# Servo Motor Control Protocol V3.0

CATALOG

# Disclaimer

Thanks for choosing RMD series motor.Please read this statement carefully before using.Once used, this statement is deemed to be approved and accepted.Please install and use this product strictly in accordance with the manual, product description and relevant laws, regulations, policies and guidelines. In the process of using the product, the user undertakes to be responsible for his own behavior and all consequences arising therefrom. MYACTUATOR will not be liable for any loss caused by improper use, installation and modification of the user.

# 1. Communication Bus Parameters and Message Format

## 1.1.CAN Bus

### 1.1.1. Parameters

Bus interface: CAN
Baud rate: 1Mbps

### 1.1.2. Message format

Identifier: Single motor command sending: 0x140 + ID(1~32)
Multi-motor command sending: 0x280
Reply: 0x240 + ID (1~32)
Frame format: data frame
Frame Type: Standard Frame
DLC: 8 bytes

## 1.2.RS485 bus

### 1.2.1. Parameters

Bus interface: RS485
Baudrate:115200bps，500Kbps，1Mbps

### 1.2.2. Message format

| Type | Data Defination | Bytes | Description |
|---|---|---|---|
| Frame header | 0x3E | 1 | Communication frame header, used for identification. |
| ID | 1~32 | 1 | Device address, corresponding to the ID number of each motor. |
| Data Length | Data Length | 1 | The length of the data field. In the standard protocol, the length is fixed to 8 bytes. |
| Data field | Data content | According to the length | The content of the data field in the standard protocol is exactly the same as that of the CAN. |
| Check | CRC Check | 2 | CRC16 check, low order first, high order last. |

# 2. Single motor command description

## 2.1. Read PID parameter command (0x30)

### 2.1.1. Instruction description

This command can read the parameters of current, speed, position loop KP and KI at one time, and the data type is uint8_t. The system sets the maximum range of PI parameters according to the motor model, and then divides it equally according to the maximum range of uint8_t of 256 units. Users only need to adjust 0-256 units.

### 2.1.2. Send data field definition

| 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 |

### 2.1.3. Reply data field definition

| Data field | Description | Data |
|---|---|---|
| DATA[0] | Command byte | 0x30 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | Current loop KP parameters | DATA[2] = (uint8_t)(CurrKP) |
| DATA[3] | Current loop KI parameters | DATA[3] = (uint8_t)(CurrKI) |
| DATA[4] | Speed loop KP parameters | DATA[4] = (uint8_t)(SpdKP) |
| DATA[5] | Speed loop KI parameters | DATA[5] = (uint8_t)(SpdKI) |
| DATA[6] | Position loop KP parameters | DATA[6] = (uint8_t)(PosKP) |

| | | |
|---|---|---|
| DATA[7] | Position loop KI parameters | DATA[7] = (uint8_t)(PosKI) |

### 2.1.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x30 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485:**

| Frame Header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x30 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x30 | CRC16L | CRC16H |

**Description:** Send command to read PID parameters.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x30 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 |

**RS485:**

| Frame Header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x30 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 | CRC16L | CRC16H |

**Description:**

Data[2] represents the current loop KP parameter, 0x55 decimal represents 85, assuming that the maximum value of the current loop set by the system is 3, then the actual value of 1 unit is 3/256 = 0.01171875, and 85 units represent the actual value 85*0.01171875 = 0.99609375, which is the actual value of the KP parameter of the current loop inside the system.

Data[3] represents the current loop KI parameter, 0x19 decimal represents 25, assuming that the maximum value of the current loop set by the system is 0.1, then the actual value of 1 unit is 0.1/256 = 0.00039062, and 25 units represent the actual value of 25* 0.00039062 = 0.0097656, which is the actual value of

the KI parameter of the current loop inside the system.

Data[4] represents the KP parameter of the speed loop, and 0x55 in decimal represents 85. Assuming that the maximum value of the speed loop set by the system is 0.1, the actual value of 1 unit is 0.1/256 = 0.00039062, and 85 units represent the actual value of 85* 0.00039062 = 0.0332027, this is the actual value of the KP parameter of the internal speed loop of the system.

Data[5] represents the speed loop KI parameter, 0x19 decimal represents 25, assuming that the maximum speed loop set by the system is 0.01, then the actual value of 1 unit is 0.01/256 = 0.00003906, and 25 units means the actual value is 25* 0.00003906 = 0.0009765, this is the actual value of the KI parameter of the speed loop inside the system.

Data[6] represents the KP parameter of the position loop, 0x55 in decimal means 85, assuming that the maximum value of the position loop set by the system is 0.1, then the actual value of 1 unit is 0.1/256 = 0.00039062, and 85 units means the actual value is 85* 0.00039062 = 0.0332027, this is the actual value of the KP parameter of the internal position loop of the system.

Data[7] represents the position loop KI parameter, 0x19 decimal represents 25, assuming that the maximum value of the position loop set by the system is 0.01, then the actual value of 1 unit is 0.01/256 = 0.00003906, and 25 units means the actual value is 25* 0.00003906 = 0.0009765, which is the actual value of the KI parameter of the internal position loop of the system.

## 2.2. Write PID parameters to RAM command (0x31)

### 2.2.1. Instruction description

This command can write the parameters of current, speed, position loop KP and KI to RAM at one time, and it will not be saved after power off. The data type is uint8_t. The system sets the maximum range of PI parameters according to the motor model, and then divides it equally according to the maximum range of uint8_t of 256 units. Users only need to adjust 0-256 units.

### 2.2.2. Send data field definition

| Data field | Description | Data |
|---|---|---|
| DATA[0] | Command byte | 0x31 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | Current loop KP parameter | DATA[2] = (uint8_t)(CurrKP) |
| DATA[3] | Current loop KI parameter | DATA[3] = (uint8_t)(CurrKI) |

| | | |
|---|---|---|
| DATA[4] | Speed loop KP parameter | DATA[4] = (uint8_t)(SpdKP) |
| DATA[5] | Speed loop KI parameter | DATA[5] = (uint8_t)(SpdKI) |
| DATA[6] | Position loop KP parameter | DATA[6] = (uint8_t)(PosKP) |
| DATA[7] | Position loop KI parameter | DATA[7] = (uint8_t)(PosKI) |

### 2.2.3. Reply data field definition

The content of the reply data is the same as the sent data.

### 2.2.4. Communication example

**Example 1:**
**Send command:**
**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x31 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 |

**RS485:**

| Frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x31 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 | CRC16L | CRC16H |

**Description:**
Data[2] represents the current loop KP parameter, 0x55 decimal represents 85, assuming that the maximum current loop set by the system is 3, then the actual value of 1 unit is 3/256 = 0.01171875, and 85 units represent the actual value of 85* 0.01171875 = 0.99609375, which is the actual value of the KP parameter of the current loop inside the system.

Data[3] represents the current loop KI parameter, 0x19 decimal represents 25, assuming that the maximum value of the current loop set by the system is 0.1, then the actual value of 1 unit is 0.1/256 = 0.00039062, and 25 units represent the actual value of 25* 0.00039062 = 0.0097656, which is the actual value of the KI parameter of the current loop inside the system.

Data[4] represents the KP parameter of the speed loop, and 0x55 in decimal represents 85. Assuming that the maximum value of the speed loop set by the system is 0.1, the actual value of 1 unit is 0.1/256 = 0.00039062, and 85 units represent the actual value of 85* 0.00039062 = 0.0332027, this is the actual value of the KP parameter of the internal speed loop of the system.

Data[5] represents the speed loop KI parameter, 0x19 decimal represents 25, assuming that the maximum speed loop set by the system is 0.01, then the actual value of 1 unit is 0.01/256 = 0.00003906, and 25 units means the actual value is 25* 0.00003906 = 0.0009765, this is the actual value of the KI parameter of the speed loop inside the system.

Data[6] represents the KP parameter of the position loop, 0x55 in decimal means 85, assuming that the maximum value of the position loop set by the system is 0.1, then the actual value of 1 unit is 0.1/256 = 0.00039062, and 85 units means the actual value is 85* 0.00039062 = 0.0332027, this is the actual value of the KP parameter of the internal position loop of the system.

Data[7] represents the position loop KI parameter, 0x19 decimal represents 25, assuming that the maximum value of the position loop set by the system is 0.01, then the actual value of 1 unit is 0.01/256 = 0.00003906, and 25 units means the actual value is 25* 0.00003906 = 0.0009765, which is the actual value of the KI parameter of the internal position loop of the system.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x31 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 |

**RS485:**

| Frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x31 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 | CRC16L | CRC16H |

## 2.3. Write PID parameters to ROM command (0x32)

### 2.3.1. Instruction description

This command can write the parameters of current, speed, position loop KP and KI to ROM at one time, which can be saved after power off. The data type is uint8_t. The system sets the maximum range of PI parameters according to the motor model, and then divides it equally according to the maximum range of uint8_t of 256 units. Users only need to adjust 0-256 units.

