AK SERIES DYNAMICAL MODULAR INSTRUCTION

CONTENTS

☆ Please read the manual carefully before use

☆ Please keep it for reference

1. SAFETY NOTES AND DISCLAIMERS

a. Please use in the working environment with the temperature range (-20℃-50℃) and humidity below 90% as specified in this document.
b. Avoid foreign body into the motor interior, resulting in abnormal rotor operation.
c. Ensure connecting is normal and stable before use.
d. Do not disassemble the motor personally, otherwise it will affect the control accuracy of the motor, and even cause abnormal operation of the motor.

2.PRODUCT OVERVIEW

2.1PRODUCT DESCRIPTION

AK series module is a high-end actuator including brushless DC motor, integrated planetary gear reducer, encoder and driver. It can be widely used in the fields of exoskeleton, walking robot, automation equipment, scientific research and education. The driver uses field oriented control (FOC) algorithm, with high precision angle sensor, precise position and torque control can be achieved. The design of 36N42P, rare earth material magnet and high precision planetary gear reducer can provide more stable and larger torque for the motor. AK80 series motor supports a variety of communication protocols, it provides a human-machine interaction interface through communication with the PC, enabling users to control the motor faster and more accurately.

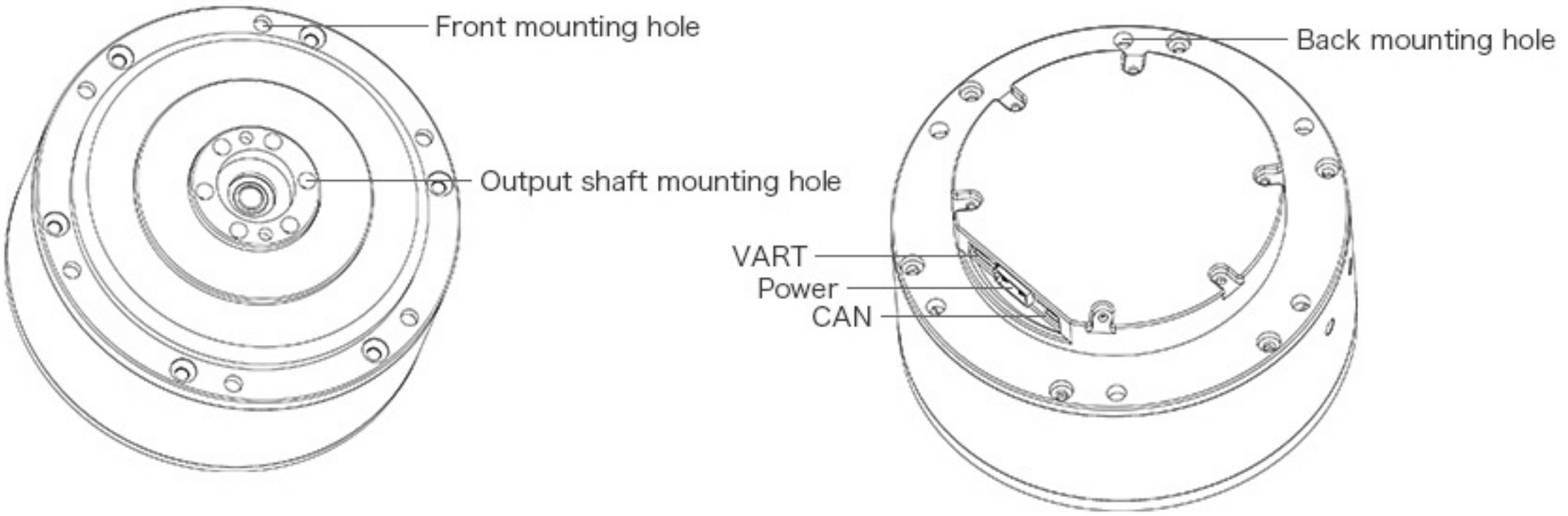
2.2 MODEL DEFINITION OF DYNAMICAL MODULAR

