MC servo driver protocol

(Version:2.4)

Disclaimer

Thank you for using the MC 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.

Version revised record:

V2.44 20211124:

1.The version information is added to the CAN command.

2. Increase the start-up time in the CAN command.

3. Execute the new instruction directly after the execution of the modified position loop is completed.

4. Add speed feed forward in TF instruction.

V2.45：20211126：

1. Add funcation that CAN, 485 and serial port as long as there is a continuous communication, it can run normally. if not, stop the current output and continue to execute after replying.

V2.46：20211207:

1. The 9A command adds the brake control command query.

2. Increase the communication interruption protection time setting.

V2.47：20211220：

1. Modify the position command data type to 32-bit type, including the zero position.

2. Remove the single-turn position command.

V2.48：20211223：

1. Modify the 0x88 command to enter the position loop enabled and open the brake.

2. Modify the communication interruption reply error repetitive problem.

3. Increase the communication interruption judgment after entering the loop mode.

V2.48：20211230：

1. Delete the 0x00F2 error command.

2. Delete the 0x0000 hardware overcurrent error instruction.

3. Delete the 0x0020 brake release failure error command.

4.Delete 0x0080 battery voltage error command

1. Format of CAN BUS AND SINGLE MOTOR COMMAND

Bus Interface：CAN

Baudrate：1Mbps

Indentifier：0x140 + ID(1~32)

Frame format：DATA

Frame type ：standard frame

DLC：8byte

Note: Before sending the control command, please send the 0x88 command to make the motor enter the enabled state (wait for 3 seconds till motor complete the initialization), then send control command required

2. Single Motor Command list

Command name	Command data
Read position loop KP Parameter	0x30
Read position loop KI Parameter	0x31
Read Velocity loop KP Parameter	0x32
Read Velocity loop KI Parameter	0x33
Read Torque loop KP Parameter	0x34
Read Torque loop KI Parameter	0x35
Write Position loop KP to RAM	0x36
Write Position loop KI to RAM	0x37
Write Velocity loop KP to RAM	0x38

Write Velocity loop KI to RAM	0x39
Write Torque loop KP to RAM	0x3A
Write Torque loop KI to RAM	0x3B
Write Position loop KP to ROM	0x3C
Write Position loop KI to ROM	0x3D
Write Velocity loop KP to ROM	0x3E
Write Velocity loop KI to ROM	0x3F
Write Torque loop KP to ROM	0x40
Write Torque loop KI to ROM	0x41
Read Acceleration	0x42
Write acceleration data to RAM	0x43
Read multiturn encoder position	0x60
Read multiturn encoder original position	0x61
Read multiturn encoder offset	0x62
Write multiturn encoder value to ROM as motor zero	0x63
Write multiturn encoder current position to ROM as motor zero	0x64
Read multiturn turns angle	0x92
Read motor status 1 and error flag	0x9A
Read motor status 2	0x9C
Read motor status 3	0x9D
Motor off	0x80
Motor stop	0x81
Motor enable	0x88
Torque closed-loop contol	0xA1
Velocity closed-loop contol	0xA2
Multi-Position closed-loop control	0xA4
Incremental Position closed-loop control	0xA8
Read motor curent working mode	0x70

Read motor current power	0x71
Read backup power voltage	0x72
TF Command	0x73
System reset command	0x76
Brake unlock command （motor turn freely ）	0x77
Brake lock (motor shaft locked)	0x78
Setup and read CAN ID	0x79
Read System running time	0xB1
Read firmware version	0xB2
Setup Communication interruption protection time	0xB3

3.Single motor Command description

3.1 Read position loop KP parameter（one frame）

The host sends the command to read the current position loop KP parameters

Data Field	Instrucation	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

Driver 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	Instrucation	Data
DATA[0]	Command byte	0x30

DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Kp parameter low byte 1	DATA[4] = *(uint8_t *)(&anglePidKp)
DATA[5]	Kp parameter byte 2	DATA[5] = *((uint8_t *)(&anglePidKp)+1)
DATA[6]	Kp parameter byte 3	DATA[6] = *((uint8_t *)(&anglePidKp)+2)
DATA[7]	Kp parameter byte 4	DATA[7] = *((uint8_t *)(&anglePidKp)+3)