### 2.3.2. Send data field definition

| Data Field | Description | Data |
|---|---|---|

| | | |
|---|---|---|
| DATA[0] | command byte | 0x32 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | Current loop KP parameters | DATA[2] = (uint8_t)(CurrKP) |
| DATA[3] | Current loop KI parameters | DATA[3] = (uint8_t)(CurrKI) |
| DATA[4] | Speed loop KP parameters | DATA[4] = (uint8_t)(SpdKP) |
| DATA[5] | Speed loop KI parameters | DATA[5] = (uint8_t)(SpdKI) |
| DATA[6] | Position loop KP parameters | DATA[6] = (uint8_t)(PosKP) |
| DATA[7] | Position loop KI parameters | DATA[7] = (uint8_t)(PosKI) |

### 2.3.3. Reply data field definition

The content of the reply data is the same as the sent data.

### 2.3.4. Communication example

**Example 1:**
**Send command:**
**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x32 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 |

**RS485：**

| frame header | ID | length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x32 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 | CRC16L | CRC16H |

**Description:**
Data[2] represents the current loop KP parameter, 0x55 decimal represents 85, assuming that the maximum value of the current loop set by the system is 3, then the actual value of 1 unit is 3/256 = 0.01171875, and 85 units represent the actual value 85*0.01171875 = 0.99609375, which is the actual value of the KP parameter of the current loop inside the system.

Data[3] represents the current loop KI parameter, 0x19 decimal represents 25, assuming that the maximum value of the current loop set by the system is 0.1, then the actual value of 1 unit is 0.1/256 = 0.00039062, and 25 units represent the actual value of 25* 0.00039062 = 0.0097656, which is the actual value of the KI parameter of the current loop inside the system.

Data[4] represents the KP parameter of the speed loop, and 0x55 in decimal represents 85. Assuming that the maximum value of the speed loop set by the system is 0.1, the actual value of 1 unit is 0.1/256 = 0.00039062, and 85 units represent the actual value of 85* 0.00039062 = 0.0332027, this is the actual value of the KP parameter of the internal speed loop of the system.

Data[5] represents the speed loop KI parameter, 0x19 decimal represents 25, assuming that the maximum speed loop set by the system is 0.01, then the actual value of 1 unit is 0.01/256 = 0.00003906, and 25 units means the actual value is 25* 0.00003906 = 0.0009765, this is the actual value of the KI parameter of the speed loop inside the system.

Data[6] represents the KP parameter of the position loop, 0x55 in decimal means 85, assuming that the maximum value of the position loop set by the system is 0.1, then the actual value of 1 unit is 0.1/256 = 0.00039062, and 85 units means the actual value is 85* 0.00039062 = 0.0332027, this is the actual value of the KP parameter of the internal position loop of the system.

Data[7] represents the position loop KI parameter, 0x19 decimal represents 25, assuming that the maximum value of the position loop set by the system is 0.01, then the actual value of 1 unit is 0.01/256 = 0.00003906, and 25 units means the actual value is 25* 0.00003906 = 0.0009765, which is the actual value of the KI parameter of the internal position loop of the system.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x32 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 |

**RS485：**

| frame header | ID | length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x32 | 0x00 | 0x55 | 0x19 | 0x55 | 0x19 | 0x55 | 0x19 | CRC16L | CRC16H |

## 2.4. Read acceleration command (0x42)

### 2.4.1. Instruction description

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

### 2.4.2. Send data field definition

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

### 2.4.3. Reply data field definition

The acceleration parameter is included in the drive response data. Acceleration data Accel is int32_t type, the unit is 1dps/s, and the parameter range is 50-60000.

| data field | Description | Data |
|---|---|---|
| DATA[0] | command byte | 0x42 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | Acceleration low byte 1 | DATA[4] = (uint8_t)(Accel) |
| DATA[5] | acceleration byte 2 | DATA[5] = (uint8_t)(Accel>>8) |
| DATA[6] | acceleration byte 3 | DATA[6] = (uint8_t)(Accel>>16) |
| DATA[7] | acceleration byte 4 | DATA[7] = (uint8_t)(Accel>>24) |

### 2.4.4. Communication example

**Example 1:**
**Send command:**
**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x42 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x42 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**
Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00002710, which means 10000 in decimal. It means that the acceleration of the motor position loop is 10000dps/s.

## 2.5. Write acceleration to RAM command (0x43)

### 2.5.1. Instruction description

The host sends this command to write the acceleration into the RAM, and the write parameters become invalid after the power is turned off. Acceleration data Accel is uint32_t type, the unit is 1dps/s, and the parameter range is 50-80000.

### 2.5.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x43 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | Acceleration low byte 1 | DATA[4] = (uint8_t)(Accel) |
| DATA[5] | acceleration byte 2 | DATA[5] = (uint8_t)(Accel>>8) |
| DATA[6] | acceleration byte 3 | DATA[6] = (uint8_t)(Accel>>16) |
| DATA[7] | acceleration byte 4 | DATA[7] = (uint8_t)(Accel>>24) |

### 2.5.3. Reply data field definition

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

### 2.5.4. Communication example

**Example 1:**
**Send command:**
**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x43 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x43 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00002710, which means 10000 in decimal. Indicates that the acceleration value of 10000dps/s is written to the motor driver, and the acceleration value is not saved after the power is turned off.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x43 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x43 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

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

## 2.6. Read multi-turn encoder position data command (0x60)

### 2.6.1. Instruction description

The host sends this command to read the multi-turn position of the encoder, which represents the rotation angle of the motor output shaft, including the multi-turn angle.

### 2.6.2. Send data field definition

| data field | Description | data |
|---|---|---|

| | | |
|---|---|---|
| DATA[0] | command byte | 0x60 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |
| DATA[7] | NULL | 0x00 |

### 2.6.3. Reply data field definition

The motor replies to the host after receiving the command, and the frame data contains the following parameters. Encoder multi-turn position encoder (int32_t type, value range of multi-turn encoder, 4 bytes of valid data), which is the value after subtracting the encoder's multi-turn zero offset (initial position) from the original position of the encoder.

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x60 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | Encoder position low byte 1 | DATA[4] = (uint8_t)(encoder) |
| DATA[5] | encoder position byte 2 | DATA[5] = (uint8_t)(encoder>>8) |
| DATA[6] | encoder position byte 3 | DATA[6] = (uint8_t)(encoder>>16) |
| DATA[7] | encoder position byte 4 | DATA[7] = (uint8_t)(encoder>>24) |

### 2.6.4. Communication example

**Example 1:**
**Send command:**
**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x60 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x60 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:** The host sends this command to read the multi-turn position of the encoder.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x60 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x60 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:** Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00002710, which means 10000 in decimal. The multi-turn encoder value representing the current relative multi-turn zero offset (initial position) of the motor is 10000 pulses.

## 2.7. Read multi-turn encoder original position data command (0x61)

### 2.7.1. Instruction description

The host sends this command to read the multi-turn encoder home position, ie the multi-turn encoder value without the zero offset (home position).

### 2.7.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x61 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |

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

### 2.7.3. Reply data field definition

The motor replies to the host after receiving the command, and the frame data contains the following parameters. Encoder multi-turn raw position encoderRaw (int32_t type, value range, valid data 4 bytes).

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x61 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | Encoder original position byte1 | DATA[4] = (uint8_t)(encoderRaw) |
| DATA[5] | Encoder original position byte 2 | DATA[5] = (uint8_t)(encoderRaw>>8) |
| DATA[6] | Encoder original position byte3 | DATA[6] = (uint8_t)(encoderRaw>>16) |
| DATA[7] | Encoder original position byte4 | DATA[7] = (uint8_t)(encoderRaw>>24) |

### 2.7.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x61 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x61 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

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

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x61 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x61 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00002710, which means 10000 in decimal. Indicates that the current multi-turn encoder value of the motor is 10000 pulses, excluding the zero offset (initial position).

## 2.8. Read multi-turn encoder zero offset data command (0x62)

### 2.8.1. Instruction description

The host sends this command to read the multi-turn zero offset value (initial position) of the encoder.

### 2.8.2. Send data field definition

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

### 2.8.3. Reply data field definition

The motor replies to the host after receiving the command, and the frame data contains the following parameters. Encoder multi-turn zero offset encoderOffset (int32_t type, value range, valid data 4 bytes).

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x62 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | Encoder Offset Byte 1 | DATA[4] = (uint8_t)(encoderOffset) |
| DATA[5] | Encoder Offset Byte2 | DATA[5] = (uint8_t)(encoderOffset>>8) |
| DATA[6] | Encoder Offset Byte3 | DATA[6] = (uint8_t)(encoderOffset>>16) |
| DATA[7] | Encoder Offset Byte4 | DATA[7] = (uint8_t)(encoderOffset>>24) |

### 2.8.4. Communication example

**Example 1:**
**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x62 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x62 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

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

**Reply command:**
**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x62 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**RS485：**

| frame | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

| header | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x62 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00002710, which means 10000 in decimal. Indicates that the current multi-turn encoder zero offset value of the motor is 10000 pulses.

## 2.9. Write encoder multi-turn value to ROM as motor zero command (0x63)

### 2.9.1. Instruction description

The host sends this command to set the zero offset (initial position) of the encoder, where the encoder multi-turn value to be written, encoderOffset, is of type int32_t, (value range, 4 bytes of valid data).

