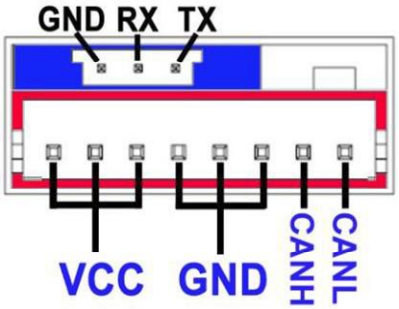
MS MF RMD-S RMD-F Motor CANBus Communication Protocol V2.35

1. CANbus parameters and single motor commands end and receive message format

Businterface: **CAN**
Baudrate: **1Mbps**
The format of message used to send commands and motor replies to asingle motor is as follows:
ID: **0x140+ID(1~32)**
Frame format: **DATA**
Frame type: **Standard**DLC:**8Byte**



MS MF 120 70 90 CAN Motor pins specifications

The following is a sample to send a **running** command to the #1 motor.

ID	Len	Data0	D1	D2	D3	D4	D5	D6	D7	Format	Type
141	8	88	00	00	00	00	00	00	00	Standard	Data

The following is a sample to send a **stop** command to the #1 motor.

ID	Len	Data0	D1	D2	D3	D4	D5	D6	D7	Format	Type
141	8	81	00	00	00	00	00	00	00	Standard	Data

The following is a sample to send a **Speed closed loop control** command to the #1 motor running at 90dps.

ID	Len	Data0	D1	D2	D3	D4	D5	D6	D7	Format	Type
141	8	A2	00	00	00	28	23	00	00	Standard	Data

The following is a sample to send a **Position + Speed closed loop control** command to the #1 motor to run at speed 90dps to 720degree

ID	Len	Data0	D1	D2	D3	D4	D5	D6	D7	Format	Type
141	8	A4	00	28	23	40	19	01	00	Standard	Data

2. Single motor command list

The current CAN control commands supported by RMD Motor drives are as follows:

	Description	Command
1	Read the command for PID parameters	0x30
2	Write the PID to the RAM command	0x31
3	Write the PID to the ROM command	0x32
4	Read acceleration data	0x33
5	Write acceleration data	0x34
6	Read encoder command	0x90
7	Write encoder value to ROM as motor zero command	0x91
8	Write current position to ROM as motor zero command	0x19
9	Read multi-turn angle command	0x92
10	Read single circle angle command	0x94
11	Clear motor angle command (set initial position)	0x95
12	Read motor status1 and error flag commands	0x9A
13	Clear motor error flag command	0x9B
14	Read motor status2 command	0x9C
15	Read motor status3 command	0x9D
16	Motor off command	0x80
17	Motor stop command	0x81
18	Motor running command	0x88
19	Torque closed loop control command	0xA1
20	Speed closed loop control command	0xA2
21	Multi loop Angle control command	0xA3
22	Multi loop Angle & speed control	0xA4
23	Single loop Angle control command	0xA5
24	Single loop Angle & speed control	0xA6
25	Increasement Angle control	0xA7
26	Increasement Angle & speed	0xA8

(20) Position closed loop control command2 (1frame)The host sends this command to control the position of the motor (multi-turn angle).The control value angleControl is int32_t, the actual position is 0.01degree/LSB, that is,36000 represents 360°,and the motor rotation direction is determined by the difference between the target position and the current position. The control value maxSpeed limits the maximum speed at which the motor rotates, of the uint16_t type, corresponding to the actual speed of 1dps/LSB.

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

Notes：

1. The control value angle Control 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 MaxAcceleration value in the host computer.
3. In this control mode, the maximum torque current of the motor is limited by the MaxTorqueCurrent value in the host computer.

Drive reply(1 frame)

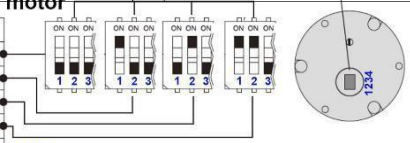
The motor responds to the host after receiving the command. The frame data contains the following parameters.

1. Motor temperature (int8_ttype,1°C/LSB).
2. The torque current value of the motor is iq(int16_ttype, range-2048~2048,corresponding to the actual torque current range-33A~33A).
3. Motor speed(int16_ttype,1dps/LSB).
4. Encoder position value encoder (uint16_ttype,14-bit encoder value range0~16383).