3.2 Read position loop KI parameter (one frame)

The host sends the command to read the current position loop KI parameters

Data Field	Instrucation	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

Driver 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	Instrucation	Data
DATA[0]	Command byte	0x31
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00

DATA[4]	Ki parameter low byte 1	DATA[4] = *(uint8_t *)(&anglePidKi)
DATA[5]	Ki parameter byte 2	DATA[5] = *((uint8_t *)(&anglePidKi)+1)
DATA[6]	Ki parameter byte 3	DATA[6] = *((uint8_t *)(&anglePidKi)+2)
DATA[7]	Ki parameter byte4	DATA[7] = *((uint8_t *)(&anglePidKi)+3)

3.3 Read Volecity Loop KP Parameter (one frame)

The host sends the command to read the current velocity loop KP parameters

Data Field	Instrucation	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

Driver reply (one frame)

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

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

Data Field	Instrucation	Data
DATA[0]	Command byte	0x32
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Kp parameter low byte 1	DATA[4] = *(uint8_t *)(&speedPidKp)
DATA[5]	Kp parameter byte 2	DATA[5] = *((uint8_t *)(&speedPidKp)+1)
DATA[6]	Kp parameter byte 3	DATA[6] = *((uint8_t *)(&speedPidKp)+2)
DATA[7]	Kp parameter byte 4	DATA[7] = *((uint8_t *)(&speedPidKp)+3)

3.4 Read Volecity Loop KI Parameter (one frame)

The host sends the command to read the current velocity loop KI parameters

Data Field	Instrucation	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

Driver reply (one frame)

The drive reply data contains the KI parameter of the velocity loop, which is converted using the Q format (Q24)
eg,ki=0.25,Velocity loop after conversion ki, velocity PidKi=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x33
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Ki parameter low byte 1	DATA[4] = *(uint8_t *)(&speedPidKi)
DATA[5]	Ki parameter byte 2	DATA[5] = *((uint8_t *)(&speedPidKi)+1)
DATA[6]	Ki parameter byte 3	DATA[6] = *((uint8_t *)(&speedPidKi)+2)
DATA[7]	Ki parameter byte4	DATA[7] = *((uint8_t *)(&speedPidKi)+3)

3.5 Read Torque loop KP parameter (one frame)

The host sends the command to read the current torque loop KP parameters。

Data Field	Instrucation	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

Driver reply （one frame）

The drive reply data contains the Kp parameter of the torque loop, which is converted using the Q format (Q24)
eg,kp=0.25,torque loop after conversion kp, anglePidKp=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x34
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Kp parameter low byte 1	DATA[4] = *(uint8_t *)(&torquePidKp)
DATA[5]	Kp parameter byte 2	DATA[5] = *((uint8_t *)(&torquePidKp)+1)
DATA[6]	Kp parameter byte 3	DATA[6] = *((uint8_t *)(&torquePidKp)+2)
DATA[7]	Kp parameter byte 4	DATA[7] = *((uint8_t *)(&torquePidKp)+3)

3.6 Read Torque loop KI parameter (one frame)

The host sends the command to read the current torque loop KI parameters

Data Field	Instrucation	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

Driver reply （one frame）

The drive reply data contains the Ki parameter of the torque loop, which is converted using the Q format (Q24)
eg,ki=0.25,torque loop after conversion ki, torquePidKi=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x35
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Ki parameter low byte 1	DATA[4] = *(uint8_t *)(&torquePidKi)
DATA[5]	Ki parameter byte 2	DATA[5] = *((uint8_t *)(&torquePidKi)+1)
DATA[6]	Ki parameter byte 3	DATA[6] = *((uint8_t *)(&torquePidKi)+2)
DATA[7]	Ki parameter byte 4	DATA[7] = *((uint8_t *)(&torquePidKi)+3)

3.7 Write position loop KP parameter to RAM (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	Instrucation	Data
DATA[0]	Command byte	0x36
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Kp parameter low byte 1	DATA[4] = *(uint8_t *)(&anglePidKp)