Note: After writing the position of the new zero point, the motor needs to be restarted to be effective. Because of the change of the zero offset, the new zero offset (initial position) should be used as a reference when setting the target position.

### 2.9.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x63 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | Encoder zero bias low byte 1 | DATA[4] = (uint8_t)(encoderOffset) |
| DATA[5] | Encoder Offset Byte2 | DATA[5] = (uint8_t)(encoderOffset>>8) |
| DATA[6] | Encoder Offset Byte3 | DATA[6] = (uint8_t)(encoderOffset>>8) |
| DATA[7] | Encoder Offset Byte4 | DATA[7] = (uint8_t)(encoderOffset>>8) |

### 2.9.3. Reply data field definition

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

### 2.9.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x63 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x63 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00002710, which means 10000 in decimal. It means to write 10000 pulses as multi-turn encoder zero offset.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x63 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x63 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

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

## 2.10. Write the current multi-turn position of the encoder to the ROM as the motor zero command (0x64)

### 2.10.1. Instruction description

Write the current encoder position of the motor as the multi-turn encoder zero offset (initial position) into the ROM

Note: After writing the new zero point position, the motor needs to be restarted to be effective. Because of the change of the zero offset, the new zero offset (initial position) should be used as a reference when setting the target position.

### 2.10.2. Send data field definition

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

### 2.10.3. Reply data field definition

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

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x64 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | Encoder zero bias low byte1 | DATA[4] = (uint8_t)(encoderOffset) |
| DATA[5] | Encoder Offset Byte2 | DATA[5] = (uint8_t)(encoderOffset>>8) |
| DATA[6] | Encoder Offset Byte3 | DATA[6] = (uint8_t)(encoderOffset>>16) |
| DATA[7] | Encoder Offset Byte4 | DATA[7] = (uint8_t)(encoderOffset>>24) |

### 2.10.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x64 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

| header | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x64 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

After sending the 0x64 command, the motor will write the current multi-turn encoder value as the zero offset (initial position) into the ROM.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x64 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x64 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00002710, which means 10000 in decimal. Indicates that the multi-turn zero offset value (initial position) written to the motor is 10,000 pulses.

## 2.11. Read multi-turn angle command (0x92)

### 2.11.1. Instruction description

The host sends this command to read the current multi-turn absolute angle value of the motor.

### 2.11.2. Send data field definition

| 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 |

### 2.11.3. Reply data field definition

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

1. Motor angle motorAngle, (int32_t type, value range, valid data 4 bytes), unit 0.01°/LSB.

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x92 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | angle low byte 1 | DATA[4] = (uint8_t)(motorAngle) |
| DATA[5] | angle bytes2 | DATA[5] = (uint8_t)(motorAngle>>8) |
| DATA[6] | angle bytes3 | DATA[6] = (uint8_t)(motorAngle>>16) |
| DATA[7] | angle bytes4 | DATA[7] = (uint8_t)(motorAngle>>24) |

### 2.11.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x92 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x92 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

After sending the 0x92 command, it will return the absolute angle of the motor output shaft.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x92 | 0x00 | 0x00 | 0x00 | 0xA0 | 0x8C | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x92 | 0x00 | 0x00 | 0x00 | 0xA0 | 0x8C | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) The 32-bit data is 0x00008CA0, which means the decimal is 36000, which is reduced by 100 times in units of 0.01°/LSB That is 36000*0.01=360°. Indicates that the motor output shaft moves 360° in the positive direction relative to the zero position.

## 2.12. Read Motor Status 1 and Error Flag Command (0x9A)

### 2.12.1. Instruction description

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

### 2.12.2. Send data field definition

| 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 |

### 2.12.3. Reply data field definition

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

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

2. Brake control command: Indicates the state of the brake control command, 1 represents the brake release command, and 0 represents the brake lock command.
2. Voltage (uint16_t type, unit 0.1V/LSB).
3. Error flag errorState (of type uint16_t, 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] | Brake release command | DATA[3] = (uint8_t)(RlyCtrlRslt) |
| DATA[4] | voltage low byte | DATA[4] = (uint8_t)(voltage) |
| DATA[5] | voltage high byte | DATA[5] = (uint8_t)(voltage>>8) |
| DATA[6] | Error Status Low Byte 1 | DATA[6] = (uint8_t)(errorState) |
| DATA[7] | error status byte 2 | DATA[7] = (uint8_t)(errorState>>8) |

**Remark:**
1. System abnormal state value System_errorState state table 1 is as follows:

| System_errorState | Status Description |
|---|---|
| 0x0002 | Motor stall |
| 0x0004 | low pressure |
| 0x0008 | overvoltage |
| 0x0010 | overcurrent |
| 0x0040 | Power overrun |
| 0x0100 | speeding |
| 0x0200 | |
| 0x0400 | |
| 0x0800 | |
| 0x1000 | Motor temperature over temperature |
| 0x2000 | Encoder calibration error |

### 2.12.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x9A | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x9A | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

After sending the 0x9A command, the temperature, voltage and error status flags of the motor will be returned.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x9A | 0x32 | 0x00 | 0x01 | 0xE5 | 0x01 | 0x04 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x9A | 0x32 | 0x00 | 0x01 | 0xE5 | 0x01 | 0x04 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment.

Data[3] indicates that the brake indicates the state of the brake control command, 1 represents the brake release command, and 0 represents the brake lock command. So 0x01 indicates that the current brake release command has been executed.

Data[4] and Data[5] (Data[4] is the low bit, Data[5] is the high bit) form 0x01E5, the decimal is 485, which is reduced by 10 times according to the unit of 0.1V/LSB, 485*0.1=48.5V, representing The current motor supply voltage is 48.5V.

Data[6] and Data[7] (Data[6] is low and Data[7] is high) form 0x0004, which indicates a low-voltage error according to the error description in the System_errorState table.

## 2.13. Read Motor Status 2 Command (0x9C)

### 2.13.1. Instruction description

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

### 2.13.2. Send data field definition

| 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 |

### 2.13.3. Reply data field definition

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

1. Motor temperature temperature (int8_t type, 1℃/LSB).
2. The torque current value iq of the motor (int16_t type, 0.01A/LSB).
3. Motor output shaft speed (int16_t type, 1dps/LSB).
4. Motor output shaft angle (int16_t type, 1degree/LSB, maximum range ±32767degree).

| 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>>8) |
| DATA[4] | Motor speed low byte | DATA[4] = (uint8_t)(speed) |

| | | |
|---|---|---|
| DATA[5] | Motor speed high byte | DATA[5] = (uint8_t)(speed>>8) |
| DATA[6] | Motor angle low byte | DATA[6] = (uint8_t)(degree) |
| DATA[7] | Motor angle high byte | DATA[7] = (uint8_t)(degree>>8) |

### 2.13.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x9C | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x9C | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

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

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x9C | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x9C | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data of Data[2] and Data[3] 0x0064 is 100 in decimal, and it is 100*0.01=1A when scaled down by 100 times, which means that the actual current of the current motor is 1A. The composite data 0x01F4 of Data[4] and Data[5] is 500 in decimal, which means the motor

output shaft speed is 500dps. There is a reduction ratio relationship between the motor output shaft speed and the motor speed. If the reduction ratio is 6, then the motor speed is 6 times higher than the output shaft speed. The composite data of Data[6] and Data[7] 0x002D is 45 in decimal, which means that the motor output shaft moves 45 degrees in the positive direction relative to the zero position. The position of the motor output shaft is related to the number of lines of the motor encoder and the reduction ratio. For example, if the number of lines of the motor encoder is 65536 and the reduction ratio is 6, then 360 degrees of the motor output shaft corresponds to 65536*6 = 393216 pulses.

## 2.14. Read Motor Status 3 Command (0x9D)

### 2.14.1. Instruction description

This command reads the current motor temperature and phase current data

### 2.14.2. Send data field definition

| 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 |

### 2.14.3. Reply data field definition

The motor replies to the host after receiving the command, and the frame data contains the following data:
1. Motor temperature temperature (int8_t type, 1℃/LSB)
2. Phase A current data, the data type is int16_t, and the corresponding actual phase current is 0.01ALSB.
3.B-phase current data, the data type is int16_t type, and the corresponding actual phase current is 0.01ALSB.
4. C-phase current data, the data type is int16_t type, and the corresponding actual phase current is 0.01ALSB.

