encoder position low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	Encoder position high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.33 Read motor status 3 command (1 frame)

This command reads the current temperature and phase current data of the motor.

Data field	**Description**	**Data**
DATA[0]	Command byte	0x9D
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters：

1.Motor temperature（int8_t type，1℃/LSB）

2.A phase current data,the data type is int16_t type, corresponding to the actual phase current is 1A/64LSB.

3.B phase current data,the data type is int16_t type,corresponding to the actual phase current is 1A/64LSB.

4.C phase current data,the data type is int16_t type,corresponding to the actual phase current is 1A/64LSB.

Data field	Description	Data
DATA[0]	Command byte	0x9D
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Phase A current low byte	DATA[2] = *(uint8_t *)(&iA)
DATA[3]	Phase A current high byte	DATA[3] = *((uint8_t *)(& iA)+1)
DATA[4]	Phase B current low byte	DATA[4] = *(uint8_t *)(&iB)
DATA[5]	Phase B current high byte	DATA[5] = *((uint8_t *)(& iB)+1)
DATA[6]	Phase C current low byte	DATA[6] = *(uint8_t *)(&iC)
DATA[7]	Phase C current high byte	DATA[7] = *((uint8_t *)(& iC)+1)

3.34 Motor-off command (1 frame)

Turn off the motor, and clear the running state of the motor and the previously received control commands at the same time.

Data field	Description	Data
DATA[0]	Command byte	0x80
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command,the frame data is the same as that sent by the host.

3.35 Motor stop command (1 frame)

Stop the motor, but do not clear the running state of the motor and the previously received control commands.

Data field	Description	Data
DATA[0]	Command byte	0x81
DATA[1]	NULL	0x00

DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command,the frame data is the same as that sent by the host.

3.36 Motor running command (1 frame)

Resume motor operation from the motor stop command (recover the control mode before the stop).

Data field	**Description**	**Data**
DATA[0]	Command byte	0x88
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive reply (one frame)

The motor responds to the host after receiving the command,the frame data is the same as that sent by the host.

3.37 Torque closed-loop control command (1 frame)

The host sends this command to control the torque current output of the motor. The control value iqControl is type of int16_t, and the value range is -2000~2000, corresponding to the actual torque current range -32A~32A (the bus current and the actual torque of the motor vary with different motors.).

Data field	**Description**	**Data**
DATA[0]	Command byte	0xA1
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00

DATA[4]	Low byte of torque current control value	DATA[4] = *(uint8_t *)(&iqControl)
DATA[5]	high byte of torque current control value	DATA[5] = *((uint8_t *)(&iqControl)+1)
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Memo:
The control value iqControl in this command is not limited by the Max Torque Current value in the host computer of debugging software.

Drive reply (one frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters：

1.Motor temperature(int8_t type，1℃/LSB).

2.Motor torque current Iq（int16_t type，Range-2048~2048, corresponding to the actual torque current range -33A~33A).

3.Motor speed（int16_t type,1dps/LSB）。

4.Encoder position value (uint16_t type, the value range of 16bit encoder is 0~65535).

Data field	**Description**	**Data**
DATA[0]	Command byte	0xA1
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Torque current low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&iq)+1)
DATA[4]	Motor speed low byte	DATA[4] = *(uint8_t *)(&speed)
DATA[5]	Motor speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	Encoder position low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	Encoder position high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

Note: During the three-loop switching process, if the motor is not in the safe state, the drive will return the three-loop operation error value, 0x00F6, and the motor will switch to the current-loop safe state. Please be aware that the following three-loop operation instructions are similar.

Non-safe operation error response:

Data field	**Description**	**Data**
DATA[0]	Command byte	0xA1
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00

DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	Error status low byte	DATA[6] =*(uint8_t *)(&errorState)
DATA[7]	Error status high byte	DATA[6] =*((uint8_t *)(&errorState)+1)

3.38 Speed closed loop control command (1 frame)

The host sends this command to control the speed of the motor. The control value speedControl is of type int32_t, corresponding to the actual speed of 0.01dps/LSB.