DATA[5]	Kp parameter byte 2	DATA[5] = *((uint8_t *)(&anglePidKp)+1)
DATA[6]	Kp parameter byte 3	DATA[6] = *((uint8_t *)(&anglePidKp)+2)
DATA[7]	Kp parameter 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 (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	Instrucation	Data
DATA[0]	Command byte	0x37
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Ki parameter low byte 1	DATA[4] = *(uint8_t *)(&anglePidKi)
DATA[5]	Ki parameter byte 2	DATA[5] = *((uint8_t *)(&anglePidKi)+1)
DATA[6]	Ki parameter byte 3	DATA[6] = *((uint8_t *)(&anglePidKi)+2)
DATA[7]	Ki parameter byte 4	DATA[7] = *((uint8_t *)(&anglePidKi)+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.

3.9 Write velocity loop KP parameter to RAM (One frame)

The host sends the command to write the Kp parameters of velocity 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,velocity loop after conversion Kp,velocity PidKp=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x38
DATA[1]	NULL	0x00

DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Kp parameter low byte 1	DATA[4] = *(uint8_t *)(&speedPidKp)
DATA[5]	Kp parameter byte 2	DATA[5] = *((uint8_t *)(&speedPidKp)+1)
DATA[6]	Kp parameter byte 3	DATA[6] = *((uint8_t *)(&speedPidKp)+2)
DATA[7]	Kp parameter byte 4	DATA[7] = *((uint8_t *)(&speedPidKp)+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.

3.10 Write velocity loop KI parameter to RAM (One frame)

The host sends the command to write the KI parameters of velocity 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,velocity loop after conversion KI,velocityPidKI=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x39
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Ki parameter low byte 1	DATA[4] = *(uint8_t *)(&speedPidKi)
DATA[5]	Ki parameter byte 2	DATA[5] = *((uint8_t *)(&speedPidKi)+1)
DATA[6]	Ki parameter byte 3	DATA[6] = *((uint8_t *)(&speedPidKi)+2)
DATA[7]	Ki parameter byte 4	DATA[7] = *((uint8_t *)(&speedPidKi)+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.

3.11 Write Torque loop KP parameter to RAM (One frame)

The host sends the command to write the Kp parameters of torque 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,torque loop after conversion Kp,torquePidKp=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x3A

DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Kp parameter low byte 1	DATA[4] = *(uint8_t *)(&torquePidKp)
DATA[5]	Kp parameter byte 2	DATA[5] = *((uint8_t *)(&torquePidKp)+1)
DATA[6]	Kp parameter byte 3	DATA[6] = *((uint8_t *)(&torquePidKp)+2)
DATA[7]	Kp parameter byte 4	DATA[7] = *((uint8_t *)(&torquePidKp)+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.

3.12 Write Torque loop KI parameter to RAM (One frame)

The host sends the command to write the Ki parameters of torque 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,torque loop after conversion Ki,torquePidKi=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x3B
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Ki parameter low byte 1	DATA[4] = *(uint8_t *)(&torquePidKi)
DATA[5]	Ki parameter byte 2	DATA[5] = *((uint8_t *)(&torquePidKi)+1)
DATA[6]	Ki parameter byte 3	DATA[6] = *((uint8_t *)(&torquePidKi)+2)
DATA[7]	Ki parameter byte 4	DATA[7] = *((uint8_t *)(&torquePidKi)+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.

3.13 Write position loop KP to ROM (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	Instrucation	Data

DATA[0]	Command byte	0x3C
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Kp parameter low byte 1	DATA[4] = *(uint8_t *)(&anglePidKp)
DATA[5]	Kp parameter byte 2	DATA[5] = *((uint8_t *)(&anglePidKp)+1)
DATA[6]	Kp parameter byte 3	DATA[6] = *((uint8_t *)(&anglePidKp)+2)
DATA[7]	Kp parameter 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.