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x9D |
| DATA[1] | Motor temperature | DATA[1] = (uint8_t)(temperature) |
| DATA[2] | Phase A current low byte | DATA[2] = (uint8_t)(iA) |
| DATA[3] | Phase A current high byte | DATA[3] = (uint8_t)(iA>>8) |
| DATA[4] | Phase B current low byte | DATA[4] = (uint8_t)(iB) |
| DATA[5] | Phase B current high byte | DATA[5] = (uint8_t)(iB>>8) |
| DATA[6] | Phase C current low byte | DATA[6] = (uint8_t)(iC) |
| DATA[7] | Phase C current high byte | DATA[7] = (uint8_t)(iC>>8) |

### 2.14.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x9D | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x9D | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

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

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x9D | 0x32 | 0xC2 | 0x0B | 0x10 | 0xFA | 0xC0 | 0xF9 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x9D | 0x32 | 0xC2 | 0x0B | 0x10 | 0xFA | 0xC0 | 0xF9 | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data 0x0BC2 of Data[2] and Data[3] is 3010 in decimal, and it is 3010*0.01=30.1A when scaled down by 100 times, which means that the actual current of the current phase A of the motor is 30.1A. The composite data 0xFA10 of Data[4] and Data[5] is -1520 in decimal, and it is -1520*0.01=-15.2A when scaled down by 100 times, which means that the actual current of the current phase B of the motor is -15.2A. The composite data 0xF9C0 of Data[6] and Data[7] is -1600 in decimal, and it is -1600*0.01=-16A when scaled down by 100 times, which means that the actual current of the current phase C of the motor is -16A.

## 2.15. Motor shutdown command (0x80)

### 2.15.1. Instruction description

Turns off the motor output and also clears the motor running state, not in any closed loop mode.

### 2.15.2. Send data field definition

| 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 |

### 2.15.3. Reply data field definition

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

### 2.15.4. Communication example

## 2.16. Motor stop command (0x81)

### 2.16.1. Instruction description

Stop the motor, the closed-loop mode where the motor is still running, just stop the motor speed.

### 2.16.2. Send data field definition

| 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 |

### 2.16.3. Reply data field definition

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

### 2.16.4. Communication example

## 2.17. Torque closed-loop control command (0xA1)

### 2.17.1. Instruction description

This command is a control command, which can be run when the motor is not faulty. The host sends this command to control the torque and current output of the motor. The control value iqControl is of type int16_t and the unit is 0.01A/LSB.

### 2.17.2. Send data field definition

| 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>>8) |
| DATA[6] | NULL | 0x00 |
| DATA[7] | NULL | 0x00 |

### 2.17.3. Reply data field definition

The motor replies to the host after receiving the command, and the frame data contains the following parameters.
5. Motor temperature temperature (int8_t type, 1℃/LSB).
6. The torque current value iq of the motor (int16_t type, 0.01A/LSB).
7. Motor output shaft speed (int16_t type, 1dps/LSB).
8. Motor output shaft angle (int16_t type, 1degree/LSB, maximum range ±32767degree).

| 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>>8) |
| DATA[4] | Motor speed low byte | DATA[4] = (uint8_t)(speed) |
| DATA[5] | Motor speed high byte | DATA[5] = (uint8_t)(speed>>8) |
| DATA[6] | Motor angle low byte | DATA[6] = (uint8_t)(degree) |
| DATA[7] | Motor angle high byte | DATA[7] = (uint8_t)(degree>>8) |

### 2.17.4. Communication example

**Example 1:**
**Send command:**
**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xA1 | 0x00 | 0x00 | 0x00 | 0x64 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA1 | 0x00 | 0x00 | 0x00 | 0x64 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] and data[5] represent the data size, Data[4] (0x64) is the low bit, and Data[5] (0x00) is the high bit. So the actual data is 0x0064, which means decimal 100, which is 100*0.01=1A when reduced by 0.01A/LSB. Driving will be performed with 1A as the target current.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xA1 | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA1 | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data of Data[2] and Data[3] 0x0064 is 100 in decimal, and it is 100*0.01=1A when scaled down by 100 times, which means that the actual current of the current motor is 1A. The composite data 0x01F4 of Data[4] and Data[5] is 500 in decimal, which means the motor output shaft speed is 500dps. There is a reduction ratio relationship between the motor output shaft speed and the motor speed. If the reduction ratio is 6, then the motor speed is 6 times higher than the output shaft speed. The composite data of Data[6] and Data[7] 0x002D is 45 in decimal, which means that the motor output shaft moves 45 degrees in the positive direction relative to the zero position. The position of the motor output shaft is related to the number of lines of the motor encoder and the reduction ratio. For example, if the number of lines of the motor encoder is 65536 and the reduction ratio is 6, then 360 degrees of the motor output shaft corresponds to 65536*6 = 393216 pulses.

**Example 2:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xA1 | 0x00 | 0x00 | 0x00 | 0x9C | 0xFF | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA1 | 0x00 | 0x00 | 0x00 | 0x9C | 0xFF | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] and data[5] represent the data size, Data[4] (0x9C) is the low bit, Data[5] (0xFF) is the high bit. So the actual data is 0xFF9C, which means decimal -100, which is -100*0.01=-1A when reduced by 0.01A/LSB. The drive will be performed with -1A as the target current.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xA1 | 0x32 | 0x9C | 0xFF | 0x0C | 0xFE | 0xD3 | 0xFF |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA1 | 0x32 | 0x9C | 0xFF | 0x0C | 0xFE | 0xD3 | 0xFF | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data of Data[2] and Data[3] 0xFF9C is -100 in decimal, and it is -100*0.01=-1A when scaled down by 100 times, which means that the actual current of the current motor is -1A. The composite data 0xFE0C of Data[4] and Data[5] is -500 in decimal, which means the motor output shaft speed is -500dps. There is a reduction ratio relationship between the motor output shaft speed and the motor speed. If the reduction ratio is 6, then the motor speed is 6 times higher than the output shaft speed. The composite data of Data[6] and Data[7] 0xFFD3 is -45 in decimal, which means that the motor output shaft moves in the opposite direction by -45 degrees relative to the zero position. The position of the motor output shaft is related to the number of motor encoder lines and the reduction ratio. For example, if the

number of motor encoder lines is 65536 and the reduction ratio is 6, then 360 degrees of the motor output shaft corresponds to 65536*6 = 393216 pulses.

## 2.18. Speed Closed-loop Control Command (0xA2)

### 2.18.1. Instruction description

This command is a control command, which can be run when the motor is not faulty. The host sends this command to control the speed of the motor output shaft. The control value speedControl is int32_t type, and the corresponding actual speed is 0.01dps/LSB.

### 2.18.2. Send data field definition

| 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>>8) |
| DATA[6] | speed control | DATA[6] = (uint8_t)(speedControl>>16) |
| DATA[7] | speed control high byte | DATA[7] = (uint8_t)(speedControl>>24) |

**Remark:**

1. The maximum torque current of the motor 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.

### 2.18.3. Reply data field definition

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

1. Motor temperature temperature (int8_t type, 1℃/LSB).
2. The torque current value iq of the motor (int16_t type, 0.01A/LSB).
3. Motor output shaft speed (int16_t type, 1dps/LSB).
4. Motor output shaft angle (int16_t type, 1degree/LSB, maximum range ±32767degree).

| 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>>8) |
| DATA[4] | Motor speed low byte | DATA[4] = (uint8_t)(speed) |
| DATA[5] | Motor speed high byte | DATA[5] = (uint8_t)(speed>>8) |
| DATA[6] | Motor angle low byte | DATA[6] = (uint8_t)(degree) |
| DATA[7] | Motor angle high byte | DATA[7] = (uint8_t)(degree>>8) |

### 2.18.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xA2 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA2 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00002710, which means 10000 in decimal. The sending command is reduced by 100 times according to 0.01dps/LSB, that is, 10000*0.01=100dps. The drive operates at the target speed of 100dps of the motor output shaft.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xA2 | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

| 0x3E | 0x01 | 0x08 | 0xA2 | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data of Data[2] and Data[3] 0x0064 is 100 in decimal, and it is 100*0.01=1A when scaled down by 100 times, which means that the actual current of the current motor is 1A. The composite data 0x01F4 of Data[4] and Data[5] is 500 in decimal, which means the motor output shaft speed is 500dps. There is a reduction ratio relationship between the motor output shaft speed and the motor speed. If the reduction ratio is 6, then the motor speed is 6 times higher than the output shaft speed. The composite data of Data[6] and Data[7] 0x002D is 45 in decimal, which means that the motor output shaft moves 45 degrees in the positive direction relative to the zero position. The position of the motor output shaft is related to the number of lines of the motor encoder and the reduction ratio. For example, if the number of lines of the motor encoder is 65536 and the reduction ratio is 6, then 360 degrees of the motor output shaft corresponds to 65536*6 = 393216 pulses.

**Example 2:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xA2 | 0x00 | 0x00 | 0x00 | 0xF0 | 0xD8 | 0xFF | 0xFF |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA2 | 0x00 | 0x00 | 0x00 | 0xF0 | 0xD8 | 0xFF | 0xFF | CRC16L | CRC16H |

**Description:**

Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0xFFFFD8F0, which means -10000 in decimal. The sending command is reduced by 100 times according to 0.01dps/LSB, that is -10000*0.01=-100dps. The drive runs at the target speed of the motor output shaft -100dps.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xA2 | 0x32 | 0x9C | 0xFF | 0x0C | 0xFE | 0xD3 | 0xFF |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA2 | 0x32 | 0x9C | 0xFF | 0x0C | 0xFE | 0xD3 | 0xFF | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data of Data[2] and Data[3] 0xFF9C is -100 in decimal, and it is -100*0.01=-1A when scaled down by 100 times, which means that the actual current of the current motor is -1A. The composite data 0xFE0C of Data[4] and Data[5] is -500 in decimal, which means the motor output shaft speed is -500dps. There is a reduction ratio relationship between the motor output shaft speed and the motor speed. If the reduction ratio is 6, then the motor speed is 6 times higher than the output shaft speed. The composite data of Data[6] and Data[7] 0xFFD3 is -45 in decimal, which means that the motor output shaft moves in the opposite direction by -45 degrees relative to the zero position. The position of the motor output shaft is related to the number of motor encoder lines and the reduction ratio. For example, if the number of motor encoder lines is 65536 and the reduction ratio is 6, then 360 degrees of the motor output shaft corresponds to 65536*6 = 393216 pulses.