Data field	Description	Data
DATA[0]	Command byte	0xA2
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Speed control low byte	DATA[4] = *(uint8_t *)(&speedControl)
DATA[5]	Speed control	DATA[5] = *((uint8_t *)(&speedControl)+1)
DATA[6]	Speed control	DATA[6] = *((uint8_t *)(&speedControl)+2)
DATA[7]	Speed control high byte	DATA[7] = *((uint8_t *)(&speedControl)+3)

Memo:

1.The maximum torque current of the motor under this command is limited by the Max Torque Current value in the host computer of debugging software.

2.In this control mode, the maximum acceleration of the motor is limited by the Max Acceleration value in the host computer of debugging software.

Drive response (1 frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters：

1.Motor temperature（int8_t type，1℃/LSB）。

2.Motor torque current(Iq)(int16_t type, range -2048~2048, corresponding to actual torque current range -33A~33A).

3.Motor speed（int16_t type,1dps/LSB）。

4.Encoder position value (uint16_t type, the value range of 16bit encoder is 0~65535).

Data field	Description	Data
DATA[0]	Command byte	0xA2
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Torque current low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&iq)+1)

DATA[4]	Motor speed low byte	DATA[4] = *(uint8_t *)(&speed)
DATA[5]	Motor speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	Encoder position low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	Encoder position high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.39 Position closed loop control command 1 (1 frame)

The host sends this command to control the position of the motor (multi-turn angle), the control value angleControl is type of int32_t, and the corresponding actual position is 0.01degree/LSB, that is, 36000 represents 360°, and the direction of rotation of the motor is determined by the difference between the target position and the current position.

Data field	**Description**	**Data**
DATA[0]	Command byte	0xA3
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Position control low byte	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	Position control	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	Position control	DATA[6] = *((uint8_t *)(&angleControl)+2)
DATA[7]	Position control high byte	DATA[7] = *((uint8_t *)(&angleControl)+3)

Memo：

1.The control value angleControl under this command is limited by the Max Angle value in the host computer of debugging software.

2.The maximum speed of the motor under this command is limited by the Max Speed value in the upper computer of debugging software.

3.In this control mode, the maximum acceleration of the motor is limited by the Max Acceleration value in the host computer of debugging software.

4.In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the host computer of debugging software.

Drive response (1 frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters：

1.Motor temperature（int8_t type,1℃/LSB）。

2.Motor torque current(Iq)(int16_t type, range -2048~2048, corresponding to actual torque current range -33A~33A).

3.Motor speed（int16_t type，1dps/LSB）。

4.Encoder position value (uint16_t type, the value range of 16bit encoder is 0~65535).

Data field	**Description**	**Data**

DATA[0]	Command byte	0xA3
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Torque current low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&iq)+1)
DATA[4]	Motor speed low byte	DATA[4] = *(uint8_t *)(&speed)
DATA[5]	Motor speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	Encoder position low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	Encoder position high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.40 Position closed loop control command 2 (1 frame)

The host sends this command to control the position of the motor (multi-turn angle), the control value angleControl is of type int32_t, and the corresponding actual position is 0.01degree/LSB, that is, 36000 represents 360°, and the direction of rotation of the motor is determined by the difference between the target position and the current position.

The control value maxSpeed limits the maximum speed of motor rotation, which is of type uint16_t, corresponding to the actual speed of 1dps/LSB.

Data field	**Description**	**Data**
DATA[0]	Command byte	0xA4
DATA[1]	NULL	0x00
DATA[2]	Speed limit low byte	DATA[2] = *(uint8_t *)(&maxSpeed)
DATA[3]	Speed limit high byte	DATA[3] = *((uint8_t *)(&maxSpeed)+1)
DATA[4]	Position control low byte	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	Position control	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	Position control	DATA[6] = *((uint8_t *)(&angleControl)+2)
DATA[7]	Position control high byte	DATA[7] = *((uint8_t *)(&angleControl)+3)

Memo:

1.The control value angleControl under this command is limited by the Max Angle value in the host computer.

2.In this control mode, the maximum acceleration of the motor is limited by the Max Acceleration value in the host computer.