Data field	Description	D
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 b	DATA[7]=*((uint8_t*)(&encoder)+1)

Set address for each motor

ID	ADDR0	ADDR1	ADDR2
#1	OFF	OFF	OFF
#2	ON	OFF	OFF
#3	OFF	ON	OFF
#4	ON	ON	OFF
#5	OFF	OFF	ON
#6	ON	OFF	ON
#7	OFF	ON	ON
#8	ON	ON	ON



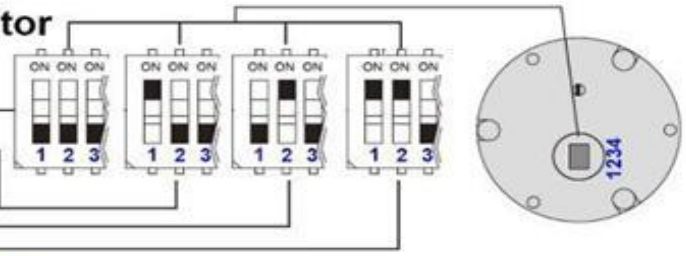
Note:
The 4th DIP switch is for CAN bus terminal resistance
Bit 4 ON means that the resistance (120Ω) is connected.
Usually the last CAN bus terminal needs to be connected to a terminal resistance 120Ω.

SMCPowersLtd.
www.smc-powers.com

1.Brief Introduction

Set address for each motor

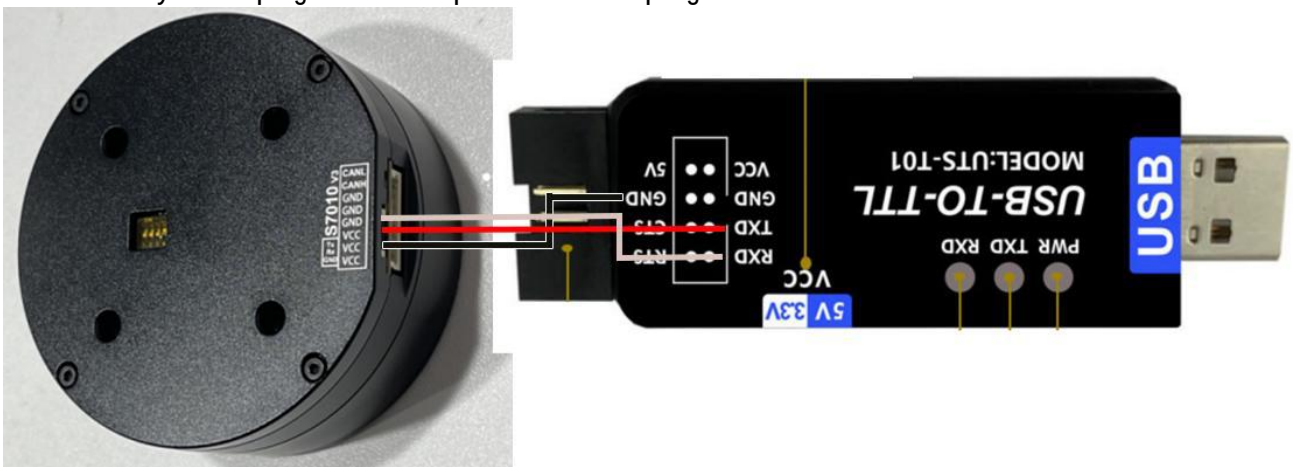
ID	ADDR1	ADDR2	ADDR3
#1	OFF	OFF	OFF
#2	ON	OFF	OFF
#3	OFF	ON	OFF
#4	ON	ON	OFF
#5	OFF	OFF	ON
#6	ON	OFF	ON
#7	OFF	ON	ON
#8	ON	ON	ON

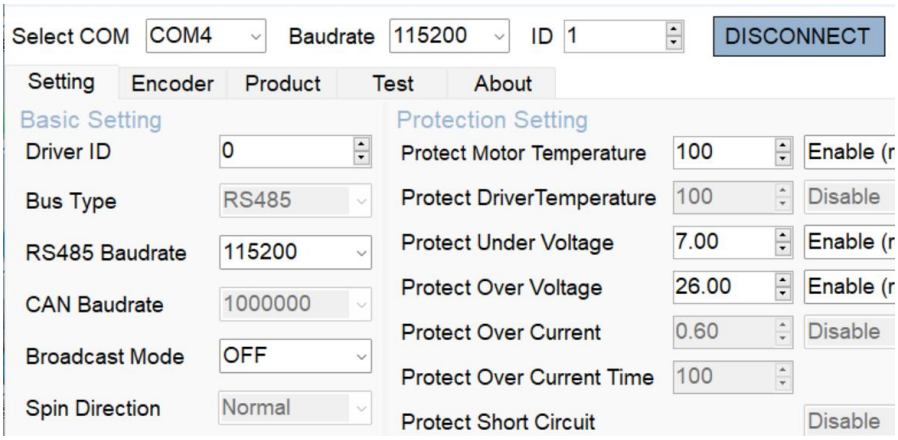


Note:
The 4th DIP switch is for CAN bus terminal resistance
Bit 4 ON means that the resistance (120Ω) is connected.
Usually the last CAN bus terminal needs to be connected to a terminal resistance 120Ω.

Connect to PC program.