## 2.19. Absolute position closed-loop control command (0xA4)

### 2.19.1. Instruction description

This command is a control command, which can be run when the motor is not faulty. The host sends this command to control the position of the motor (multi-turn angle). The control value angleControl is int32_t type, and the corresponding actual position is 0.01degree/LSB, that is, 36000 represents 360°, and the rotation direction of the motor is determined by the difference between the target position and the current position . The control value maxSpeed limits the maximum speed of the motor output shaft rotation, which is of type uint16_t, corresponding to the actual speed of 1dps/LSB.

### 2.19.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0xA4 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | speed limit low byte | DATA[2] = (uint8_t)(maxSpeed) |
| DATA[3] | speed limit high byte | DATA[3] = (uint8_t)(maxSpeed>>8) |

| | | |
|---|---|---|
| DATA[4] | position control low byte | DATA[4] = (uint8_t)(angleControl) |
| DATA[5] | position control | DATA[5] = (uint8_t)(angleControl>>8) |
| DATA[6] | position control | DATA[6] = (uint8_t)(angleControl>>16) |
| DATA[7] | position control high byte | DATA[7] = (uint8_t)(angleControl>>24) |

### 2.19.3. Reply data field definition

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

1. Motor temperature temperature (int8_t type, 1℃/LSB).
2. The torque current value iq of the motor (int16_t type, 0.01A/LSB).
3. Motor output shaft speed (int16_t type, 1dps/LSB).
4. Motor output shaft angle (int16_t type, 1degree/LSB, maximum range ±32767degree).

| 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>>8) |
| DATA[4] | Motor speed low byte | DATA[4] = (uint8_t)(speed) |
| DATA[5] | Motor speed highbyte | DATA[5] = (uint8_t)(speed>>8) |
| DATA[6] | Motor angle low byte | DATA[6] = (uint8_t)(degree) |
| DATA[7] | Motor angle high byte | DATA[7] = (uint8_t)(degree>>8) |

### 2.19.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xA4 | 0x00 | 0xF4 | 0x01 | 0xA0 | 0x8C | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA4 | 0x00 | 0xF4 | 0x01 | 0xA0 | 0x8C | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[2] and Data[3] form one (Data[2] is the low bit, Data[3] is the high bit) 16-bit data is 0x01F4, which means the decimal 500dps motor output shaft speed. The drive will run the position loop at this speed as the maximum speed. Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00008CA0, which means 36000 in decimal. The sending command is reduced by 100 times according to 0.01degree/LSB, that is, 36000*0.01=360°. The motor will move 360° in the positive direction with the output shaft relative to the zero position.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xA4 | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA4 | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data of Data[2] and Data[3] 0x0064 is 100 in decimal, and it is 100*0.01=1A when scaled down by 100 times, which means that the actual current of the current motor is 1A. The composite data 0x01F4 of Data[4] and Data[5] is 500 in decimal, which means the motor output shaft speed is 500dps. There is a reduction ratio relationship between the motor output shaft speed and the motor speed. If the reduction ratio is 6, then the motor speed is 6 times higher than the output shaft speed. The composite data of Data[6] and Data[7] 0x002D is 45 in decimal, which means that the motor output shaft moves 45 degrees in the positive direction relative to the zero position. The position of the motor output shaft is related to the number of lines of the motor encoder and the reduction ratio. For example, if the number of lines of the motor encoder is 65536 and the reduction ratio is 6, then 360 degrees of the motor output shaft corresponds to 65536*6 = 393216 pulses.

**Example 2:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xA4 | 0x00 | 0xF4 | 0x01 | 0x60 | 0x73 | 0xFF | 0xFF |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA4 | 0x00 | 0xF4 | 0x01 | 0x60 | 0x73 | 0xFF | 0xFF | CRC16L | CRC16H |

**Description:**

Data[2] and Data[3] form one (Data[2] is the low bit, Data[3] is the high bit) 16-bit data is 0x01F4, which means the decimal 500dps motor output shaft speed. The drive will run the position loop at this speed as the maximum speed. Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0xFFFF7360, which means -36000 in decimal. The sending command is reduced by 100 times according to 0.01degree/LSB, ie -36000*0.01=-360°. The motor will move -360° in the opposite direction with the output shaft relative to the zero position.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xA4 | 0x32 | 0x9C | 0xFF | 0x0C | 0xFE | 0xD3 | 0xFF |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA4 | 0x32 | 0x9C | 0xFF | 0x0C | 0xFE | 0xD3 | 0xFF | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data of Data[2] and Data[3] 0xFF9C is -100 in decimal, and it is -100*0.01=-1A when scaled down by 100 times, which means that the actual current of the current motor is -1A. The composite data 0xFE0C of Data[4] and Data[5] is -500 in decimal, which means the motor output shaft speed is -500dps. There is a reduction ratio relationship between the motor output shaft speed and the motor speed. If the reduction ratio is 6, then the motor speed is 6 times higher than the output shaft speed. The composite data of Data[6] and Data[7] 0xFFD3 is -45 in decimal, which means that the motor output shaft moves in the opposite direction by -45 degrees relative to the zero position. The position of the motor output shaft is related to

the number of lines of the motor encoder and the reduction ratio. For example, if the number of lines of the motor encoder is 65536 and the reduction ratio is 6, then 360 degrees of the motor output shaft corresponds to 65536*6 = 393216 pulses.

## 2.20. Incremental position closed-loop control command (0xA8)

### 2.20.1. Instruction description

This command is a control command, which can be run when the motor is not faulty. The host sends this command to control the incremental position (multi-turn angle) of the motor, and run the input position increment with the current position as the starting point. The control value angleControl is of type int32_t, and the corresponding actual position is 0.01degree/LSB, that is, 36000 represents 360°, and the rotation direction of the motor is determined by the incremental position symbol.
The control value maxSpeed limits the maximum speed of the motor output shaft rotation, which is of type uint16_t, corresponding to the actual speed of 1dps/LSB.

### 2.20.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0xA8 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | speed limit low byte | DATA[2] = (uint8_t)(maxSpeed) |
| DATA[3] | speed limit high byte | DATA[3] = (uint8_t)(maxSpeed>>8) |
| DATA[4] | position control low byte | DATA[4] = (uint8_t)(angleControl) |
| DATA[5] | position control | DATA[5] = (uint8_t)(angleControl>>8) |
| DATA[6] | position control | DATA[6] = (uint8_t)(angleControl>>16) |
| DATA[7] | position control high byte | DATA[7] = (uint8_t)(angleControl>>24) |

### 2.20.3. Reply data field definition

The motor replies to the host after receiving the command, and the frame data contains the following parameters.
5. Motor temperature temperature (int8_t type, 1℃/LSB).
6. The torque current value iq of the motor (int16_t type, 0.01A/LSB).
7. Motor output shaft speed (int16_t type, 1dps/LSB).
8. Motor output shaft angle (int16_t type, 1degree/LSB, maximum range

±32767degree).

| 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>>8) |
| DATA[4] | Motor speed low byte | DATA[4] = (uint8_t)(speed) |
| DATA[5] | Motor speed high byte | DATA[5] = (uint8_t)(speed>>8) |
| DATA[6] | Motor angle low byte | DATA[6] = (uint8_t)(degree) |
| DATA[7] | Motor angle high byte | DATA[7] = (uint8_t)(degree>>8) |

### 2.20.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xA8 | 0x00 | 0xF4 | 0x01 | 0xA0 | 0x8C | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA8 | 0x00 | 0xF4 | 0x01 | 0xA0 | 0x8C | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[2] and Data[3] form one (Data[2] is the low bit, Data[3] is the high bit) 16-bit data is 0x01F4, which means the decimal 500dps motor output shaft speed. The drive will run the position loop at this speed as the maximum speed. Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit) 32-bit data is 0x00008CA0, which means 36000 in decimal. The sending command is reduced by 100 times according to 0.01degree/LSB, that is, 36000*0.01=360°. The motor will move 360° in the positive direction with the output shaft relative to the current position.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xA8 | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA8 | 0x32 | 0x64 | 0x00 | 0xF4 | 0x01 | 0x2D | 0x00 | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data of Data[2] and Data[3] 0x0064 is 100 in decimal, and it is 100*0.01=1A when scaled down by 100 times, which means that the actual current of the current motor is 1A. The composite data 0x01F4 of Data[4] and Data[5] is 500 in decimal, which means the motor output shaft speed is 500dps. There is a reduction ratio relationship between the motor output shaft speed and the motor speed. If the reduction ratio is 6, then the motor speed is 6 times higher than the output shaft speed. The composite data of Data[6] and Data[7] 0x002D is 45 in decimal, which means that the motor output shaft moves 45 degrees in the positive direction relative to the zero position. The position of the motor output shaft is related to the number of lines of the motor encoder and the reduction ratio. For example, if the number of lines of the motor encoder is 65536 and the reduction ratio is 6, then 360 degrees of the motor output shaft corresponds to 65536*6 = 393216 pulses.