3.In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the host computer.

Drive response (1 frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters：

1.Motor temperature（int8_t type，1℃/LSB）。

2.Motor torque current(Iq)(int16_t type, range -2048~2048, corresponding to actual torque current range -33A~33A).

3.Motor speed（int16_t type，1dps/LSB）。

4.Encoder position value (uint16_t type, the value range of 16bit encoder is 0~65535).

Data field	Description	Data
DATA[0]	Command byte	0xA4
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Torque current low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&iq)+1)
DATA[4]	Motor speed low byte	DATA[4] = *(uint8_t *)(&speed)
DATA[5]	Motor speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	Encoder position low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	Encoder position high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.41 Position closed loop control command 3 (1 frame)

The host sends this command to control the position of the motor (single-turn angle), the control value angleControl is of uint16_t type, the value range is 0~35999, and the corresponding actual position is 0.01degree/LSB, that is, the actual angle range is 0°~359.99°.

The control value spinDirection sets the direction of motor rotation, which is of type uint8_t, 0x00 means clockwise, 0x01 means counterclockwise.

Data field	Description	Data
DATA[0]	Command byte	0xA5
DATA[1]	Rotation direction byte	DATA[1] = spinDirection
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Position control low byte	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	Position control high byte	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Memo:

1.The maximum speed of the motor under this command is limited by the Max Speed value in the host computer.

2.In this control mode, the maximum acceleration of the motor is limited by the Max Acceleration value in the host computer.

3.In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the host computer.

Drive response (1 frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters：

1.Motor temperature（int8_t type，1℃/LSB）。

2.Motor torque current(Iq)(int16_t type, range -2048~2048, corresponding to actual torque current range -33A~33A).

3.Motor speed（int16_t type,1dps/LSB）。

4.Encoder position value (uint16_t type, the value range of 16bit encoder is 0~65535).

Data field	**Description**	**Data**
DATA[0]	Command byte	0xA5
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Torque current low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&iq)+1)
DATA[4]	Motor speed low byte	DATA[4] = *(uint8_t *)(&speed)
DATA[5]	Motor speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	Encoder position low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	Encoder position high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.42 Position closed loop control command 4 (1 frame)

The host sends this command to control the position of the motor (single turn angle).

1.AngleControl is of uint16_t type, the value range is 0~35999, and the corresponding actual position is 0.01degree/LSB, that is, the actual angle range is 0°~359.99°.

2.spinDirection sets the direction of motor rotation, which is of uint8_t type, 0x00 means clockwise, 0x01 means counterclockwise

3.maxSpeed limits the maximum speed of motor rotation, which is of uint16_t type, corresponding to the actual speed of 1dps/LSB.

Data field	**Description**	**Data**
DATA[0]	Command byte	0xA6
DATA[1]	Rotation direction byte	DATA[1] = spinDirection
DATA[2]	Speed limit low byte	DATA[2] = *(uint8_t *)(&maxSpeed)
DATA[3]	Speed limit high byte	DATA[3] = *((uint8_t *)(&maxSpeed)+1)
DATA[4]	Position control low byte	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	Position control high byte	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Memo:

1.In this control mode, the maximum acceleration of the motor is limited by the Max Acceleration value in the host computer.
2.In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the host computer.

Drive response (1 frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters：
1. Motor temperature（int8_t type，1℃/LSB）。
2.Motor torque current(Iq)(int16_t type, range -2048~2048, corresponding to actual torque current range -33A~33A).
3.Motor speed（int16_t type，1dps/LSB）。
4.Encoder position value (uint16_t type, the value range of 16bit encoder is 0~65535)

Data field	Description	Data
DATA[0]	Command byte	0xA6
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Torque current low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&iq)+1)
DATA[4]	Motor speed low byte	DATA[4] = *(uint8_t *)(&speed)
DATA[5]	Motor speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	Encoder position low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	Encoder position high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.43 Position closed loop control command 5 (1 frame)

The host sends this command to control the incremental position of the motor (multi-turn angle), and run the input position increment with the current position as the starting point. The control value angleControl is of type int32_t, and the corresponding actual position is 0.01degree/LSB, that is, 36000 represents 360° , The direction of motor rotation is determined by the incremental position sign.

