SHANGHAI LINGKONG TECHNOLOGY CO.,LTD

CAN PROTOCOL V2.35

Catalog

Disclaimer

Thank you for purchasing motor and driver integrated system from Shanghai LingKong Technology Co.,Ltd. Please read this statement carefully before using our product. Please read this statement carefully before using.It's considered to be the recognition and acceptance of the entire statement once using.Please ensure all the manual,relevant laws,regulations and policies are strictly observed when you operate the product.The user take responsibility for his own behavior during the process.We will not be liable for any loss caused by improper use,improper installation and modification by users.

LK-TECH is a trademark of Shanghai LingKong Technology Co.,Ltd.The product names, brands, etc. appearing in this article are trademarks or registered trademarks belong to the company.

This product and manual copyright belong to LK-TECH. Reproduction in any form may not be made without permission.

The final interpretation of the statement is owned by Shanghai Lingkong Technology Co., Ltd.

1.CAN bus parameters

Interface:CAN

Baud rate:(Normal mode,single motor command):

1Mbps(default)

500kbps

250kbps

125kbps

100kbps

Baud rate(broadcast mode,multi motor command):

1Mbps

500kbps

2.Single motor command

Up to 32 (depending on the bus load) can be mounted on the same bus, in order to prevent bus conflicts, each driver needs to set a different ID, ID number 1~32.

The master sends a single-motor command frame to the bus, and the corresponding ID motor executes after receiving the command, and sends a reply frame with the same ID to the master after a period of time (within 0.25ms). The command frame and reply frame message format are as follows:

Identifier:0x140 + ID(1~32)

Frame format: data frame

Frame type: standard frame

DLC:8bytes

1. Motor off command

Switch the motor from the on state (the default state after power-on) to the off state, clear motor turns and the earlier command. The LED changes from always on to slow flashing. The motor can still reply to commands, but will not perform actions.

Data field	Instruction	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

Driver respond

Same as the host sending command.

2. Motor on command

Swift the motor from off state to on state, LED changes from slow flashing to always on.We can send commands to control the motor now.

Data field	Instruction	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

Driver respond

Same as the host sending command.

3. Motor stop command

Stop motor but don't clear the motor state.Send commands again can control the motor.

Data field	Instruction	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

Driver respond(1 frame)

Same as the host sending command.

4. Open loop control command(The command can only be applied to MS series motor)

Host send commands to control the open loop voltage. PowerControl value is int16_t, range is -850~850,(Motor current and torque is different depends on motor).

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

Remark:

1. In this command, powerControl value isn't limited by Max Power in LK motor tool.

Driver respond(1 frame)

Motor reply to the host after receiving the commands. The frame data include the following parameters:

1. Motor temperature,int8_t,1℃/LSB.
2. Motor output power,int16_t, range is -850~850.
3. Motor speed,int16_t,1dps/LSB.
4. Encoder position,uint16_t, 15bit encoder range is 0~32767;18bit encoder range is 0~65535(keeping high 16bit, Omit the lower 2 bit).

Data field	Instruction	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 *)(&power)
DATA[3]	Torque current high byte	DATA[3] = *((uint8_t *)(&power)+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. Torque closed loop control command(the command can only be applied to MF,MH,MG series)

Host send commands to control the torque current output, iqControl value is int16_t, range is -2048~2048, corresponding MF motor actual torque current range is -16.5A~16.5A, corresponding MG motor actual torque current range is -33A~33A. The bus current and the actual torque of the motor vary from motor to motor.

Data field	Instruction	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

Remark:

1. In this command, iqControl is not limited by Max Torque Current of LK motor tool.

Driver respond

Motor reply to the host after receiving the commands. The frame data include the following parameters:

1. Motor temperature,int8_t,1℃/LSB.
2. Motor torque current value iq,int16_t, range is -2048~2048,corresponding MF motor actual torque current range is -16.5A~16.5A, corresponding MG motor actual torque current range is-33A~33A.
3. Motor speed,int16_t,1dps/LSB.
4. Encoder position,uint16_t,14bit encoder range is 0~16383;15bit encoder range is 0~32767;18bit encoder range is 0~65535(keeping high 16bit, Omit the lower 2 bit).