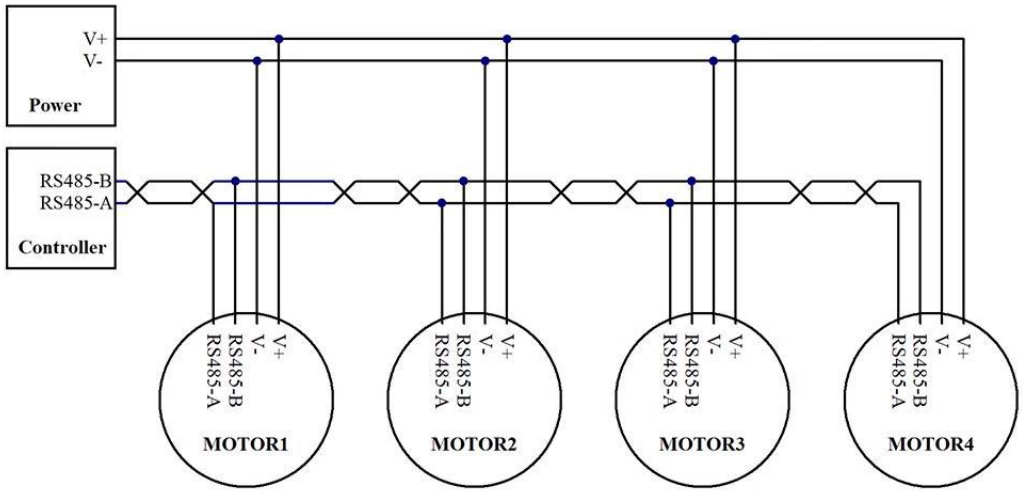
You can use USB to UART Adaptor with Molex 1.25 connection 3pin wires to connect to the driver on Tx and Rx socket. Then you can plug the PC USB port to connect program to the driver.



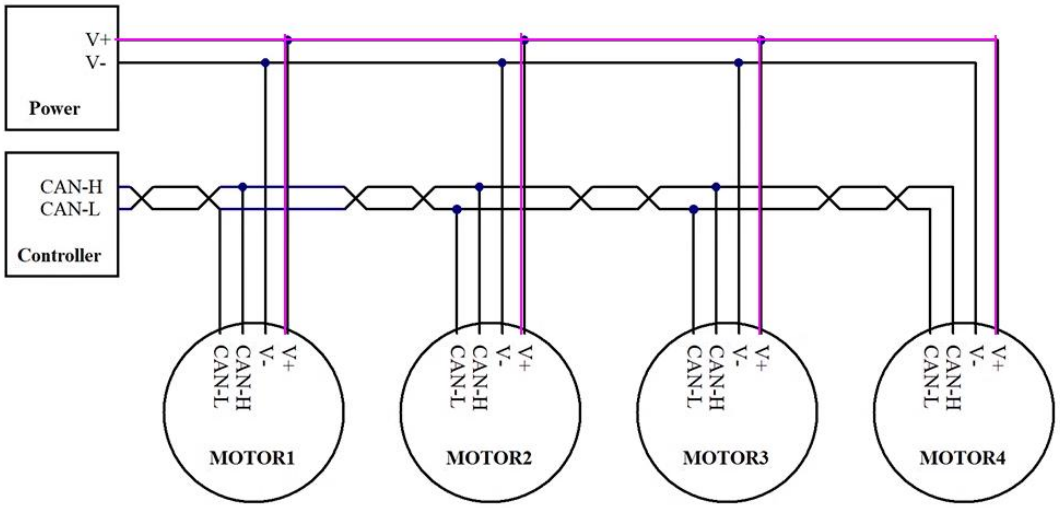


Connect to Power and RS485 or CAN.

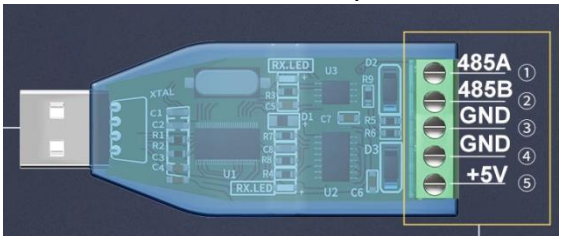
The following is the sample for connect to power and RS485.



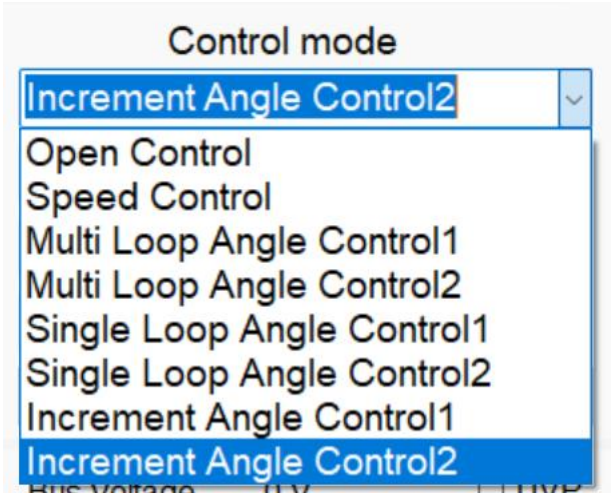
The following is the sample for connect to power and CAN.