3.14 Write position loop KI to ROM (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	Instrucation	Data
DATA[0]	Command byte	0x3D
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Ki parameter low byte 1	DATA[4] = *(uint8_t *)(&anglePidKi)
DATA[5]	Ki parameter byte 2	DATA[5] = *((uint8_t *)(&anglePidKi)+1)
DATA[6]	Ki parameter byte 3	DATA[6] = *((uint8_t *)(&anglePidKi)+2)
DATA[7]	Ki parameter byte 4	DATA[7] = *((uint8_t *)(&anglePidKi)+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.

3.15 Write velocity loop KP to ROM (One frame)

The host sends the command to write the Kp parameters of velocity 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,velocity loop after conversion Kp,velocity PidKp=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x3E
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Kp parameter low byte 1	DATA[4] = *(uint8_t *)(&speedPidKp)
DATA[5]	Kp parameter byte 2	DATA[5] = *((uint8_t *)(&speedPidKp)+1)
DATA[6]	Kp parameter byte 3	DATA[6] = *((uint8_t *)(&speedPidKp)+2)
DATA[7]	Kp parameter byte 4	DATA[7] = *((uint8_t *)(&speedPidKp)+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.

3.16 Write velocity loop KI to ROM (One frame)

The host sends the command to write the KI parameters of velocity 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,velocity loop after conversion KI,velocity PidKI=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x3F
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Ki parameter low byte 1	DATA[4] = *(uint8_t *)(&speedPidKi)
DATA[5]	Ki parameter byte 2	DATA[5] = *((uint8_t *)(&speedPidKi)+1)
DATA[6]	Ki parameter byte 3	DATA[6] = *((uint8_t *)(&speedPidKi)+2)
DATA[7]	Ki parameter byte 4	DATA[7] = *((uint8_t *)(&speedPidKi)+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.

3.17 Write Torque loop KP to ROM (One frame)

The host sends the command to write the Kp parameters of torque 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,torque loop after conversion Kp,torque PidKp=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x40
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Kp parameter low byte 1	DATA[4] = *(uint8_t *)(&torquePidKp)
DATA[5]	Kp parameter byte 2	DATA[5] = *((uint8_t *)(&torquePidKp)+1)
DATA[6]	Kp parameter byte 3	DATA[6] = *((uint8_t *)(&torquePidKp)+2)
DATA[7]	Kp parameter byte 4	DATA[7] = *((uint8_t *)(&torquePidKp)+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.

3.18 Write torque loop KI to ROM (One frame)

The host sends the command to write the KI parameters of torque 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,torque loop after conversion KI,torque PidKI=0.25*16777216=4194304;

Data Field	Instrucation	Data
DATA[0]	Command byte	0x41
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Ki parameter low byte 1	DATA[4] = *(uint8_t *)(&torquePidKi)
DATA[5]	Ki parameter byte 2	DATA[5] = *((uint8_t *)(&torquePidKi)+1)
DATA[6]	Ki parameter byte 3	DATA[6] = *((uint8_t *)(&torquePidKi)+2)
DATA[7]	Ki parameter byte 4	DATA[7] = *((uint8_t *)(&torquePidKi)+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.

3.19 Read Acceleration (one frame)

The host send the command to read motor acceleration data

Data Field	Instrucation	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	Instrucation	Data
DATA[0]	Command byte	0x42
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Accel Low byte 1	DATA[4] = *(uint8_t *)(&Accel)
DATA[5]	Accel byte 2	DATA[5] = *((uint8_t *)(&Accel)+1)
DATA[6]	Accel byte 3	DATA[6] = *((uint8_t *)(&Accel)+2)
DATA[7]	Accel byte 4	DATA[7] = *((uint8_t *)(&Accel)+3)

3.20 **Write acceleration to RAM (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	Instrucation	Data
DATA[0]	Command byte	0x43

DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Accel Low byte 1	DATA[4] = *(uint8_t *)(&Accel)
DATA[5]	Accel byte 2	DATA[5] = *((uint8_t *)(&Accel)+1)
DATA[6]	Accel byte 3	DATA[6] = *((uint8_t *)(&Accel)+2)
DATA[7]	Accel byte 4	DATA[7] = *((uint8_t *)(&Accel)+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.