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

6. Speed closed loop control command

Host send commands to control motor speed. SpeedControl value is int32_t, corresponding actual speed is 0.01dps/LSB.

Data field	Instruction	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)

Remark:

1. In this command, speed control is limited by Max speed value in LK motor tool.
2. In this control mode, max acceleration is limited by Max Acceleration in LK motor tool.
3. In this control mode,max torque current of MF/MH/MG motor is limited by Max Torque Current in LK motor tool. Max power of MS motor is limited by Max Power in LK motor tool.

Driver respond(1 frame)

Motor respond after receiving the host command.MS motor respond data is same as open loop control command(only command byte is different, it's 0xA2). MF/MH/MG motor respond data is same as toque closed loop control command(only command byte is different, it's 0xA2).

7. Multi loop angle control command 1(1 frame)

Host send this command to control the position of the motor(Multi turn angle). anglecontrol is int32_t, corresponding actual position is 0.01degree/LSB, i.e 36000 corresponding to 360°,motor spin direction is determined by the difference between the target position and the current position.

Data field	Instruction	Data
DATA[0]	Command byte	0xA3
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00

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

Remark:

1. In this command, angle control is limited by Max Angle value in LK motor tool.
2. In this command, max speed is limited by Max Speed in LK motor tool.
3. In this control mode, max acceleration is limited by Max Acceleration in LK motor tool.
4. In this control mode,max torque current of MF/MH/MG motor is limited by Max Torque Current in LK motor tool. Max power of MS motor is limited by Max Power in LK motor tool.

Driver respond(1 frame)

Motor respond after receiving the host command.MS motor respond data is same as open loop control command(only command byte is different, it's 0xA3). MF/MH/MG motor respond data is same as toque closed loop control command(only command byte is different, it's 0xA3).

8. Multi loop angle control command 2(1 frame)

Host send this command to control the position of the motor(Multi turn angle).

1. angleControl is int32_t,corresponding actual position is 0.01degree/LSB, i.e 36000 corresponding to 360°, motor spin direction is determined by the difference between the target position and the current position.
2. maxSpeed limit the max speed, it is uint16_t, corresponding actual speed is 1dps/LSB, i.e 360 corresponding to 360dps.

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

Remark:

1. In this command, angleControl value is limited by Max Angle of LK motor tool.
2. In this control mode, max acceleration of motor is limited by Max acceleration of LK motor tool.
3. In this control mode, max torque current of MF/MH/MG motor is limited by Max Torque Current of LK motor tool.Max power of MS motor is limited by Max Power of LK motor tool.

Driver respond(1 frame)

Motor respond after receiving the host command.MS motor respond data is same as open loop control command(only command byte is different, it's 0xA4). MF/MH/MG motor respond data is same as toque closed loop control command(only command byte is different, it's 0xA4).

9. Single loop angle control 1(1 frame)

Host send this command to control the position of the motor(Single turn angle)

1. spinDirection for motor spin direction setting, is uint8_t, 0x00 means clockwise,0x01 means counterclockwise.
2. angleControl is uint32_t, corresponding actual position is 0.01degree/LSB, i.e 36000 means 360°.

Data field	Instruction	Data
DATA[0]	Command byte	0xA5
DATA[1]	Spin direction byte	DATA[1] = spinDirection
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Angle control 1 (bit0 : bit7)	DATA[4] = *(uint8_t *)(&angleControl)
DATA[5]	Angle control 2 (bit8 : bit15)	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	Angle control 3 (bit16 : bit23)	DATA[6] = *((uint8_t *)(&angleControl)+2)
DATA[7]	Angle control 4 (bit24: bit31)	DATA[7] = *((uint8_t *)(&angleControl)+3)

Remark:

1. In this command, max speed of motor is limited by the Max speed of LK motor tool.
2. In this control mode, max acceleration of motor is limited by Max Acceleration of LK motor tool.
3. In this control mode, max torque current of MF/MH/MG motor is limited by Max Torque Current of LK motor tool.Max power of MS motor is limited by Max Power of LK motor tool.