**Example 2:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xA8 | 0x00 | 0xF4 | 0x01 | 0x60 | 0x73 | 0xFF | 0xFF |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA8 | 0x00 | 0xF4 | 0x01 | 0x60 | 0x73 | 0xFF | 0xFF | CRC16L | CRC16H |

**Description:**

Data[2] and Data[3] form one (Data[2] is the low bit, Data[3] is the high bit) 16-bit data is 0x01F4, which means the decimal 500dps motor output shaft speed. The drive will run the position loop at this speed as the maximum speed. Data[4] to data[7] form one (Data[4] is the lowest bit, Data[7] is the highest bit)

32-bit data is 0xFFFF7360, which means -36000 in decimal. The sending command is reduced by 100 times according to 0.01degree/LSB, ie -36000*0.01=-360°. The motor will move -360° in the opposite direction relative to the current position with the output shaft.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xA8 | 0x32 | 0x9C | 0xFF | 0x0C | 0xFE | 0xD3 | 0xFF |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xA8 | 0x32 | 0x9C | 0xFF | 0x0C | 0xFE | 0xD3 | 0xFF | CRC16L | CRC16H |

**Description:**

Data[1] = 0x32 is 50 in decimal, which means the motor temperature is 50 degrees at the moment. The composite data of Data[2] and Data[3] 0xFF9C is -100 in decimal, and it is -100*0.01=-1A when scaled down by 100 times, which means that the actual current of the current motor is -1A. The composite data 0xFE0C of Data[4] and Data[5] is -500 in decimal, which means the motor output shaft speed is -500dps. There is a reduction ratio relationship between the motor output shaft speed and the motor speed. If the reduction ratio is 6, then the motor speed is 6 times higher than the output shaft speed. The composite data of Data[6] and Data[7] 0xFFD3 is -45 in decimal, which means that the motor output shaft moves in the opposite direction by -45 degrees relative to the zero position. The position of the motor output shaft is related to the number of lines of the motor encoder and the reduction ratio. For example, if the number of lines of the motor encoder is 65536 and the reduction ratio is 6, then 360 degrees of the motor output shaft corresponds to 65536*6 = 393216 pulses.

## 2.21. System operating mode acquisition (0x70)

### 2.21.1. Instruction description

This command reads the current motor running mode.

### 2.21.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x70 |

| | | |
|---|---|---|
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |
| DATA[7] | NULL | 0x00 |

### 2.21.3. Reply data field definition

The motor replies to the host after receiving the command, and the drive reply data contains the running state of the parameter runmode, which is of type uint8_t.

The motor operation mode has the following 4 states:

1. Current loop mode (0x01).
2. Speed loop mode (0x02).
3. Position loop mode (0x03).

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x70 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |
| DATA[7] | Motor operating mode | DATA[7] = (uint8_t)(runmode) |

### 2.21.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x70 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x70 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

This command reads the current motor running mode.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x70 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x03 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x70 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x03 | CRC16L | CRC16H |

**Description:**

Data[7] = 0x03, according to the definition of the reply frame, it means that the current system is in the position loop mode.

## 2.22. Motor power acquisition (0x71)

### 2.22.1. Instruction description

This command reads the current motor power.

### 2.22.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x71 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |

| | | |
|---|---|---|
| DATA[7] | NULL | 0x00 |

### 2.22.3. Reply data field definition

The motor replies to the host after receiving the command, and the drive reply data contains the motor power parameter motorpower, which is of type uint16_t, the unit is watt, and the unit is 0.1w/LSB.

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x71 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | Motor running power low byte | DATA[6] = (uint8_t)(motorpower) |
| DATA[7] | Motor running power high byte | DATA[7] = (uint8_t)(motorpower>>8) |

### 2.22.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x71 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x71 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

This command reads the current motor power.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x71 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0xD0 | 0x07 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x71 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0xD0 | 0x07 | CRC16L | CRC16H |

**Description:**

The composition of Data[6] and Data[7] = 0x07D0, decimal 2000, reduced by 10 times according to the unit of 0.1W/LSB, 2000*0.1=200W. Indicates that the current power of the motor is 200W.

## 2.23. System reset command (0x76)

### 2.23.1. Instruction description

This command is used to reset the system program.

### 2.23.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x76 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |
| DATA[7] | NULL | 0x00 |

### 2.23.3. Reply data field definition

The motor will reset after receiving the command and will not return to the command.

### 2.23.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0x76 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0x76 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

After sending the command, the system is reset and the program runs again.

## 2.24. System brake release command (0x77)

### 2.24.1. Instruction description

This command is used to open the system brake. The system will release the holding brake, and the motor will be in a movable state without being restricted by the holding brake.

### 2.24.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x77 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |
| DATA[7] | NULL | 0x00 |

### 2.24.3. Reply data field definition

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

### 2.24.4. Communication example

## 2.25. System brake lock command (0x78)

### 2.25.1. Instruction description

This command is used to close the system holding brake. The holding brake locks the motor and the motor can no longer run. The holding brake is also in this state after the system is powered off.

### 2.25.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x78 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |
| DATA[7] | NULL | 0x00 |

### 2.25.3. Reply data field definition

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

### 2.25.4. Communication example

## 2.26. System runtime read command (0xB1)

### 2.26.1. Instruction description

This command is used to obtain the system running time in ms.

### 2.26.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0xB1 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |

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

### 2.26.3. Reply data field definition

The motor replies to the host after receiving the command, and the drive reply data contains the system running time SysRunTime, which is uint32_t type, and the unit is ms.

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0xB1 |
| DATA[0] | NULL | 0x00 |
| DATA[0] | NULL | 0x00 |
| DATA[0] | NULL | 0x00 |
| DATA[4] | SysRunTime low byte1 | DATA[4] = (uint8_t)(SysRunTime) |
| DATA[5] | SysRunTime byte2 | DATA[5] = (uint8_t)(SysRunTime>>8) |
| DATA[6] | SysRunTime byte3 | DATA[6] = (uint8_t)(SysRunTime>>16) |
| DATA[7] | SysRunTime byte4 | DATA[7] = (uint8_t)(SysRunTime>>24) |

### 2.26.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xB1 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB1 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

This command reads the running time of the current system.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xB1 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x10 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB1 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x10 | CRC16L | CRC16H |

**Description:**

Data[4] to Data[7] (Data[4] is low and Data[7] is high) = 0x10000000, decimal 268435456, indicating that the system has run for 268435456ms after restarting or resetting, about 74 Hour.

## 2.27. System software version date read command (0xB2)

### 2.27.1. Instruction description

This command is used to get the update date of the system software version.

### 2.27.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0xB2 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |
| DATA[7] | NULL | 0x00 |

### 2.27.3. Reply data field definition

The motor will reply to the host after receiving the command. The driver reply data contains the latest version date of the system software, VersionDate, which is of type uint32_t. The date format is in the format of year, month, and day, such as 20211126.

| data field | Description | data |
|---|---|---|

| | | |
|---|---|---|
| DATA[0] | command byte | 0xB2 |
| DATA[0] | NULL | 0x00 |
| DATA[0] | NULL | 0x00 |
| DATA[0] | NULL | 0x00 |
| DATA[4] | VersionDate low byte1 | DATA[4] = (uint8_t)(&VersionDate) |
| DATA[5] | VersionDate byte2 | DATA[5] = (uint8_t)(VersionDate>>8) |
| DATA[6] | VersionDate byte3 | DATA[6] = (uint8_t)(VersionDate>>16) |
| DATA[7] | VersionDate byte4 | DATA[7] = (uint8_t)(VersionDate>>24) |

### 2.27.4. Communication example

**Example 1:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xB2 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB2 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

This command reads the current software version date.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xB2 | 0x00 | 0x00 | 0x00 | 0x2E | 0x89 | 0x34 | 0x01 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB2 | 0x00 | 0x00 | 0x00 | 0x2E | 0x89 | 0x34 | 0x01 | CRC16L | CRC16H |

**Description:**

Data[4] to Data[7] (Data[4] is low and Data[7] is high) = 0x0134892E, decimal 20220206, indicating that the software version date is February 6, 2022.

## 2.28. Communication interruption protection time setting command (0xB3)

### 2.28.1. Instruction description

This command is used to set the communication interruption protection time in ms. If the communication is interrupted for more than the set time, it will cut off the output brake lock. To run again, you need to establish stable and continuous communication first. Writing 0 means that the communication interruption protection function is not enabled.