3.21 Read multi-turn encoder position data (one frame)

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

Data Field	Instrucation	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 (int32_t type, multi-turn encoder value range:valid data 4 bytes),which is the encoder original position minus the encoder multiturn zero offset value

Data Field	Instrucation	Data
DATA[0]	Command byte	0x60
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00

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)

3.22 Read multiturn encoder original position data (one frame)

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

Data Field	Instrucation	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(int32_t type, range: valid data 4 bytes)

Data Field	Instrucation	Data
DATA[0]	Command byte	0x61
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Encoder original position byte1	DATA[4] = *(uint8_t *)(&encoderRaw)
DATA[5]	Encoder original position byte2	DATA[5] = *((uint8_t *)(&encoderRaw)+1)
DATA[6]	Encoder original position byte3	DATA[6] = *((uint8_t *)(&encoderRaw)+2）
DATA[7]	Encoder original position byte4	DATA[7] = *((uint8_t *)(&encoderRaw)+3)

3.23 Read multiturn encoder zero offset data (one frame)

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

Data Field	Instrucation	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.

encoderOffset（int32_tint32_t type, range: valid data 4 bytes)

Data Field	Instrucation	Data
DATA[0]	Command byte	0x62
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	encoderOffset byte 1	DATA[4] = *(uint8_t *)(&encoderOffset)
DATA[5]	encoderOffset byte 2	DATA[5] = *((uint8_t *)(&encoderOffset)+1)
DATA[6]	encoderOffset byte 3	DATA[6] = *((uint8_t *)(&encoderOffset)+2)
DATA[7]	encoderOffset byte 4	DATA[7] = *((uint8_t *)(&encoderOffset)+3)

3.24 Write encoder value to ROM as motor zero (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 int32_t type,(range：4 bytes)

Notice：

1. This command valid after re-power

2.This command will write the zero postion to ROM of drive.Multiple write will affect the life of MCU. Don’t suggest operate it in high frequency

Data Field	Instrucation	Data
DATA[0]	Command byte	0x63
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	encoderOffset byte 1	DATA[4] = *(uint8_t *)(&encoderOffset)
DATA[5]	encoderOffset byte 2	DATA[5] = *((uint8_t *)(&encoderOffset)+1)
DATA[6]	encoderOffset byte 3	DATA[6] = *((uint8_t *)(&encoderOffset)+2)
DATA[7]	encoderOffset byte 4	DATA[7] = *((uint8_t *)(&encoderOffset)+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.

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

Notice：

1. This command valid after re-power

2.This command will write the zero postion to ROM of drive.Multiple write will affect the life of MCU. Don’t suggest operate it in high frequency

Data Field	Instrucation	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, the data of encoderoffset is the setted zero offset value.

Data Field	Instrucation	Data
DATA[0]	Command byte	0x64
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	encoderOffset byte 1	DATA[4] = *(uint8_t *)(&encoderOffset)
DATA[5]	encoderOffset byte 2	DATA[5] = *((uint8_t *)(&encoderOffset)+1)
DATA[6]	encoderOffset byte 3	DATA[6] = *((uint8_t *)(&encoderOffset)+2)
DATA[7]	encoderOffset byte 4	DATA[7] = *((uint8_t *)(&encoderOffset)+3)

3.26 Read Multi-turn angle command (one frame)

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

Data Field	Instrucation	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，(int32_t type, range:4 bytes) unit: 0.01°/LSB.

Data Field	Instrucation	Data
DATA[0]	Command byte	0x92
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Angle low byte 1	DATA[4] = *(uint8_t *)(&motorAngle)
DATA[5]	Angle byte 2	DATA[5] = *((uint8_t *)(&motorAngle)+1)
DATA[6]	Angle byte 3	DATA[6] = *((uint8_t *)(&motorAngle)+2)
DATA[7]	Angle byte 4	DATA[7] = *((uint8_t *)(&motorAngle)+3)

3.27 Read motor status 1 and error flag (one frame)

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

Data Field	Instrucation	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.brake control command :Indicate the brake status ;(1: represents brake unlocked ;0 represents brake locked)