Driver respond(1 frame)

Motor respond after receiving the host command.MS motor respond data is same as open loop control command(only command byte is different, it's 0xA5). MF/MH/MG motor respond data is same as toque closed loop control command(only command byte is different, it's 0xA5).

10. Single loop angle control 2(1 frame)

Host send this command to control the position of the motor(single turn angle).

1. .spinDirection for motor spin direction setting, is uint8_t, 0x00 means clockwise,0x01 means counterclockwise
2. angleControl is uint32_t, corresponding actual position is 0.01degree/LSB, i.e 36000 means 360°.
3. maxSpeed is uint16_t,corresponding actual speed is 1dps/LSB,i.e 360 means 360dps.

Data field	Instruction	Data
DATA[0]	Command byte	0xA6
DATA[1]	Spin direction byte	DATA[1] = spinDirection
DATA[2]	Speed limit byte1 (bit0: bit7)	DATA[2] = *(uint8_t *)(&maxSpeed)
DATA[3]	Speed limit byte2 (bit8: bit15)	DATA[3] = *((uint8_t *)(&maxSpeed)+1)
DATA[4]	Angle control byte1 (bit0: bit7)	DATA[4] = *(uint8_t *)(&angleControl)

DATA[5]	Angle control byte2 (bit8: bit15)	DATA[5] = *((uint8_t *)(&angleControl)+1)
DATA[6]	Angle control byte3 (bit16: bit23)	DATA[6] = *((uint8_t *)(&angleControl)+2)
DATA[7]	Angle control byte4 (bit24: bit31)	DATA[7] = *((uint8_t *)(&angleControl)+3)

Remark:

1. In this control mode, max acceleration of motor is limited by Max Acceleration of LK motor tool.
2. In this control mode, max torque current of MF/MH/MG motor is limited by Max Torque Current of LK motor tool.Max power of MS motor is limited by Max Power of LK motor tool.

Driver respond(1 frame)

Motor respond after receiving the host command.MS motor respond data is same as open loop control command(only command byte is different, it's 0xA6). MF/MH/MG motor respond data is same as toque closed loop control command(only command byte is different, it's 0xA6).

11. Increment angle control command1(1 frame)

Host send commands to control the increment angle of the motor.

angleIncrement is int32_t, corresponding actual position is 0.01degree/LSB, i.e 36000 corresponding to 360°,motor spin direction is determined by the symbol of parameter.

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

Remark:

1. In this command, max speed of motor is limited by the Max speed of LK motor tool.
2. In this control mode, max acceleration of motor is limited by Max Acceleration of LK motor tool.
3. In this control mode, max torque current of MF/MH/MG motor is limited by Max Torque Current of LK motor tool.Max power of MS motor is limited by Max Power of LK motor tool.

Driver respond(1 frame)

Motor respond after receiving the host command.MS motor respond data is same as open loop control command(only command byte is different, it's 0xA7). MF/MH/MG motor respond data is same as toque closed loop control command(only command byte is different, it's 0xA7).

12. Increment angle control command 2(1 frame)

Host send commands to control the increment angle of the motor.

1. angleIncrement is int32_t, corresponding actual position is 0.01degree/LSB, i.e 36000 corresponding to 360°,motor spin direction is determined by the symbol of parameter.

2. maxSpeed is uint32_t ,corresponding actual speed is 1dps/LSB, i.e360 means 360dps.

Data field	Instruction	Data
DATA[0]	Command byte	0xA8
DATA[1]	NULL	0x00
DATA[2]	Speed limit low byte	DATA[2] = *(uint8_t *)(&maxSpeed)
DATA[3]	Speed limit high byte	DATA[3] = *((uint8_t *)(&maxSpeed)+1)
DATA[4]	Angle control low byte	DATA[4] = *(uint8_t *)(& angleIncrement)
DATA[5]	Angle control	DATA[5] = *((uint8_t *)(& angleIncrement)+1)
DATA[6]	Angle control	DATA[6] = *((uint8_t *)(& angleIncrement)+2)
DATA[7]	Angle control high byte	DATA[7] = *((uint8_t *)(& angleIncrement)+3)

Remark:

1. In this control mode, max acceleration of motor is limited by Max Acceleration of LK motor tool.
2. In this control mode, max torque current of MF/MH/MG motor is limited by Max Torque Current of LK motor tool.Max power of MS motor is limited by Max Power of LK motor tool.