### 2.28.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0xB3 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | NULL | 0x00 |
| DATA[3] | NULL | 0x00 |
| DATA[4] | CanRecvTime_MS low byte1 | DATA[4] = (uint8_t)(CanRecvTime_MS) |
| DATA[5] | CanRecvTime_MS byte2 | DATA[5] = (uint8_t)(CanRecvTime_MS>>8) |
| DATA[6] | CanRecvTime_MS byte3 | DATA[6] = (uint8_t)(CanRecvTime_MS>>16) |
| DATA[7] | CanRecvTime_MS byte4 | DATA[7] = (uint8_t)(CanRecvTime_MS>>24) |

### 2.28.3. Reply data field definition

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

### 2.28.4. Communication example

**Example 1:**
**Send command:**
**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xB3 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB3 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Note:**

The data values are all 0, which means that the communication interruption protection function is not enabled. If the communication is interrupted, the motor will continue to execute the current command.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xB3 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB3 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

The frame data is the same as the command sent by the host.

**Example 2:**

**Send command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xB3 | 0x00 | 0x00 | 0x00 | 0xE8 | 0x03 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB3 | 0x00 | 0x00 | 0x00 | 0xE8 | 0x03 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

Data[4] to Data[7] (Data[4] is low and Data[7] is high) constitute data 0x000003E8, decimal is 1000ms. Indicates that the communication interruption protection time is set to 1000ms, which is stored in the ROM and saved after power failure. Then, if the communication interval exceeds 1000ms, the communication interruption protection will be triggered, and the output lock brake will be cut off. When the communication interval is restored to within 1000ms, normal operation can be resumed.

**Reply command:**

**CAN:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0xB3 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**RS485：**

| frame header | ID | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L | CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB3 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L | CRC16H |

**Description:**

The frame data is the same as the command sent by the host.

## 2.29. Communication baud rate setting command (0xB4)

### 2.29.1. Instruction description

This instruction can set the communication baudrate of CAN and RS485 bus. The parameters will be saved in ROM after setting, and will be saved after power off, and will run at the modified baudrate when powered on again.

Baudrate:

RS485: 0 represents 115200bps baud rate,
1 stands for 500Kbps baud rate,
2 stands for 1Mbps baud rate,
3 represents 1.5Mbps baud rate,
4 represents 2Mbps baud rate;

CAN: 0 means 500Kbps baud rate,
1 stands for 1Mbps baud rate;

### 2.29.2. Send data field definition

| Data Field | Description | Data |
|---|---|---|
| DATA[0] | command byte | 0xB4 |
| 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] | baudrate | DATA[7] = (uint8_t)baudrate |

### 2.29.3. Reply data field definition

Since the communication baud rate is modified, the reply command is random and need not be processed.

### 2.29.4. Communication example

Example 1:
Send command:
CAN:

| ID NO. | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xB4 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

RS485:

| Frame Header | ID NO. | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB4 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L CRC16H |

Description: Data[7] = 0, which means the baud rate of RS485 is changed to 115200bps, and the baud rate of CAN is changed to 500Kbps.

Example 2:
Send command:
CAN:

| ID NO. | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xB4 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x01 |

RS485:

| Frame Header | ID NO. | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB4 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x01 | CRC16L CRC16H |

Description: Data[7] = 1, which means the RS485 baud rate is changed to 500Kbps, and the CAN baud rate is changed to 1Mbps.

Example 3:
Send command:
CAN:

| ID NO. | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x141 | 0xB4 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x02 |

RS485:

| Frame Header | ID NO. | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0x01 | 0x08 | 0xB4 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x02 | CRC16L CRC16H |

Description: Data[7] = 2, which means the RS485 baud rate is changed to 1Mbps, and CAN is invalid.

# 3. Multi-motor command (0x280 + command)

## 3.1. Instruction description

The ID number is 280, which means that multiple motors correspond to the same command at the same time. The content and function of the instruction are the same as those of the single-motor instruction. For details, please refer to the single-motor instruction.

## 3.2. Communication example

Suppose there are 4 motors on the CAN bus, and the ID numbers are 141, 142, 143, and 144 respectively.

**Example 1:**
**Send command:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x280 | 0x80 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**Description:**
4 motors receive the 0x80 motor shutdown command at the same time (see 2.30 for details), and then all 4 motors immediately execute the motor shutdown command.
**Reply command:**

4 motors reply at the same time, and the reply ID is their own ID number respectively. The reply sequence depends on the respective delays on the bus.

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x80 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**Description:**

The motor whose ID number is 0x241 returns the corresponding command.

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x242 | 0x80 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**Description:**

The motor whose ID number is 0x242 returns the corresponding command.

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x243 | 0x80 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**Description**:

The motor whose ID number is 0x243 returns the corresponding command.

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x244 | 0x80 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**Description:**

The motor whose ID number is 0x244 returns the corresponding command.

**Example 2:**

**Send command:**

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x280 | 0x60 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

**Description:**

4 motors receive the 0x60 read multi-turn encoder position data command at the same time (see 2.21 for details), and then the 4 motors reply to their respective multi-turn encoder position data.

Reply command:

4 motors reply at the same time, and the reply ID is their own ID number respectively. The reply sequence depends on the respective delays on the bus.

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x241 | 0x60 | 0x00 | 0x00 | 0x00 | 0x10 | 0x27 | 0x00 | 0x00 |

**Description:**

The motor reply data with ID number 0x241 consists of Data[4] to data[7] (Data[4] is the lowest bit, Data[7] is the highest bit). The 32-bit data is 0x00002710, which means the decimal is 10000. The multi-turn encoder value

representing the current relative multi-turn zero offset (initial position) of the motor is 10000 pulses.

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x242 | 0x60 | 0x00 | 0x00 | 0x00 | 0x20 | 0x4E | 0x00 | 0x00 |

**Description:**

The motor reply data with ID number 0x242 consists of Data[4] to data[7] (Data[4] is the lowest bit, Data[7] is the highest bit). The 32-bit data is 0x00004E20, which means 20000 in decimal. The multi-turn encoder value representing the current relative multi-turn zero offset (initial position) of the motor is 20000 pulses.

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x243 | 0x60 | 0x00 | 0x00 | 0x00 | 0x30 | 0x75 | 0x00 | 0x00 |

**Description**:

The motor reply data with ID number 0x243 consists of Data[4] to data[7] (Data[4] is the lowest bit, Data[7] is the highest bit). The 32-bit data is 0x00007530, which means 30000 in decimal. The multi-turn encoder value representing the current relative multi-turn zero offset (initial position) of the motor is 30000 pulses.

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x244 | 0x60 | 0x00 | 0x00 | 0x00 | 0x40 | 0x9C | 0x00 | 0x00 |

**Description**:

The motor reply data with ID number 0x244 consists of Data[4] to data[7] (Data[4] is the lowest bit, Data[7] is the highest bit). The 32-bit data is 0x00009C40, which means 40000 in decimal. The multi-turn encoder value representing the current relative multi-turn zero offset (initial position) of the motor is 40000 pulses.

# 4. CANID setting command (0x79)

## 4.1. Instruction description

This command is used to set and read CAN ID.

The host sends this command to set and read the CAN ID, the parameters are as follows.

1. The read and write flag bit wReadWriteFlag is bool type, 1 read 0 write.
2. CANID, size range (#1~#32), uint16_t type (synchronized with the upper computer function), device identifier 0x140 + ID (1~32).

## 4.2. Send data field definition

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x79 |

| | | |
|---|---|---|
| DATA[1] | NULL | 0x00 |
| DATA[2] | read and write flags | DATA[2] = wReadWriteFlag |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |
| DATA[7] | CANID | DATA[7] = CANID(1~32) |

## 4.3. Reply data field definition

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

| data field | Description | data |
|---|---|---|
| DATA[0] | command byte | 0x79 |
| DATA[0] | NULL | 0x00 |
| DATA[0] | read and write flags | DATA[2] = wReadWriteFlag |
| DATA[0] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | CANID low byte1 | DATA[6] = (uint8_t *)(CANID) |
| DATA[7] | CANID byte2 | DATA[7] = (uint8_t)(CANID>>8) |

## 4.4. Communication example

## Example 1:

## Send command:

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x300 | 0x79 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x02 |

## Description:

Data[2] = 0 means write CANID. Data[7] = 1 means that the motor CANID is set to 2, that is, the send ID is 0x142, and the reply ID is 0x242.

### Reply command:

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x300 | 0x79 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x02 |

### Description:

Same as sending command.

### Example 2:

### Send command:

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x300 | 0x79 | 0x00 | 0x01 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 |

### Description:

Data[2] = 1 means reading CANID.

### Reply command:

| ID | Data[0] | Data[1] | Data[2] | Data[3] | Data[4] | Data[5] | Data[6] | Data[7] |
|---|---|---|---|---|---|---|---|---|
| 0x300 | 0x79 | 0x00 | 0x01 | 0x00 | 0x00 | 0x00 | 0x42 | 0x02 |

### Description:

Data[6] and Data[7] form 0x242, which means that the motor send ID is 0x142, and the reply ID is 0x242.