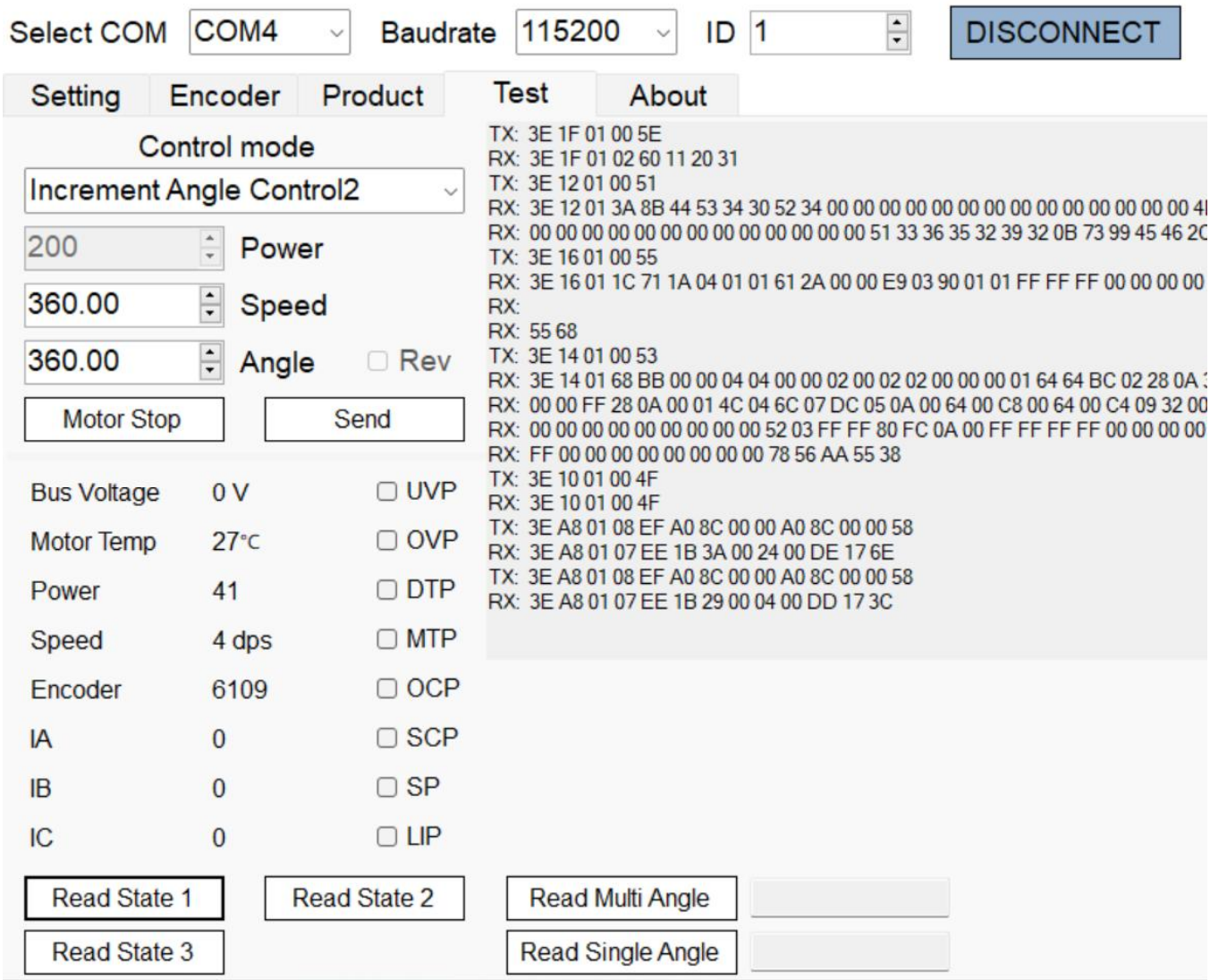
You can use USB to UART Adapter to connect the driver to PC.



You can also test running the motor on the PC program with following control mode.



The following is a copy of test page.