FOR EXAMPLE:

AK80-9

- 9 — stands for reduction ratio.
- 8 — stands for motor size
- 0 — stands for included driver board
- AK — stands for module actuator

2.3 MOTOR APPEARANCE, MOUNTING STRUCTURE AND CONNECTION INTERFACE DEFINITION



2.4 SERIAL SIGNAL PORT

Connect the PC terminal through UART, set the zero position of the modular, calibrate the encoder, set the modular ID number, maximum current and other parameters.

Wire sequence: From top to bottom A(GND), B(RX), C(TX) . Wire length 200mm.

2.5 POWER CONNECTOR

Connect the power supply (rated voltage 24 V or 48V) through the XT30 connector to power the modular.

Wire sequence: from top to bottom are A(power negative pole--black color), B(power positive pole--red color).

2.6 CAN SIGNAL INTERFACE

External equipment can send control instructions through CAN signal and feedback status information of motor.
CAN bus bit rate is 1 Mbps, Host ID number defaults to 0 x00.

Wire sequence: A (CAN_H), B (CAN_L) from top to bottom. The length of wire: 200 mm.

3. INSTRUCTIONS FOR USE

3.1 CAN COMMUNICATION PROTOCOL

CAN Rate: 1 MHZ

Motor receives message format.

It is used to send control commands to the motor to control the position, speed and current.

3.2 SPECIAL CAN CODE

Enter motor control mode {0 xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0XFC}

Exit motor control mode {0 xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0XFD}

Set the current position of the motor to zero {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0XFE }

Attention: When using the CAN communication to control motor, you must enter the motor control mode first!

IDENTIFIER: SET MOTOR ID NUMBER (DEFAULT IS 1) FRAME TYPE: STANDARD FRAME

FRAME FORMAT: DATA DLC: 8 BYTES

Data Field	DATA[0]	DATA[1]	DATA[2]	DATA[3]	
Data bits	7-0	7-0	7-0	7-4	3-0
Data content	Motor position 8-H	Motor position 8-L	Motor speed 8-H	Motor speed 4-L	KP 4-H

Data Field	DATA[4]	DATA[5]	DATA[6]		DATA[7]
Data bits	7-0	7-0	7-4	3-0	0-7
Data content	KP 8-L	KD 8-H	KD 4-L	Current 4-H	Current 8-L

MOTOR SEND MESSAGE FORMAT

IDENTIFIER: 0 X00+ DRIVER ID FRAME TYPE: STANDARD FRAME

FRAME FORMAT: DATA DLC: 6 BYTES

Data Field	DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[4]
Data bits	7-0	7-0	7-0	7-0	7-4
Data content	Drive ID No.	Motor position 8-H	Motor speed 8-L	Motor speed 8-H	Motor speed 4-L