Driver respond(1 frame)

Motor respond after receiving the host command.MS motor respond data is same as open loop control command(only command byte is different, it's 0xA8). MF/MH/MG motor respond data is same as toque closed loop control command(only command byte is different, it's 0xA8).

13. Read PID parameter command(1 frame)

Host send this command to read current motor PID parameters.

Data field	Instruction	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 respond(1 frame)

Driver respond data includes PI parameters of position/speed/toque loop.

Data field	Instruction	Data
DATA[0]	Command byte	0x30
DATA[1]	NULL	0x00
DATA[2]	Position loop P parameter	DATA[2] = anglePidKp
DATA[3]	Position loop I parameter	DATA[3] = anglePidKi
DATA[4]	Speed loop P parameter	DATA[4] = speedPidKp
DATA[5]	Speed loop I parameter	DATA[5] = speedPidKi
DATA[6]	Toque loop P parameter	DATA[6] = iqPidKp
DATA[7]	Toque loop I parameter	DATA[7] = iqPidKi

14. Write PID parameters to RAM command (1 frame)

Host send this command to write PID parameters to RAM, parameter are invalid when power off.

DATA[0]	Command byte	0x31
DATA[1]	NULL	0x00
Data field	**Instruction**	**Data**
DATA[2]	Position loop P parameter	DATA[2] = anglePidKp
DATA[3]	Position loop I parameter	DATA[3] = anglePidKi
DATA[4]	Speed loop P parameter	DATA[4] = speedPidKp
DATA[5]	Speed loop I parameter	DATA[5] = speedPidKi
DATA[6]	Toque loop P parameter	DATA[6] = iqPidKp
DATA[7]	Toque loop I parameter	DATA[7] = iqPidKi

Driver respond(1 frame)

Driver respond after receiving the command. Responding command is same as receiving.

15. Write PID parameters to ROM command(1 frame)

Host sent this command to write PID paramters to ROM, parameters are valid when power off.

Data field	Instruction	Data
DATA[0]	Command byte	0x32
DATA[1]	NULL	0x00
DATA[2]	Position loop P parameter	DATA[2] = anglePidKp
DATA[3]	Position loop I parameter	DATA[3] = anglePidKi
DATA[4]	Speed loop P parameter	DATA[4] = speedPidKp
DATA[5]	Speed loop I parameter	DATA[5] = speedPidKi
DATA[6]	Toque loop P parameter	DATA[6] = iqPidKp
DATA[7]	Toque loop I parameter	DATA[7] = iqPidKi

Driver respond(1 frame)

Driver respond after receiving the command. Responding command is same as receiving.

16. Read acceleration command(1 frame)

Host sent this command to read motor acceleration parameter.

Data field	Instruction	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 respond(1 frame)

Driver respond data includes acceleration data. DataAccel is int32_t ,unit is 1dps/s

Data field	Instruction	Data
DATA[0]	Command byte	0x33
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Acceleration low byte1	DATA[4] = *(uint8_t *)(&Accel)
DATA[5]	Acceleration byte 2	DATA[5] = *((uint8_t *)(&Accel)+1)
DATA[6]	Acceleration byte 3	DATA[6] = *((uint8_t *)(&Accel)+2)
DATA[7]	Acceleration byte 4	DATA[7] = *((uint8_t *)(&Accel)+3)

17. Write acceleration to RAM command(1 frame)

Host send commands to write acceleration to RAM.Parameter are invalid when power off. DataAccel is int32_t,unit is 1dps/s

Data field	Instruction	Data
DATA[0]	Command byte	0x34
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Acceleration low byte1	DATA[4] = *(uint8_t *)(&Accel)
DATA[5]	Acceleration byte 2	DATA[5] = *((uint8_t *)(&Accel)+1)
DATA[6]	Acceleration byte 3	DATA[6] = *((uint8_t *)(&Accel)+2)
DATA[7]	Acceleration byte 4	DATA[7] = *((uint8_t *)(&Accel)+3)

Driver respond(1 frame)

Driver respond after receiving the command. Responding command is same as receiving.