2.Motor receiving message format

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

3.Control Command list

SN	COMMANDNAME	COMMANDDATA
1.	Read the command for setting parameters	0x30
2.	Write the Settings to the RAM command	0x31
3.	Write the Settings to the ROM command	0x32
4.	Read acceleration data command	0x33
5.	Write acceleration data to RAM command	0x34
6.	Read encoder data command	0x90
7.	Write encoder off set command	0x91
8.	Write current position to ROM as motor zero command	0x19
9.	Read multi turns angle command	0x92
10.	Read single circle angle command	0x94
11	Make motor current position as zero position	0x95
12	Readmotorstatus1anderrorflagcommands	0x9A
13	Clearmotor error flag command	0x9B
14.	Read motor status2	0x9C
15.	Read motor status3	0x9D
16.	Motor off command	0x80
17	Motor stop command	0x81
18.	Motor running command	0x88
19	Torque open-loop command	0xA0
20.	Torque closed-loop command	0xA1
21	Speed closed-loop command	0xA2
22	Multi loop Angle control command	0xA3
23	Multi loop Angle & speed control	0xA4
24	Single loop Angle control command	0xA5
25	Single loop Angle & speed control	0xA6
26	Increasement Angle control	0xA7
27	Increasement Angle & speed control	0xA8

4.Single motor Command description

4.1. Read PID parameters command

The host sends this command to read the PID parameters of the current motor.

Datafield	Description	Data
DATA[0]	commandbyte	0x30
DATA[1]		00

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

For example: 0x30 00 00 00 00 00 00 00 to read PID

Driver reply(one frame)

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

Datafield	Description	Data
DATA[0]	commandbyte	0x30
DATA[1]		00
DATA[2]	Angle loop Kp	DATA[2] = anglePidKp
DATA[3]	Angle loop Ki	DATA[3] = anglePidKi
DATA[4]	Speed loop Kp	DATA[4] = speedPidKp
DATA[5]	Speed loop Ki	DATA[5] = speedPidKi
DATA[6]	Torque loop Kp	DATA[6] = iqPidKp
DATA[7]	Torque loop Ki	DATA[7] = iqPidKi

For example: 0x30 00 64 64 32 28 32 32

Angle Kp = 0x64 = 100, Angle Ki = 0x64 = 100,

Speed Kp = 0x32 = 50, Speed Ki = 0x28 = 40,

Torque Kp = 0x32 = 50, Torque Ki = 0x32 = 50.

4.2 Write the PID Settings to the RAM command

The host sends this command to write the PID parameter to RAM, and the write parameter becomes invalid after power off. See motor setting parameters for the data sent

Datafield	Description	Data
DATA[0]	Commandbyte	0x31
DATA[1]		00
DATA[2]	Angle loop Kp	DATA[2] = anglePidKp
DATA[3]	Angle loop Ki	DATA[3] = anglePidKi
DATA[4]	Speed loop Kp	DATA[4] = speedPidKp
DATA[5]	Speed loop Ki	DATA[5] = speedPidKi
DATA[6]	Torque loop Kp	DATA[6] = iqPidKp
DATA[7]	Torque loop Ki	DATA[7] = iqPidKi

Driver reply (one frame)

The motor responds to the host after receiving the command.

Datafield	Description	Data
DATA[0]	commandbyte	0x31
DATA[1]		00
DATA[2]	Angle loop Kp	DATA[2] = anglePidKp
DATA[3]	Angle loop Ki	DATA[3] = anglePidKi
DATA[4]	Speed loop Kp	DATA[4] = speedPidKp
DATA[5]	Speed loop Ki	DATA[5] = speedPidKi
DATA[6]	Torque loop Kp	DATA[6] = iqPidKp
DATA[7]	Torque loop Ki	DATA[7] = iqPidKi

4.3 Writes the set PID parameters to the ROM command

The host sends this command to write the PID parameters to the ROM. The parameters are still valid after power off. See motor setting parameters for the data sent

Datafield	Description	Data
DATA[0]	Commandbyte	0x32
DATA[1]		00
DATA[2]	Angle loop Kp	DATA[2] = anglePidKp
DATA[3]	Angle loop Ki	DATA[3] = anglePidKi
DATA[4]	Speed loop Kp	DATA[4] = speedPidKp
DATA[5]	Speed loop Ki	DATA[5] = speedPidKi
DATA[6]	Torque loop Kp	DATA[6] = iqPidKp
DATA[7]	Torque loop Ki	DATA[7] = iqPidKi

For example: 0x32 00 64 63 32 27 33 31
Angle Kp = 0x64 = 100, Angle Ki = 0x63 = 99,
Speed Kp = 0x32 = 50, Speed Ki = 0x27 = 39,
Torque Kp = 0x33 = 51, Torque Ki = 0x31 = 49.

Driver reply(one frame)

The motor responds to the host after receiving the command.

Datafield	Description	Data
DATA[0]	Commandbyte	0x32
DATA[1]		00
DATA[2]	Angle loop Kp	DATA[2] = anglePidKp
DATA[3]	Angle loop Ki	DATA[3] = anglePidKi
DATA[4]	Speed loop Kp	DATA[4] = speedPidKp
DATA[5]	Speed loop Ki	DATA[5] = speedPidKi
DATA[6]	Torque loop Kp	DATA[6] = iqPidKp
DATA[7]	Torque loop Ki	DATA[7] = iqPidKi

4.4. Read acceleration data command(oneframe)

The host sends the command to read the acceleration data. Acceleration data Accel is int32_t type, unit 1dps/s

Datafield	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 motor responds to the host after receiving the command as follows. Acceleration data Accel is int32_t type, unit 1dps/s

Datafield	Description	Data
DATA[0]	Command byte	0x33
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00

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

4.5. Write acceleration data to RAM command(oneframe)

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

Datafield	Description	Data
DATA[0]	Command byte	0x34
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Acceleration	DATA[4]=*(uint8_t*)(&Accel)
DATA[5]	Acceleration	DATA[5] =*((uint8_t*)(&Accel)+1)
DATA[6]	Acceleration	DATA[6] =*((uint8_t*)(&Accel)+2)
DATA[7]	Acceleration	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

4.6 Read encoder data command (one frame)

The host sends the command to read the current position of the encoder

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 framed at a contains the following parameters.