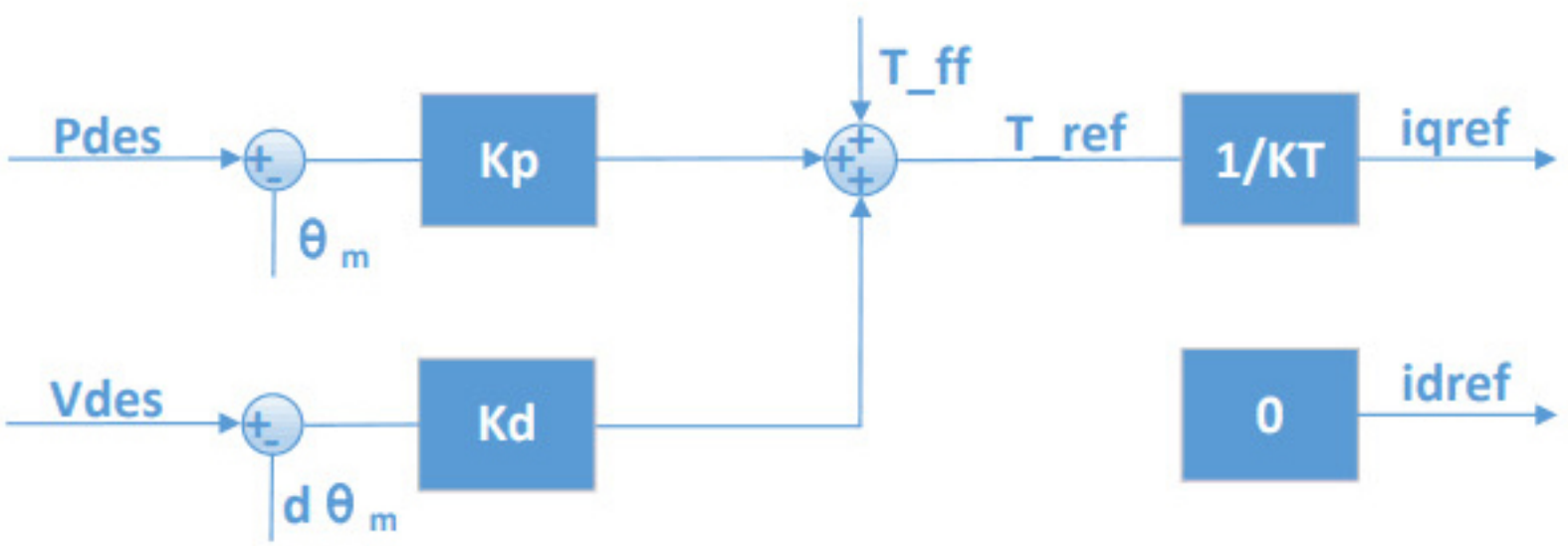
Data Field	DATA[4]	DATA[5]
Data bits	3-0	7-0
Data content	Current 4-H	Current 8-L

3.3 PARAMETER RANGE

Module Motor	AK60-6	AK80-6	AK80-9	AK10-9		AK70-10		AK80-80	
Motor Position (rad)	-12.5f~12.5f								
Motor Speed rad/S	-41.87~41.87	-38.2~38.2	-25.64~25.64	24V	48V	24V	48V	24V	48V
				-23.24~23.24	-46.57~46.57	-20.94~20.94	-41.87~41.87	-2.93~2.93	-5.86~5.86
Motor Torque Nm	-9~9	-12~12	-18~18	-54~54		-24.8~24.8		-144~144	
KP Range	0~500								
KD Range	0~5								

3.4 INTERNAL CONTROLLER PID SCHEMATIC

For only position, only speed, and only torque control loop, only the corresponding variable need to be set to a value and the rest set as 0. For example, if you want to control the position, you will set the motor position to a value when sending the data, and send 0 for torque and speed. Kp controls the parameters of the position loop and Kd controls the parameters of the speed loop, so theoretically Kd should be set 0 in only position mode and Kp should be set 0 in only speed mode.The control commands sent via CAN include: expected position Pdes, expected speed Vdes, feedforward torque ttf, and control parameters Kp and Kd. As can be seen from the code, these control commands are combined into a closed loop in the following control block diagram. θm refers to the mechanical angle, dθm refers to the mechanical angular velocity, ttf refers to the reference torque, KT is the torque constant, and iqrefl and idref represent the reference currents in the q-axis and d-axis respectively, as inputs to the FOC control. It can be seen that such a control method is flexible, which can be either only position control, only speed control, only torque control, or position plus torque feedforward, speed plus torque feedforward, etc. to control the motor.