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

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

Data Field	Instrucation	Data
DATA[0]	Command byte	0x9A
DATA[1]	Motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	NULL	0x00
DATA[3]	brake status byte	DATA[3] = *(uint8_t *)(&RlyCtrlRslt)
DATA[4]	voltage low byte	DATA[4] = *(uint8_t *)(&voltage)
DATA[5]	voltage high byte	DATA[5] = *((uint8_t *)(& voltage)+1)
DATA[6]	error status low byte 1	DATA[6] =*(uint8_t *)(&errorState)
DATA[7]	error status byte 2	DATA[7] = *((uint8_t *)(& errorState)+1)

Memo：

1.System_errorState list 1：

System_errorState value	State explanation
0x0002	motor stall
0x0004	low voltage
0x0008	high voltage
0x0010	over current
0x0040	Main bus current error
0x0100	over speed
0x0200	position loop overflow error
0x0400	VDD error
0x0800	DSP inner sensor over temperature
0x1000	motor overheat
0x2000	encoder calibrate error

sheet 1

System_errorState list 2：

CAN_errorState value	State explanation

0x00F0	PID parameter write ROM protection, non-safe operation
0x00F1	Encoder written into ROM protection, non-safe operation
0x00F3	Brake locked
0x00F4	Motor writen ROM protection, non-safe operation

sheet2

3.28 Read motor state 2 command (one frame)

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

Data Field	Instrucation	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	Instrucation	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]	motor speed low byte	DATA[4] = *(uint8_t *)(&speed)

DATA[5]	motor speed high byte	DATA[5] = *((uint8_t *)(&speed)+1)
DATA[6]	encoder low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	encoder high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.29 Read motor state 3 command (one frame)

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

Data Field	Instrucation	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. data type : int16_t type, corresponding to the actual phase current is 1A/64LSB.

3.B phase current data, data type : 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	Instrucation	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)

3.30 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	Instrucation	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.

3.31 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	Instrucation	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.

3.32 Motor Enabled command (one frame)

After sending the command, the device will enter the initialization process(3 seconds), then clear the fault code, enter the position loop working mode (to prevent other modes from moving under external force)and open the brake at the same time.

Data Field	Instrucation	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.

3.33 Torque closed-loop control (one 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	Instrucation	Data
DATA[0]	Command byte	0xA1
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	torque current control Value low byte	DATA[4] = *(uint8_t *)(&iqControl)
DATA[5]	torque current control	DATA[5] = *((uint8_t *)(&iqControl)+1)

	Valuehigh byte	
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	Instrucation	Data
DATA[0]	Command byte	0xA1
DATA[1]	motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	torque current control Value low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	torque current control Valuehigh 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 low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	encoder 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	Instrucation	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[7] =*((uint8_t *)(&errorState)+1)

3.34 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	Instrucation	Data
DATA[0]	Command byte	0xA2
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	velocity control low byte	DATA[4] = *(uint8_t *)(&speedControl)
DATA[5]	velocity control byte	DATA[5] = *((uint8_t *)(&speedControl)+1)
DATA[6]	velocity control byte	DATA[6] = *((uint8_t *)(&speedControl)+2)
DATA[7]	velocity 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	Instrucation	Data
DATA[0]	Command byte	0xA2
DATA[1]	motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	torque current control Value low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	torque current control Valuehigh 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 low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	encoder high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.35 Multi-turn position closed loop control (one 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, the corresponding actual position is 0.01degree/LSB, 36000 represents 360°, and the direction of rotation of the motor determined by the difference between the target position and the current position.
The control value maxSpeed limits the maximum speed of motor rotation, which data type is uint16_t, corresponding to the actual speed of 1dps/LSB.