1. Encoder current position(uint16_t type, 14bit encoder value range0~16383), which is the encoder original position minus the encoder zero off set value
2. Encoder original position (encoder Raw) (uint16_t type, 14bit encoder value range0~16383)
3. Encoder Offset（uint16_t type, 14bit encoder value range0~16383）

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 offset low byte	DATA[6]=*(uint8_t*)(&encoderOffset)
DATA[7]	Encoder offset high byte	DATA[7]=*((uint8_t*)(&encoderOffset)+1

4.7 Write encoder offset command(one frame)

The host sends the command to set encoder offset, written the encoder offset with uint16_t type, 14bit encoder value range0~16383

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 off set low byte	DATA[6]=*(uint8_t*)(&encoderOffset)
DATA[7]	Encoder off set 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 contains the following parameters.

1. New encoder Offset（uint16_t type, 14bit encoder value range0~16383）

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 offset low byte	DATA[6]=*(uint8_t*)(&encoderOffset)
DATA[7]	Encoder offset high byte	DATA[7]=*((uint8_t*)(&encoderOffset)+1)

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

Notice:

1. This command needs to be powered on again to take effect.
2. This command will write the zero position to ROM. Multiple writes will affect the chip life. So it is not recommended to use it frequently.

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

Driver reply(one frame)

The motor returns to the host after receiving the command, and the data is 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 offset low byte	DATA[6]=*(uint8_t*)(&encoderOffset)
DATA[7]	Encoder offset high byte	DATA[7]=*((uint8_t*)(&encoderOffset)+1)

4.9 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.

1.Motorangle, int64_t type data, positive value indicates clockwise cumulative angle, negative value indicates counterclockwise cumulative angle, unit 0.01°/LSB.

Data field	Description	Data
DATA[0]	Command byte	0x92
DATA[1]	Anglelowbyte1	DATA[1]=*(uint8_t*)(&motorAngle)
DATA[2]	Anglebyte2	DATA[2] =*((uint8_t*)(&motorAngle)+1)
DATA[3]	Anglebyte3	DATA[3] =*((uint8_t*)(&motorAngle)+2)
DATA[4]	Anglebyte4	DATA[4] =*((uint8_t*)(&motorAngle)+3)
DATA[5]	Anglebyte5	DATA[5] =*((uint8_t*)(&motorAngle)+4)
DATA[6]	Anglebyte6	DATA[6] =*((uint8_t*)(&motorAngle)+5)
DATA[7]	Anglebyte7	DATA[7] =*((uint8_t*)(&motorAngle)+6)

4.10 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.

1.CircleAngle, uint16_t type data, starting from the encoder zero point, increased by clockwise rotation, and returning to zero when it reaches zero again, the unit is 0.01°/LSB.

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)

4.11 Make motor current position as zero position

The command makes the current position as zero position, clear single angle and multi angle data. It will be invalid when power off.

Data field	Description	Data
DATA[0]	Command byte	0x95
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 the host sent

4.12 Read motor status1 and error flag commands(one frame)

This command reads the motor's error status and voltage, temperature and other information.

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. ErrorState（uint8_t type, Each bit represents a 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]	Voltage low byte	DATA[3]=*(uint8_t*)(&voltage)
DATA[4]	Voltage high byte	DATA[4]=*((uint8_t*)(&voltage)+1)
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	Error State byte	DATA[7]=errorState

Memo：

Error State:The specific status table of each bit is as follows

ErrorState	Statusde scription	0	1
0	Voltage status	Normal	Low voltage protection
1	invalid		
2	invalid		
3	Temperature status	Normal	Over temperature protection
4	invalid		
5	invalid		
6	invalid		
7	invalid		

4.13 Clear motor error flag command (one frame)

This command clears the error status of the current motor

Data field	Description	Data
DATA[0]	Command byte	0x9B
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. ErrorState（uint8_t type, Each bit represents a 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]	Voltage low byte	DATA[3] =*(uint8_t*)(&voltage)
DATA[4]	Voltage high byte	DATA[4]=*((uint8_t*)(&voltage)+1)
DATA[5]	NULL	0x00
DATA[6]	NULL	0x00
DATA[7]	Error State byte	DATA[7]=errorState

Memo: the error flag cannot be cleared when the motor status does not return to normal.