3.5 CAN COMMUNICATION TRANSCEIVER CODE ROUTINE

SEND ROUTINE CODE

Note* The floating point type defined in the following functions should be set according to the motor model, and the parameters are for reference only.Please set them according to the table in "3.3 Parameter Range" above.

```
void pack_cmd(CANMessage * msg, float p_des, float v_des, float kp, float kd, float t_ff){
/// limit data to be within bounds ///

float P_MIN =-95.5;
float P_MAX =95.5;
float V_MIN =-30;
float V_MAX =30;
float T_MIN =-18;
float T_MAX =18;
float Kp_MIN =0;
float Kp_MAX =500;
float Kd_MIN =0;
float Kd_MAX =5;
float Test_Pos=0.0;

p_des = fminf(fmaxf(P_MIN, p_des), P_MAX);
v_des = fminf(fmaxf(V_MIN, v_des), V_MAX);
kp = fminf(fmaxf(Kp_MIN, kp), Kp_MAX);
kd = fminf(fmaxf(Kd_MIN, kd), Kd_MAX);
t_ff = fminf(fmaxf(T_MIN, t_ff), T_MAX);

/// convert floats to unsigned ints ///

int p_int = float_to_uint(p_des, P_MIN, P_MAX, 16);
int v_int = float_to_uint(v_des, V_MIN, V_MAX, 12);
int kp_int = float_to_uint(kp, KP_MIN, KP_MAX, 12);
int kd_int = float_to_uint(kd, KD_MIN, KD_MAX, 12);
int t_int = float_to_uint(t_ff, T_MIN, T_MAX, 12);
```

```
/// pack ints into the can buffer ///


msg->data[0] = p_int>>8;          //position 8-H
msg->data[1] = p_int&0xFF;            //position 8-L
msg->data[2] = v_int>>4;              //speed 8-H
msg->data[3] = ((v_int&0xF)<<4)|(kp_int>>8);    //speed 4-L  KP-8H
msg->data[4] = kp_int&0xFF;           //KP 8-L
msg->data[5] = kd_int>>4;             //Kd 8-H
msg->data[6] = ((kd_int&0xF)<<4)|(t_int>>8);    //KP 4-L torque 4-H
msg->data[7] = t_int&0xff;                  //torque 8-L
}
```

PROGRAM SCREENSHOT OF THE ROUTINE:

```
90 void pack_cmd(uint8_t *msg, float p_des, float v_des, float kp, float kd, float t_ff)
91 {
92     /// limit data to be within bounds ///
93     float P_MIN =-95.5;
94     float P_MAX =95.5;
95     float V_MIN =-30;
96     float V_MAX =30;
97     float T_MIN =-18;
98     float T_MAX =18;
99     float Kp_MIN =0;
100    float Kp_MAX =500;
101    float Kd_MIN =0;
102    float Kd_MAX =5;
103    float Test_Pos=0.0;
104
105
106    p_des = fminf(fmaxf(P_MIN, p_des), P_MAX);
107    v_des = fminf(fmaxf(V_MIN, v_des), V_MAX);
108    kp = fminf(fmaxf(Kp_MIN, kp), Kp_MAX);
109    kd = fminf(fmaxf(Kd_MIN, kd), Kd_MAX);
110    t_ff = fminf(fmaxf(T_MIN, t_ff), T_MAX);
111    /// convert floats to unsigned ints ///
112
113    int p_int = float_to_uint(p_des, P_MIN, P_MAX, 16);
114    int v_int = float_to_uint(v_des, V_MIN, V_MAX, 12);
115    int kp_int = float_to_uint(kp, Kp_MIN, Kp_MAX, 12);
116    int kd_int = float_to_uint(kd, Kd_MIN, Kd_MAX, 12);
117    int t_int = float_to_uint(t_ff, T_MIN, T_MAX, 12);
118
119    /// pack ints into the can buffer ///
120
121    msg[0] = p_int>>8; //位置高 8                                    (High position
122    /// pack ints into the can buffer ///
123
124    msg[0] = p_int>>8; //位置高 8                                    (High position
125    msg[1] = p_int&0xFF; //位置低 8                                  (Low position
126    msg[2] = v_int>>4; //速度高 8 位                                 (High speed 8)
127    msg[3] = ((v_int&0xF)<<4)|(kp_int>>8); //速度低 4 位 KP 高 4 位    (The speed is
128    msg[4] = kp_int&0xFF; //KP 低 8 位                               (KP Low 4 )
129    msg[5] = kd_int>>4; //Kd 高 8 位                                 (KP Higt 4 )
130    msg[6] = ((kd_int&0xF)<<4)|(t_int>>8); //KP 低 4 位扭矩高 4 位    (KP 4 lower to
131    msg[7] = t_int&0xff;  //扭矩低 8 位                              (Torque is 8 b
132
133
134 }
135
```