Data Field	Instrucation	Data
DATA[0]	Command byte	0xA4
DATA[1]	NULL	0x00
DATA[2]	Veocity limited low byte	DATA[2] = *(uint8_t *)(&maxSpeed)
DATA[3]	Veocity limited high byte	DATA[3] = *((uint8_t *)(&maxSpeed)+1)
DATA[4]	Angle control byte	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	Angle control byte	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	Angle control byte	DATA[6] = *((uint8_t *)(&angleControl)+2)
DATA[7]	Angle control high byte	DATA[7] = *((uint8_t *)(&angleControl)+3)

Memo:

1.The 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, range: 0~65535)

Data Field	Instrucation	Data
DATA[0]	Command byte	0xA4
DATA[1]	motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	torque current control Value low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	torque current control Valuehigh 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 low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	encoder high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.36 Incremental position loop mode （one frame）

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

The current position set as the starting point and then send incremental angle. The control value angleControl type is int32_t, corresponding actual position is 0.01degree/LSB,36000 represents 360°. Motor rotation direction determined by the incremental 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	Instrucation	Data
DATA[0]	Command byte	0xA8

DATA[1]	NULL	0x00
DATA[2]	Veocity limited low byte	DATA[2] = *(uint8_t *)(&maxSpeed)
DATA[3]	Veocity limited high byte	DATA[3] = *((uint8_t *)(&maxSpeed)+1)
DATA[4]	Angle control byte	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	Angle control byte	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	Angle control byte	DATA[6] = *((uint8_t *)(&angleControl)+2)
DATA[7]	Angle 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, range: 0~65535)

Data Field	Instrucation	Data
DATA[0]	Command byte	0xA8
DATA[1]	motor temperature	DATA[1] = *(uint8_t *)(&temperature)
DATA[2]	torque current control Value low byte	DATA[2] = *(uint8_t *)(&iq)
DATA[3]	torque current control Valuehigh 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 low byte	DATA[6] = *(uint8_t *)(&encoder)
DATA[7]	encoder high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

3.37 Read motor current working mode (one frame)

Data Field	Instrucation	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	Instrucation	Data
DATA[0]	Command	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 run mode byte	DATA[7] = *(uint8_t *)(&runmode)

3.38 Read motor power (one frame)

Data Field	Instrucation	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 reply (one frame)

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

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

3.39 Read extra battery voltage (one frame)

Read the extra battery voltage if have .

Data Field	Instrucation	Data
DATA[0]	Command	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, data type uint8_t, unit / volts, (0.1v/LSB)

Data Field	Instrucation	Data
DATA[0]	Comamnd 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]	Batvoltage high	DATA[7] = *(uint8_t *)(&batvoltage)

3.40 TF command (one frame) -Torque feedforward

This command set feedforward current, the parameters are as follows.

1.Motor speed feedforward value SpeedFeed (int16_t type, 1dps/LSB)

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

Data Field	Instrucation	Data
DATA[0]	Comamnd byte	0x73
DATA[1]	NULL	0x00
DATA[2]	Speed limited low byte	DATA[2] = *(uint8_t *)(&maxSpeed)
DATA[3]	Speed limited high byte	DATA[3] = *((uint8_t *)(&maxSpeed)+1)
DATA[4]	Encoder low byte 1	DATA[4] = *(uint8_t *)(&encoder)
DATA[5]	Encoder byte 2	DATA[5] = *((uint8_t *)(&encoder)+1)
DATA[6]	Encoder byte 3	DATA[6] = *((uint8_t *)(&encoder)+2)

DATA[7]	Encoder 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,100 times magnification, 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	Instrucation	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]	speed low byte	DATA[2] = *(uint8_t *)(&speed)
DATA[3]	Speed high byte	DATA[3] = *((uint8_t *)(&speed)+1)
DATA[4]	Encoder low byte 1	DATA[4] = *(uint8_t *)(&encoder)
DATA[5]	Encoder byte 2	DATA[5] = *((uint8_t *)(&encoder)+1)
DATA[6]	Encoder byte 3	DATA[6] = *((uint8_t *)(&encoder)+2)
DATA[7]	Encoder byte 4	DATA[7] = *((uint8_t *)(&encoder)+3)

3.41 System reset (one frame)

Data Field	Instrucation	Data
DATA[0]	Command	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

3.42 Brake unlock command (one frame)

Data Field	Instrucation	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

3.43 Brake lock command (one frame)

Data Field	Instrucation	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

3.44 Setup and read CAN ID (One 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, read :1/ write: 0.
2.CANID, range (#1~#32), uint16_t type (synchronized with myactuator GUI), device identifier 0x140 + ID (1~32).