Data field	Description	Data
DATA[0]	Command byte	0xA7
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Position control low byte	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	Position control	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	Position control	DATA[6] = *((uint8_t *)(&angleControl)+2)
DATA[7]	Position control high byte	DATA[7] = *((uint8_t *)(&angleControl)+3)

Memo:

1. The control value angleControl under this command is limited by the Max Angle value in the host computer.
2. The maximum speed of the motor under this command is limited by the Max Speed value in the host computer.
3. In this control mode, the maximum acceleration of the motor is limited by the Max Acceleration value in the host computer.
4. In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the host computer.

Drive response (1 frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters：

1. motor temperature（int8_t type，1℃/LSB）.
2. Motor torque current(Iq)(int16_t type, range -2048~2048, corresponding to actual torque current range -33A~33A).
3. motor speed(int16_t type，1dps/LSB).
4. Encoder position value (uint16_t type, the value range of 16bit encoder is 0~65535)

Data field	Description	Data
DATA[0]	Command byte	0xA7
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Torque current low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&iq)+1)
DATA[4]	Motor speed low byte	DATA[4] = *(uint8_t *)(&speed)
DATA[5]	Motor speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	Encoder position low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	Encoder position high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.44 Position closed loop control command 6 (1 frame)

The host sends this command to control the incremental position (multi-turn angle) of the motor, and runs the input position increment with the current position as the starting point. The control value angleControl is of type int32_t, and the corresponding actual position is 0.01degree/LSB, that is, 36000 represents 360°, and the motor rotation direction is determined by the incremental position sign.

The control value maxSpeed limits the maximum speed of motor rotation, which is of type uint16_t, corresponding to the actual speed of 1dps/LSB.

Data field	Description	Data
DATA[0]	Command byte	0xA8
DATA[1]	NULL	0x00
DATA[2]	Speed limit low byte	DATA[2] = *(uint8_t *)(&maxSpeed)

DATA[3]	Speed limit high byte	DATA[3] = *((uint8_t *)(&maxSpeed)+1)
DATA[4]	Position control low byte	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	Position control	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	Position control	DATA[6] = *((uint8_t *)(&angleControl)+2)
DATA[7]	Position control high byte	DATA[7] = *((uint8_t *)(&angleControl)+3)

Memo:

1.The control value angleControl under this command is limited by the Max Angle value in the host computer.

2. In this control mode, the maximum acceleration of the motor is limited by the Max Acceleration value in the host computer.

3. In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the host computer.

Drive response (1 frame)

The motor responds to the host after receiving the command, the frame data contains the following parameters:

1.Motor temperature（int8_t type，1℃/LSB）。

2.Motor torque current(Iq)(int16_t type, range -2048~2048, corresponding to actual torque current range -33A~33A)

3.Motor speed（int16_t type，1dps/LSB）。

4.Encoder position value (uint16_t type, the value range of 16bit encoder is 0~65535).

Data field	**Description**	**Data**
DATA[0]	Command byte	0xA8
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	Torque current low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&iq)+1)
DATA[4]	Motor speed low byte	DATA[4] = *(uint8_t *)(&speed)
DATA[5]	Motor speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	Encoder position low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	Encoder position high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.45 System operation mode acquisition (1 frame)

This command reads the current motor running mode.

Data field	**Description**	**Data**
DATA[0]	Command byte	0x70
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00

DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive response (1 frame)

The motor responds to the host after receiving the command, and the drive reply data contains the parameter run mode operating status, which is of type uint8_t.

The motor operation mode has the following 4 states：

1.Current loop mode(0x00).

2.Speed loop mode(0x01).

3.Position loop mode(0x02).

4.Power-on initialization state, not in three-ring mode(0xFF).

Data field	Description	Data
DATA[0]	Command byte	0x70
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	Motor running mode	DATA[7] = *(uint8_t *)(&runmode)

3.46 Motor power acquisition (1 frame)

This command reads the current motor running mode.