All the numbers are converted to integers by the following functions before they are sent to the motors.

```
int float_to_uint(float x, float x_min, float x_max, unsigned int bits)
{
```

```
/// Converts a float to an unsigned int, given range and number of bits ///
float span = x_max - x_min;
if(x < x_min) x = x_min;
else if(x > x_max) x = x_max;
return (int) ((x- x_min)*((float)((1<<bits)-1)/span));
}
```

PROGRAM SCREENSHOT OF THE ROUTINE:

```
133 int float_to_uint(float x, float x_min, float x_max, unsigned int bits)
134 {
135   /// Converts a float to an unsigned int, given range and number of bits ///
136   float span = x_max - x_min; //计算差值
137   /*****   判断是否小于最小值    ****/
138   if(x < x_min) //小于最小值，取最小
139     x = x_min;
140   else if(x > x_max)//大于最大值，取最大
141     x = x_max;
142   return (int) ((x- x_min)*((float)((1<<bits)-1)/span));
143 }
```

RECEIVING ROUTINE CODE

```
void unpack_reply(CANMessage msg){
/// unpack ints from can buffer ///
int id = msg.data[0]; //Driver ID number
int p_int = (msg.data[1]<<8)|msg.data[2];              //Motor position data
int v_int = (msg.data[3]<<4)|(msg.data[4]>>4);         //Motor speed data
int i_int = ((msg.data[4]&0xF)<<8)|msg.data[5];        //Motor torque data
/// convert ints to floats ///
float p = uint_to_float(p_int, P_MIN, P_MAX, 16);
float v = uint_to_float(v_int, V_MIN, V_MAX, 12);
float i = uint_to_float(i_int, -T_MAX, T_MAX, 12);
if(id == 1){
position = p;              //Read the corresponding data according to the ID number
speed = v;
torque = i;
}
```

PROGRAM SCREENSHOT OF THE ROUTINE:

```
138 void unpack_reply(uint8_t *data)
139 {
140   /// unpack ints from can buffer ///
141   int id = data[0]; //驱动 ID 号
142   int p_int = (data[1]<<8)|data[2];         //电机位置数据
143   int v_int = (data[3]<<4)|(data[4]>>4);    //电机速度数据
144   int i_int = ((data[4]&0xF)<<8)|data[5];   //电机扭矩数据
145   /// convert ints to floats ///
146   float p = uint_to_float(p_int, P_MIN, P_MAX, 16);
147   float v = uint_to_float(v_int, V_MIN, V_MAX, 12);
148   float i = uint_to_float(i_int, -T_MAX, T_MAX, 12);
149   if(id == 1)
150   {
151     postion = p;      //根据 ID 号读取对应数据
152     speed = v;
153     torque = i;
154   }
155 }
156
```

Position Global variables, location

Speed Global variables, speed

Torque Global variables, torque

All the numbers are converted to floating points by the following functions during packet collection.

float uint_to_float(int x_int, float x_min, float x_max, int bits)

{

/// converts unsigned int to float, given range and number of bits ///

float span = x_max - x_min;

float offset = x_min;