ErrorState: the specific status table of each bit is as follows

Error State	Status description	0	1
0	Voltage status	Normal	Low voltage protection
1	invalid		
2	invalid		
3	Temperature status	Normal	Over temperature protection
4	invalid		
5	invalid		
6	invalid		
7	invalid		

4.14 Read motor status2 (one frame)

This command reads motor temperature, voltage, speed, encoder position

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, unit 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, 14 bit encoder value range 0~16383）

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)

4.15 Read motor status3(one frame)

This command reads the phase current status 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 returns to the host after receiving the command. The frame data contains three-phase current data, the data type is int16_t type, corresponding to the actual phase current is 1A/64LSB.

Datafield	Description	Data
DATA[0]	Command byte	0x9D
DATA[1]	NULL	0x00
DATA[2]	PhaseA current low byte	DATA[2]=*(uint8_t*)(&iA)
DATA[3]	PhaseA current high byte	DATA[3]=*((uint8_t*)(&iA)+1)
DATA[4]	PhaseB current low byte	DATA[4]=*(uint8_t*)(&iB)
DATA[5]	PhaseB current high byte	DATA[5]=*((uint8_t*)(&iB)+1)
DATA[6]	PhaseC current low byte	DATA[6]=*(uint8_t*)(&iC)
DATA[7]	PhaseC current high byte	DATA[7]=*((uint8_t*)(&iC)+1)

4.16 Motor off command(one frame)

Turn off motor, while clearing the motor operating status and previously received control commands

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 the host sent

4.17 Motor stop command(one frame)

Stop motor, but do not clear the motor operating state and 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 the host sent

4.18 Motor running command(one frame)

Resume motor operation from motor stop command (Recovery control mode before stop motor)

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 the host sent

5.1 Torque open-loop command (one frame)

The host sends the command to control torque current output of the motor. powerControl is int16_t type, the value range:-850~850, (the bus current and the actual torque of the motor vary with different motors)

Data field	Description	Data
DATA[0]	Command byte	0xA0
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Power value low byte	DATA[4] =*(uint8_t*)(& powerControl)
DATA[5]	Torque value high byte	DATA[5]=*((uint8_t*)(& powerControl)+1)
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Memo：

1. powerControl int his command is not limited by the MaxTorque Current value in the host computer.

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,unit1℃/LSB）
2. Motor Power（int16_t type, Range:-850~850）
3. Motor speed（int16_t type,1dps/LSB）
4. Encoder position value（uint16_t type, 14bit encoder value range0~16383）

Data field	Description	Data
DATA[0]	Command byte	0xA0
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 byte high	DATA[7] =*((uint8_t*)(&encoder)+1)

5.2 Torque closed-loop command (one frame)

The host sends the command to control torque current output of the motor. IqControl is int16_t type, the value range:-2048~2048, corresponding to the actual torque current range -16.5A~16.5A for MF, RMD-L motors, -32A~32A for MG motors (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]	Torque current low byte	DATA[4] =*(uint8_t*)(&iqControl)
DATA[5]	Torque current high byte	DATA[5]=*((uint8_t*)(&iqControl)+1)
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Memo：

2. IqControl int his command is not limited by the MaxTorque Current value in the host computer.

Drive reply（one frame）

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

5. Motor temperature（int8_t type,unit1℃/LSB）
6. Motor torque current(Iq)（int16_t type, Range:-2048~2048, real torque current range -16.5A~16.5A for MF, RMD-L motors, -32A~32A for MG motors）
7. Motor speed（int16_t type,1dps/LSB）
8. Encoder position value（uint16_t type, 14bit encoder value range0~16383）

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]	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 byte high	DATA[7] =*((uint8_t*)(&encoder)+1)

5.3 Speed closed-loop command (one frame)

The host sends this command to control the speed of the motor. SpeedControl is int32_t, which corresponds 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 under this command is limited by the Max Torque Current 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.

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. 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, 14bit encoder value range 0~16383）

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]	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)

5.4 Multi loop Angle control command(one frame)

The host sends this command to control the position of the motor (multi-turn angle). AngleControl is int32_t type, and the actual position is 0.01degree/LSB, 36000 represents 360°. The motor rotation direction 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[3]=*(uint8_t*)(&angleControl)
DATA[5]	Position control	DATA[4]=*((uint8_t*)(&angleControl)+1)
DATA[6]	Position control	DATA[5]=*((uint8_t*)(&angleControl)+2)
DATA[7]	Position control high byte	DATA[6]=*((uint8_t*)(&angleControl)+3)

Memo：

1. Angle Control under this command is limited by the Max Angle value in the host computer.
2. The maximum speed under this command is limited by the MaxSpeed 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 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,unit1℃/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, 14bit encoder value range 0~16383）

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]	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)

5.5 Multi loop Angle & speed control command (one frame)

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

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

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

Memo：

1. Angle Control under this command is limited by the MaxAngle value in the host computer.
2. In this control mode, the maximum acceleration of the motor is limited by the MaxAcceleration value in the host computer.
3. In this control mode, the maximum torque current of the motor is limited by the MaxTorque Current value in the host computer.