18. Read encoder command (1 frame)

Host send commands to read current encoder position.

Data field	Instruction	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

Driver respond(1 frame)

Motor reply to the host after receiving the commands. The frame data include the following parameters:

1. encoder(uint16_t, 14 bit encoder range is 0~16383), is encoder raw value minus encoder offset value.

2. encoderRaw(uint16_t,14 bit encoder range is 0~16383).
3. encoderOffset(uint16_t,14 bit encoder range is 0~16383). This point is the initial zero position of the motor.

Data field	Instruction	Data
DATA[0]	Command byte	0x90
DATA[1]	NULL	0x00
DATA[2]	Encoder low byte	DATA[2] = *(uint8_t *)(&encoder)
DATA[3]	Encoder high byte	DATA[3] = *((uint8_t *)(&encoder)+1)
DATA[4]	Encoder raw position low byte	DATA[4] = *(uint8_t *)(&encoderRaw)
DATA[5]	Encoder raw 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)

19. Write encoder value to ROM as the motor zero point command(1 frame)

Host send this command to set encoder offset. EncoderOffset is uint16_t,14bit encoder range is 0~16383.

Data field	Instruction	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)

Driver respond(1 frame)

Driver respond after receiving the command. Responding command is same as receiving.

20. Write current position to ROM as the motor zero point command(1 frame)

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

Remark:

1. The command will be valid only after reset power.
2. This command will write the zero point to the driver's ROM, multiple writes will affect the chip life, and frequent use is not recommended.

Data field	Instruction	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 respond(1 frame)

Driver respond after receive commands from host. EncoderOffset is 0 bias.

Data field	Instruction	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]	EncoderOffset low byte	DATA[6] = *(uint8_t *)(&encoderOffset)
DATA[7]	EncoderOffset high byte	DATA[7] = *((uint8_t *)(&encoderOffset)+1)

21. Read multi angle loop command(1 frame)

Host send this command to read current motor multi angle absolute value.

Data field	Instruction	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

Driver respond(1 frame)

Motor reply to the host after receiving the commands. The frame data include the following parameters:

1. motorAngle is int64_t, positive values represent clockwise cumulative angles, and negative values represent counterclockwise cumulative angles, unit 0.01°/LSB.

Data field	Instruction	Data
DATA[0]	Command byte	0x92
DATA[1]	Angle low byte1	DATA[1] = *(uint8_t *)(&motorAngle)
DATA[2]	Angle byte2	DATA[2] = *((uint8_t *)(& motorAngle)+1)
DATA[3]	Angle byte3	DATA[3] = *((uint8_t *)(& motorAngle)+2)
DATA[4]	Angle byte4	DATA[4] = *((uint8_t *)(& motorAngle)+3)
DATA[5]	Angle byte5	DATA[5] = *((uint8_t *)(& motorAngle)+4)
DATA[6]	Angle byte6	DATA[6] = *((uint8_t *)(& motorAngle)+5)
DATA[7]	Angle byte7	DATA[7] = *((uint8_t *)(& motorAngle)+6)

22. Read single angle loop command(1 frame)

Host send this command to read motor current single angle value.

Data field	Instruction	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

Driver respond(1 frame)

Motor reply to the host after receiving the commands. The frame data include the following parameters:

1. circleAngle is motor single loop, is uint32_t,Starting at the zero point of the encoder, it increases clockwise, and the value returns to 0 when the zero point is reached again,unit is 0.01°/LSB,value range is 0~36000-1.

