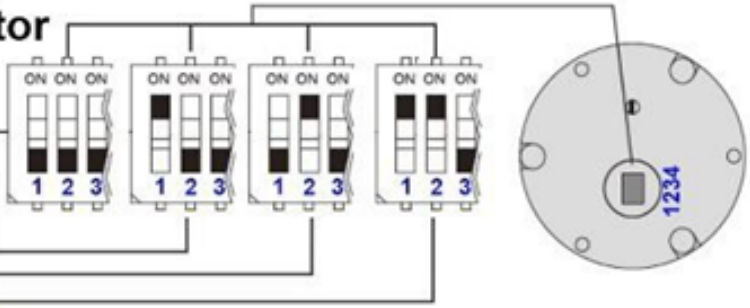
MG RMD MOTOR CONTROL PROTOCOL（CAN BUS）V2.3

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.

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	COMMAND NAME	COMMAND DATA
1.	Read PID data command	0x30
2.	Write PID to RAM command	0x31
3.	Write PID to 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 offset 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	Read motor status 1 and error flag commands	0x9A
13	Clear motor error flag command	0x9B
14.	Read motor status 2	0x9C
15.	Read motor status 3	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

- **Read PID parameter command(one frame)**

  The host sends the command to read the current PID parameters.

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

Drive reply(one frame)

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

Data field	Description	Data
DATA[0]	command byte	0x30
DATA[1]	NULL	0x00
DATA[2]	Position loop Kp	DATA[2] = anglePidKp
DATA[3]	Position 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

- **Write PID to RAM parameter command (one frame)**

The host sends the command to write the PID parameters to the RAM, parameters are invalid when power turned off.

Data field	Description	Data
DATA[0]	Command byte	0x31
DATA[1]	NULL	0x00
DATA[2]	Position loop KP	DATA[2] = anglePidKp
DATA[3]	Position 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

Drive reply(one frame)

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

- **Write PID to ROM parameter command (one frame)**

The host sends the command to write the PID parameters to the ROM, the parameter still valid when power turned off.

Data field	Description	Data
DATA[0]	Command byte	0x32
DATA[1]	NULL	0x00
DATA[2]	Position loop KI	DATA[2] = anglePidKp
DATA[3]	Position 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

Drive reply(one frame)

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

- **Read acceleration data command (one frame)**

The host send the command to read motor acceleration data

Data field	Description	Data

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

Drive reply（one frame）

The driver reply data include acceleration data, data type :int32_t, unit:1dps/s

Data field	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)

- **Write acceleration data to RAM command (one frame)**

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

Data field	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

- **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 frame data contains the following parameters.

1. Encoder current position (uint16_t type, 14bit encoder value range 0~16383), which is the encoder original position minus the encoder zero offset value
2. Encoder original position (encoderRaw) (uint16_t type, 14bit encoder value range 0~16383)
3. EncoderOffset（uint16_t type, 14bit encoder value range 0~16383）

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]	Encoderposition 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)

- **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 range 0~16383

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)

Drive reply (one frame)

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

1. New encoderOffset（uint16_t type, 14bit encoder value range 0~16383）

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)

- **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 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)

- **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. Motor angle, 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]	Angle low byte 1	DATA[1] = *(uint8_t *)(&motorAngle)
DATA[2]	Angle byte 2	DATA[2] = *((uint8_t *)(& motorAngle)+1)
DATA[3]	Angle byte 3	DATA[3] = *((uint8_t *)(& motorAngle)+2)
DATA[4]	Angle byte 4	DATA[4] = *((uint8_t *)(& motorAngle)+3)
DATA[5]	Angle byte 5	DATA[5] = *((uint8_t *)(& motorAngle)+4)
DATA[6]	Angle byte 6	DATA[6] = *((uint8_t *)(& motorAngle)+5)

DATA[7]	Angle byte 7	DATA[7] = *((uint8_t *)(& motorAngle)+6)

- **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)

- **Make motor current position as zero position (not available yet)**

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

- **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

- **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

- **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

- **Read motor status 1 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. Error State（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]=error State

Memo：

Error State: 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		

- **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 rply (one frame)

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

1. Motor temperature（int8_t type,unit 1℃/LSB）
2. Voltage（uint16_t type,unit 0.1V/LSB）
3. Error State（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.

Error State: 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		

- **Read motor status 2 (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,14bit 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]	Encoderposition high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

- **Read motor status 3 (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.

Data field	Description	Data
DATA[0]	Command byte	0x9D
DATA[1]	NULL	0x00
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)

- **Torque current control command(one frame)**

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

Data field	Description	Data
DATA[0]	Command byte	0xA1
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Torque current control value low byte	DATA[4] = *(uint8_t *)(&iqControl)
DATA[5]	Torque current control value high byte	DATA[5] = *((uint8_t *)(&iqControl)+1)
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Memo：

1. Iq Control in this command is not 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,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	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 high byte	DATA[7] = *((uint8_t *)(&encoder)+1)

- **Speed control command (one frame)**

The host sends this command to control the speed of the motor. Speed Control 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)

- **Multi loop Angle control command(one frame)**

The host sends this command to control the position of the motor (multi-turn angle).Angle Control 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 Max Speed value in the host computer.
3. In this control mode, the maximum acceleration of the motor is limited by the Max Acceleration value in the host computer.
4. In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the host computer.

Drive 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	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)

- **Multi loop Angle & speed control command (one frame)**

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

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 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 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	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)

- **Single loop Angle control command (one frame)**

The host sends this command to control the position of the motor (single-turn angle). Angle Control 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 value spin Direction 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]	Spin Direction byte	DATA[1] = spinDirection
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	position control low byte	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	position control high byte	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	NULL	0x00
DATA[7]	NULL	0x00

Memo：

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

Drive 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	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)

- ## Single loop Angle & speed control (one frame)

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

1. Angle Control 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 spin Direction 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]	Spin Direction 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 Max Acceleration value in the host computer.
2. In this control mode, the maximum torque current of the motor is limited by the Max Torque Current value in the host computer.

Drive 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	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)

• Increasement Angle control (one frame)

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

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

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

Memo：

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.

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

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

Motor speed（int16_t type,1dps/LSB）

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)

• Increasement Angle & Speed control (one frame)

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

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

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	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)

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.

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

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

Motor speed（int16_t type,1dps/LSB）

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)

4.Multi motors control command

- **Multiple motor torque closed loop control commands(one frame)**

  The format of the message used to send commands to multiple motors at the same time, as followed:

  Identifier：0x280　　Frame format：DATA
  Frame type: standard frame　　DLC：8byte

  The host simultaneously send this command to control the torque current output up to 4 motors. The control value iqControl is int16_t type, the value range is -2000~2000, corresponding to the actual torque current range -32A~32A (The bus current and the actual torque of the motor vary from motor to motor).

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

- **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 differ encoder type from 12bit to 18bit , so the reply encoder value range vary from motor to motor. It depands on which type you purchased.