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,unit1℃/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, 14bit encoder value range0~16383）

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]	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)

5.6 Single loop Angle control command (one frame)

The host sends this command to control the position of the motor (single-turn angle). AngleControl is uint16_t type, the value range is 0~35999, and the actual position is 0.01degree/LSB, the actual angle range is 0°~359.99° The control values pinDirection sets the direction in which the motor rotates, which is uint8_t type, 0x00 for clockwise and 0x01 for counterclockwise.

Data field	Description	Data
DATA[0]	Command byte	0xA5
DATA[1]	SpinDirection 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 under this command is limited by the MaxSpeed 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 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,unit1℃/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, 14bit encoder value range 0~16383）

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]	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)

5.7 Single loop Angle & speed control (one frame)

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

1. AngleControl is uint16_t type, the value range is 0~35999, and the actual position is 0.01degree/LSB, the actual angle range is 0°~359.99°。
2. The control value spinDirection sets the direction in which the motor rotates, which is uint8_t type, 0x00 for clockwise and 0x01 for counterclockwise.
3. Max Speed limits the maximum speed of motor rotation, which is uint16_t type, corresponding to the actual speed of 1dps/LSB.

Data field	Description	Data
DATA[0]	Command byte	0xA6
DATA[1]	SpinDirection byte	DATA[1]=spinDirection
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]	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 MaxAcceleration value in the host computer.
2. In this control mode, the maximum torque current of the motor is limited by the MaxTorque Current value in the host computer.

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,unit1℃/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, 14bit encoder value range 0~16383）

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]	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)

5.8 Increasement Angle control (one frame)

The host sends this command to control the angle increasement of the motor

AngleControl is int32_t type, the actual position is 0.01degree/LSB. The direction of rotation is determined by positive or negative.

Datafield	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：

In this control mode, the maximum acceleration of the motor is limited by the Max Acceleration value in the PC setting tool.

In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the PC setting tool.

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,unit1℃/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, 14bit encoder value range 0~16383）

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]	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)

5.9 Increasement Angle & Speed control (one frame)

The host sends this command to control the angle increasement of the motor

AngleControl is int32_t type, the actual position is 0.01degree/LSB. The direction of rotation is determined by positive or negative.

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

Data field	Description	Data
DATA[0]	Command byte	0xA8
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]	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)

The following is a sample for sending the command to make the motor to rotate 30 degrees at 36dps
30 degrees = 3000 = 0x0B B8
dps is Degree Per Second, ^{0}/s. 1 dps is equal to **0.16666666666667 rpm.** 1 rad = $360^0/(2*3.14159) = 57^0 17'45"$.
36dps = 3600 = 0x0E 10

ID	Len	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Format	Type
141	8	A8	00	10	0E	B8	0B	00	00	Standard	Data
		Cmd		Speed 0.01dps/LSB		Increasing Angle 0.01degree/LSB					

The following is a sample for sending the command to make the motor to rotate -30 degrees at 36dps

ID	Len	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	Format	Type
141	8	A8	00	10	0E	48	F4	FF	FF	Standard	Data
		Cmd		Speed 0.01dps/LSB		Increasing Angle 0.01degree/LSB					

Memo：

In this control mode, the maximum acceleration of the motor is limited by the MaxAcceleration value in the PC setting tool.

In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the PC setting tool.

Drive reply(one frame)

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

1. Motor temperature（int8_t type,unit1℃/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, 14bit encoder value range 0~16383）

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]	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)

6 Multi motors control command

6.1 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 sends this command to control the torque current output up to 4motors. 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).

Data field	Description	Data
DATA[0]	Torque current1 control value low byte	DATA[0]=*(uint8_t*)(&iqControl_1)
DATA[1]	Torque current1 control value high byte	DATA[1]=*((uint8_t*)(&iqControl_1)+1)
DATA[2]	Torque current2 control value low byte	DATA[2]=*(uint8_t*)(&iqControl_2)
DATA[3]	Torque current2 control value high byte	DATA[3]=*((uint8_t*)(&iqControl_2)+1)
DATA[4]	Torque current3 control value low byte	DATA[4]=*(uint8_t*)(&iqControl_3)
DATA[5]	Torque current3 control value high byte	DATA[5]=*((uint8_t*)(&iqControl_3)+1)
DATA[6]	Torque current4 control value low byte	DATA[6]=*(uint8_t*)(&iqControl_4)
DATA[7]	Torque current4 control value high byte	DATA[7]=*((uint8_t*)(&iqControl_4)+1)

6.2 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.

Note: We have different coder type from 12bit to 18bit, so the reply encoder value range varies from motor to motor. It depends on which type you purchased.