Data field	Instruction	Data
DATA[0]	Command byte	0x94
DATA[1]	NULL	0x00
DATA[2]	NULL	0x00
DATA[3]	NULL	0x00
DATA[4]	Single angle low byte1	DATA[4] = *(uint8_t *)(& circleAngle)
DATA[5]	Single angle byte2	DATA[5] = *((uint8_t *)(& circleAngle)+1)
DATA[6]	Single angle byte3	DATA[6] = *((uint8_t *)(& circleAngle)+2)
DATA[7]	Single angle high byte4	DATA[7] = *((uint8_t *)(& circleAngle)+3)

23. Clear motor angle loop command(1 frame)Not yet unavailable

This command clear motor multi turn and single turn data and set current position as motor zero point. It's invalid when power off.

Remark: This command will clear all the position loop command at the same time.

Data field	Instruction	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

Driver respond(1 frame)

Driver respond after receiving the host command.Frame data is same as the host sending command.

24. Read motor state 1 and error state command(1 frame)

This command read current motor temperature, voltage and error state.

Data field	Instruction	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

Driver respond(1 frame)

Motor reply to the host after receiving the commands. The frame data include the following parameters:

1. Motor temperature(int8_t, 1℃/LSB).
2. Motor voltage(uint16_t, 0.1V/LSB).
3. errorState(uint8_t, the bits represent different motor states)

Data field	Instruction	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

Remark:

1. errorState each bit specific state sheet:

errorState bit	State Instruction	0	1
0	Voltage state	Voltage is normal	Under Voltage protect
1	invalid		
2	invalid		
3	Temperature state	Temperature is normal	Over temperature protect
4	invalid		
5	invalid		
6	invalid		
7	invalid		

25. Clear motor error state(1 frame)

This command is to clear motor current error state.

Data field	Instruction	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

Driver respond(1 frame)

Motor reply to the host after receiving the commands. The frame data include the following parameters:

1. Motor temperature(int8_t, 1℃/LSB).
2. Motor voltage(uint16_t,0.1V/LSB).
3. errorState(uint8_t,the bits represent different motor states).

Data field	Instruction	Data
DATA[0]	Command byte	0x9A
DATA[1]	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

Remark:

1. Error can't be cleared unless the motor state is back to normal.
2. Take errorState each bit specific state sheet for reference.

26. Read motor state 2 command(1 frame)

This command read current temperature,voltage,speed and encoder position.

Data field	Instruction	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

Driver respond(1 frame)

Motor reply to the host after receiving the commands. The frame data include the following parameters:

1. Temperature(int8_t,1℃/LSB).
2. Toque current iq(int16_t,range is -2048~2048, corresponding actual current is -33A~33A).
3. Motor speed(int16_t,1dps/LSB).
4. Encoder position (uint16_t,14bit encoder range is 0~16383).

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

27. Read motor state 3 command (1 frame)

This command read current motor temperature and phase current data.

Data field	Instruction	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

Driver respond(1 frame)

Motor reply to the host after receiving the commands. The frame data include the following parameters:

1. temperature(int8_t ,1℃/LSB)
2. A phase current, int16_t,corresponding actual phase current is 1A/64LSB.
3. B phase current, int16_t,corresponding actual phase current is 1A/64LSB.
4. C phase current, int16_t,corresponding actual phase current is 1A/64LSB.

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

3.Multi motor command

Multi motor command need to send in setting software. Multi motor command and single motor

command can't be sent at the same time.

1. **Multi motor torque closed loop control command (1 frame)**

Format are as follows:

Identifier:0x280

Frame format:DATA

Frame type:standard frame

DLC:8byte

Motor send this command to control torque current output of 4 motors at most. iqControl is int16_t, range is -2000~ 2000, corresponding actual torque current is -32A~32A(Bus current and motor actual torque are different based on different motor)

Motor ID should be #1~#4 and can not be repeat,corresponding to 4 torque current of frame data.

Data field	Instruction	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 respond(1 frame)

Format are as follows:

Identifier:0x140 + ID(1~4)

Frame format:DATA

Frame type:standard frame

DLC:8byte

Each motor reply in order from ID 1 to ID 4. Respond data of each motor are same as the respond data of single motor torque closed loop command.