# 5.Motion Mode Control Command_CAN (0x400 + ID)

## 5.1. Instruction Description

The command consists of 5 input parameters:
p_des (desired position),
v_des(desired velocity),
t_ff (feedforward torque),
kp (position deviation coefficient),
kd (speed deviation coefficient).
Each parameter has a preset range size:
p_des: -12.5 to 12.5 in rad;
v_des: -45 to 45, in rad/s;
t_ff: -24 to 24, unit N-m;

kp: 0 to 500;
kd: 0 to 5;

Function expression: IqRef = [kp*(p_des - p_fd_actual position) + kd*(v_des - v_fb_actual speed) + t_ff]*KT_torque coefficient; IqRef is the output current of the last given motor.

## 5.2. Send data field definition

| Data field | Data partition | Data combination | Data definition | Data range |
|---|---|---|---|---|
| DATA[0] | 0-3bit | p_des[8-15] | p_des Upper 8-bit data | 16-bit range |
| | 4-7bit | | | |
| DATA[1] | 0-3bit | p_des[0-7] | p_des Lower 8-bit data | |
| | 4-7bit | | | |
| DATA[2] | 0-3bit | v_des[4-11] | v_des Upper 8-bit data | 12-bit range |
| | 4-7bit | | | |
| DATA[3] | 0-3bit | v_des[0-3] | v_des Lower 8-bit data | |
| | 4-7bit | kp[8-11] | kp Upper 4-bit data | 12-bit range |
| DATA[4] | 0-3bit | kp[0-7] | kp Lower 8-bit data | |
| | 4-7bit | | | |
| DATA[5] | 0-3bit | kd[4-11] | kd Upper 8-bit data | 12-bit range |
| | 4-7bit | | | |
| DATA[6] | 0-3bit | kd[0-3] | kd Lower 4-bit data | |
| | 4-7bit | t_ff[8-11] | t_ff Upper 4-bit data | 12-bit range |
| DATA[7] | 0-3bit | t_ff[0-7] | t_ff Lower 8-bit data | |
| | 4-7bit | | | |

## 5.3. Reply data field definition

| Data field | Data partition | Data combination | Data definition | Data range |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| DATA[0] | 0-3bit | p_des[8-15] | p_des Upper 8-bit data | 16-bit range |
| | 4-7bit | | | |
| DATA[1] | 0-3bit | p_des[0-7] | p_des Lower 8-bit data | |
| | 4-7bit | | | |
| DATA[2] | 0-3bit | v_des[4-11] | v_des Upper 8-bit data | 12-bit range |
| | 4-7bit | | | |
| DATA[3] | 0-3bit | v_des[0-3] | v_des Lower 4-bit data | |
| | 4-7bit | kp[8-11] | t_ff Upper 4-bit data | 12-bit range |
| DATA[4] | 0-3bit | t_ff[0-7] | t_ff Lower 8-bit data | |
| | 4-7bit | | | |
| DATA[5] | 0-3bit | NULL | NULL | NULL |
| | 4-7bit | | | |
| DATA[6] | 0-3bit | NULL | NULL | |
| | 4-7bit | NULL | NULL | NULL |
| DATA[7] | 0-3bit | NULL | NULL | |
| | 4-7bit | | | |

## 5.4. Communication example

Example 1:

Send command: ID number 0x401

| Data field | Data | Data partition | | Data definition | Data range | Data calculation instructions |
|---|---|---|---|---|---|---|
| DATA[0] | 0xE6 | 0-3bit | 0xE | p_des value is 0xE665 decimal is (58981) | (-)12.5rad~12.5rad total 25rad | p_des=(58981/65535)*25 + (-12.5) = 9.99 rad |
| | | 4-7bit | 0x6 | | | |
| DATA[1] | 0x65 | 0-3bit | 0x6 | | | |
| | | 4-7bit | 0x5 | | | |
| DATA[2] | 0x82 | 0-3bit | 0x8 | v_des value is 0x82E decimal is (2094) | (-)45rad/s~45rad/s total 90rad/s | v_des=(2094/4095)*90 + (-45) = 1.021 rad/s |
| | | 4-7bit | 0x2 | | | |
| DATA[3] | 0xE0 | 0-3bit | 0xE | | | |
| | | 4-7bit | 0x0 | kp value is 0x52 decimal is | 0~500 total 500 | kp=(82/4095)*500 + 0 |
| DATA[4] | 0x52 | 0-3bit | 0x5 | | | |
| | | 4-7bit | 0x2 | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | (82) | | |
| DATA[5] | 0x33 | 0-3bit | 0x3 | kd value is 0x333 decimal is (819) | 0~5 total 5 | kd=(819/4095)*5 + 0 |
| | | 4-7bit | 0x3 | | | |
| DATA[6] | 0x3B | 0-3bit | 0x3 | | | |
| | | 4-7bit | 0xB | t_ff value is 0xB55 decimal is (2901) | (-)24N-m~24N-m totol 48N-m | t_ff=(2901/4095)*48 + (-24) = 10.004 N-m |
| DATA[7] | 0x55 | 0-3bit | 0x5 | | | |
| | | 4-7bit | 0x5 | | | |

## Reply command: ID No. 0x501

| Data field | Data | Data partition | | Data definition | Data range | Data calculation instructions |
|---|---|---|---|---|---|---|
| DATA[0] | 0xE6 | 0-3bit | 0xE | p_des value is 0xE665 decimal is (58981) | (-)12.5rad~12.5rad total 25rad | p_des=(58981/65535)*25 + (-12.5) = 9.99 rad |
| | | 4-7bit | 0x6 | | | |
| DATA[1] | 0x65 | 0-3bit | 0x6 | | | |
| | | 4-7bit | 0x5 | | | |
| DATA[2] | 0x82 | 0-3bit | 0x8 | v_des value is 0x82E decimal is (2094) | (-)45rad/s~45rad/s total 90rad/s | v_des=(2094/4095)*90 + (-45) = 1.021 rad/s |
| | | 4-7bit | 0x2 | | | |
| DATA[3] | 0xEB | 0-3bit | 0xE | | | |
| | | 4-7bit | 0xB | t_ff value is 0xB55 decimal is (2901) | (-)24N-m~24N-m total 48N-m | t_ff=(2901/4095)*48 + (-24) = 10.004 N-m |
| DATA[4] | 0x52 | 0-3bit | 0x5 | | | |
| | | 4-7bit | 0x2 | | | |
| DATA[5] | | 0-3bit | | | | |
| | | 4-7bit | | | | |
| DATA[6] | | 0-3bit | | | | |
| | | 4-7bit | | | | |
| DATA[7] | | 0-3bit | | | | |
| | | 4-7bit | | | | |

# 6. RS485-ID setting command (0x79)

## 6.1. Instruction Description

This command is used to set and read RS485 ID. Communication ID uses 0xCD, all devices on the bus will receive and process this command,
When modifying, you need to pay attention to whether multiple devices are connected, so that the IDs of multiple devices may be modified to the same at the same time.
The host sends this command to set and read the RS485 ID, the parameters are as follows.
3. The read and write flag bit wReadWriteFlag is bool type, 1 read 0 write.
4. RS485-ID, size range (#1~#32), uint16_t type (synchronized with the upper computer function), device identifier ID (1~32).

## 6.2. Send Data Field Definition

| Data field | Explanation | Data |
|---|---|---|
| DATA[0] | command byte | 0x79 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | read and write flags | DATA[2] = wReadWriteFlag |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | NULL | 0x00 |
| DATA[7] | RS485ID | DATA[7] = RS485ID(1~32) |

## 6.3. Reply data field definition

4. The motor replies to the host after receiving the command, which is divided into the following two situations:
5. Set RS485ID, the range is 1-32, and return to the original command.
6. Read RS485ID, the return parameters are as follows.

| Data field | Explanation | Data |
|---|---|---|
| DATA[0] | command byte | 0x79 |
| DATA[1] | NULL | 0x00 |
| DATA[2] | read and write flags | DATA[2] = wReadWriteFlag |
| DATA[3] | NULL | 0x00 |
| DATA[4] | NULL | 0x00 |
| DATA[5] | NULL | 0x00 |
| DATA[6] | RS485 Low byte 1 | DATA[6] = (uint8_t *)(RS485ID) |
| DATA[7] | RS485 ID 2 | DATA[7] = (uint8_t)(RS485ID>>8) |

## 6.4. Communication example

Example 1:
Send command:

| Frame Header | ID No. | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0xCD | 0x08 | 0x79 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x02 | CRC16L CRC16H |

Description: Data[2] = 0 means write RS485ID. Data[7] = 1 means to set the motor RS485ID to 2.

Reply command:

| Frame Header | ID No. | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0xCD | 0x08 | 0x79 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x02 | CRC16L CRC16H |

Description: Same as sending command.

Example 2:
Send command:

| Frame Header | ID No. | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0xCD | 0x08 | 0x79 | 0x00 | 0x01 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | CRC16L CRC16H |

Description: Data[2] = 1 means to read RS485ID.

Reply command:

| Frame Header | ID No. | Length | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | CRC16L CRC16H |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x3E | 0xCD | 0x08 | 0x79 | 0x00 | 0x01 | 0x00 | 0x00 | 0x00 | 0x00 | 0x02 | CRC16L CRC16H |

# 7. Version revision information