Data Field	Instrucation	Data
DATA[0]	Command byte	0x79
DATA[1]	NULL	0x00
DATA[2]	ReadWriteFlag byte	DATA[2] = wReadWriteFlag
DATA[3]	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)

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, return to the original command.
3.Read CANID, return parameters are as follows.

Data Field	Instrucation	Data
DATA[0]	Command byte	0x79
DATA[0]	NULL	0x00
DATA[0]	ReadWriteFlag byte	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)

3.45 Read System Operate Time (One frame)

This command in order to abtain system running time,unit:ms

Data Field	Instrucation	Data
DATA[0]	Command byte	0xB1
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 data include system Runtime ,data type:uint32_t ; unit: ms

Data Field	Instrucation	Data
DATA[0]	Command byte	0xB1
DATA[0]	NULL	0x00
DATA[0]	NULL	0x00
DATA[0]	NULL	0x00
DATA[4]	SysRunTime low byte 1	DATA[4] = *(uint8_t *)(&SysRunTime)
DATA[5]	SysRunTime byte 2	DATA[5] = *((uint8_t *)(&SysRunTime)+1)
DATA[6]	SysRunTime byte 3	DATA[6] = *((uint8_t *)(&SysRunTime)+2)
DATA[7]	SysRunTime byte 4	DATA[7] = *((uint8_t *)(&SysRunTime)+3)

3.46 Read firmware version (one frame)

Send this command to learn motor firmware version .

Data Field	Instrucation	Data
DATA[0]	Command byte	0xB2
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 data include firmware version date;data type:uint32_t ; date format example :20211126

Data Field	Instrucation	Data
DATA[0]	Command byte	0xB2
DATA[0]	NULL	0x00
DATA[0]	NULL	0x00
DATA[0]	NULL	0x00
DATA[4]	VersionDate low byte 1	DATA[4] = *(uint8_t *)(&VersionDate)
DATA[5]	VersionDate byte2	DATA[5] = *((uint8_t *)(&VersionDate)+1)
DATA[6]	VersionDate byte3	DATA[6] = *((uint8_t *)(&VersionDate)+2)
DATA[7]	VersionDate byte4	DATA[7] = *((uint8_t *)(&VersionDate)+3)

3.47 Setup communication interruption protection time (one frame)

This command is used to set the communication interruption protection time, unit : ms. If the communication is interrupted for more than set time, the power output will be cut off and brake locked. To run again, you need to establish a stable and continuous communication first then send the 0x88 command. At this time, it will be locked in the position loop, you can send other commands to run other modes.

Data Field	Instrucation	Data
DATA[0]	Command byte	0xB3
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00

DATA[3]	NULL	0x00
DATA[4]	CanRecvTime_MS low byte1	DATA[4] = *(uint8_t *)(&CanRecvTime_MS)
DATA[5]	CanRecvTime_MS byte2	DATA[5] = *((uint8_t *)(&CanRecvTime_MS)+1)
DATA[6]	CanRecvTime_MS byte3	DATA[6] = *((uint8_t *)(&CanRecvTime_MS)+2)
DATA[7]	CanRecvTime_MSbyte 4	DATA[7] = *((uint8_t *)(&CanRecvTime_MS)+3)

Drive reply(one frame)

Data Field	Instrucation	Data
DATA[0]	Command byte	0xB3
DATA[0]	NULL	0x00
DATA[0]	NULL	0x00
DATA[0]	NULL	0x00
DATA[4]	CanRecvTime_MS low byte1	DATA[4] = *(uint8_t *)(&CanRecvTime_MS)
DATA[5]	CanRecvTime_MS byte2	DATA[5] = *((uint8_t *)(&CanRecvTime_MS)+1)
DATA[6]	CanRecvTime_MS byte3	DATA[6] = *((uint8_t *)(&CanRecvTime_MS)+2)
DATA[7]	CanRecvTime_MSbyte 4	DATA[7] = *((uint8_t *)(&CanRecvTime_MS)+3)

4. Multi-Motor control command

4.1 Multi-motor torque loop control command (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

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.