Data field	Description	Data
DATA[0]	Command byte	0x71
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive response (1 frame)

The motor responds to the host after receiving the command. The drive response data contains the motor power parameter motor power, which is of type uint16_t, the unit is watts, and the unit is 0.1w/LSB.

Data field	Description	Data
DATA[0]	Command byte	0x71
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	Motor running power low byte	DATA[6] = *(uint8_t *)(&motorpower)
DATA[7]	Motor running power high byte	DATA[7] = *((uint8_t *)(&motorpower)+1)

3.47 Obtaining battery voltage value (1 frame)

This command reads the current auxiliary battery voltage.

Data field	Description	Data
DATA[0]	Command byte	0x72
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive response (1 frame)

The motor will reply to the host after receiving the command. The driver reply data contains the auxiliary battery voltage parameter batvoltage, which is of type uint8_t, the unit is volts, and the unit is 0.1v/LSB.

Data field	Description	Data
DATA[0]	Command byte	0x72
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00

DATA[6]	NULL	0x00
DATA[7]	High battery voltage	DATA[7] = *(uint8_t *)(&batvoltage)

3.48 TF command setting (1 frame)

This command sets the current feedforward current size, the parameters are as follows.

1.TF feedforward current value TFCurrent of the motor (int16_t type,unit A, zoom in 100 times in current mode, such as 10 corresponds to 0.1A)

2.Encoder position multi-turn encoder (int32_t type).

Data field	Description	Data
DATA[0]	Command byte	0x73
DATA[1]	NULL	0x00
DATA[2]	TF feedforward current value low byte	DATA[6] = *(uint8_t *)(&TFCurrent)
DATA[3]	TF feedforward current value high byte	DATA[7] = *((uint8_t *)(&TFCurrent)+1)
DATA[4]	Encoder position low byte 1	DATA[4] = *(uint8_t *)(&encoder)
DATA[5]	Encoder position low byte 2	DATA[5] = *((uint8_t *)(&encoder)+1)
DATA[6]	Encoder position low byte 3	DATA[6] = *((uint8_t *)(&encoder)+2)
DATA[7]	Encoder position low byte 4	DATA[7] = *((uint8_t *)(&encoder)+3)

Drive response (1 frame)

The motor responds to the host after receiving the command,the following parameters are included in the driver response data.

1 .Motor torque current(Iq)(int16_t type，zoom in 100 times in current mode, such as 10 corresponds to 0.1A)

2 .motor speed（int16_t type，1dps/LSB）。

3 .Encoder position multi-turn encoder (int32_t type).

Data field	Description	Data
DATA[0]	Torque current low byte	DATA[0] = *(uint8_t *)(&iq)
DATA[1]	Torque current high byte	DATA[1] = *((uint8_t *)(&iq)+1)
DATA[2]	Motor speed low byte	DATA[2] = *(uint8_t *)(&speed)
DATA[3]	Motor speed low byte	DATA[3] = *((uint8_t *)(&speed)+1)
DATA[4]	Encoder position low byte 1	DATA[4] = *(uint8_t *)(&encoder)
DATA[5]	Encoder position byte 2	DATA[5] = *((uint8_t *)(&encoder)+1)
DATA[6]	Encoder position byte 3	DATA[6] = *((uint8_t *)(&encoder)+2)
DATA[7]	Encoder position byte 4	DATA[7] = *((uint8_t *)(&encoder)+3)

3.49 System reset command (1 frame)

This command is used to reset the system software.

Data field	Description	Data
DATA[0]	Command byte	0x76
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive response (1 frame)

The motor responds to the host after receiving the command,the frame data is the same as that sent by the host.

Data field	Description	Data
DATA[0]	Command byte	0x76
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

3.50 System brake opening command (1 frame)

This command is used to open the system brake.

Data field	Description	Data
DATA[0]	Command byte	0x77
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00

DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive response (1 frame)

The motor responds to the host after receiving the command,the frame data is the same as that sent by the host.

Data field	Description	Data
DATA[0]	Command byte	0x77
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

3.51 System brake close command (1 frame)

This command is used to open the system brake.

Data field	Description	Data
DATA[0]	Command byte	0x78
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Drive response (1 frame)

The motor responds to the host after receiving the command,the frame data is the same as that sent by the host.

Data field	Description	Data
DATA[0]	Command byte	0x78
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00

DATA[3]	NULL	0x00
DATA[4]	NULL	0x00
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

3.52 CAN ID setting and reading command (1 frame)

This command is used to set and read CAN ID.
The host sends this command to set and read CAN ID. The parameters are as follows.
1.The read and write flag bit is bool type, 1 read 0 write.
2.CANID,Size range (#1~#32), uint16_t type (synchronized with the host computer function), device identifier 0x140 + ID (1~32).

Data field	Description	Data
DATA[0]	Command byte	0x79
DATA[1]	NULL	0x00
DATA[2]	Read and write flags	DATA[2] = wReadWriteFlag
DATA[3]	NULL	0x00
DATA[4]	CANID low byte1	DATA[4] = *(uint8_t *)(&CANID)
DATA[5]	CANID byte 2	DATA[5] = *((uint8_t *)(&CANID)+1)
DATA[6]	CANID byte 3	DATA[6] = *((uint8_t *)(&CANID)+2)
DATA[7]	CANID byte 4	DATA[7] = *((uint8_t *)(&CANID)+3)

Driver reply (one frame)

1.The motor responds to the host after receiving the command,which is divided into the following two Situations.
2.Set CANID, range 1-32, and return to the original command.
3.Read CANID, return parameters are as follows.

Data field	Description	Data
DATA[0]	Command byte	0x79
DATA[0]	NULL	0x00
DATA[0]	Read and write flags	DATA[2] = wReadWriteFlag
DATA[0]	NULL	0x00
DATA[4]	CANID low byte 1	DATA[4] = *(uint8_t *)(&CANID)
DATA[5]	CANID byte 2	DATA[5] = *((uint8_t *)(&CANID)+1)
DATA[6]	CANID byte 3	DATA[6] = *((uint8_t *)(&CANID)+2)
DATA[7]	CANID byte 4	DATA[7] = *((uint8_t *)(&CANID)+3)

4. Multi-motor command

4.1 Multiple motor torque closed loop control commands(one frame)

The format of the message used to send commands to multiple motors at the same time, as followed:
Identifier：0x280
Frame format：DATA
Frame type：standard frame
DLC：8byte
The host simultaneously send this command to control the torque current output up to 4 motors. The control value iqControl is int16_t type, the value range is -2000~2000, corresponding to the actual torque current range -32A~32A (The bus current and the actual torque of the motor vary from motor to motor). The motor ID should be set to #1~#4, and cannot be repeated, corresponding to the 4 torque currents in the frame data.

Data field	Description	Data
DATA[0]	Torque current 1 control value low byte	DATA[0] = *(uint8_t *)(&iqControl_1)
DATA[1]	Torque current 1 control value high byte	DATA[1] = *((uint8_t *)(&iqControl_1)+1)
DATA[2]	Torque current 2 control value low byte	DATA[2] = *(uint8_t *)(&iqControl_2)
DATA[3]	Torque current 2 control value high byte	DATA[3] = *((uint8_t *)(&iqControl_2)+1)
DATA[4]	Torque current 3 control value low byte	DATA[4] = *(uint8_t *)(&iqControl_3)
DATA[5]	Torque current 3 control value high byte	DATA[5] = *((uint8_t *)(&iqControl_3)+1)
DATA[6]	Torque current 4 control value low byte	DATA[6] = *(uint8_t *)(&iqControl_4)
DATA[7]	Torque current 4 control value high byte	DATA[7] = *((uint8_t *)(&iqControl_4)+1)

4.2 Driver reply (one frame)

The message format of each motor reply command is as follows:
Identifier：0x140 + ID(1~4)
Frame format：DATA
Frame type：standard frame
DLC：8byte

Each motor reply according to the ID from small to large, and the reply data of each motor is the same as the single motor torque closed-loop control command reply data.