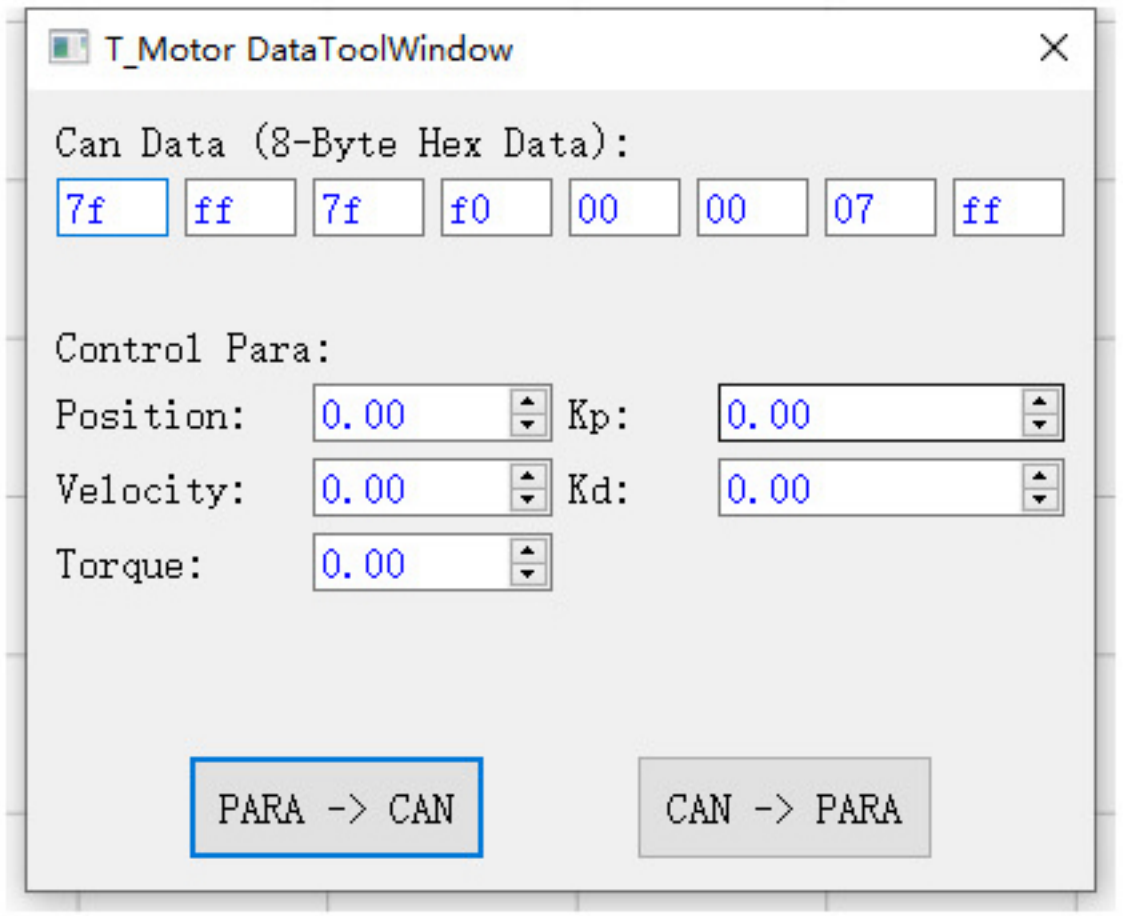
return ((float)x_int)*span/((float)((1<<bits)-1)) + offset;

}

PROGRAM SCREENSHOT OF THE ROUTINE:

```
30 float uint_to_float(int x_int, float x_min, float x_max, int bits)
31 {
32   /// converts unsigned int to float, given range and number of bits ///
33   float span = x_max - x_min;
34   float offset = x_min;
35   return ((float)x_int)*span/((float)((1<<bits)-1)) + offset;
36 }
37
```

SCREENSHOT OF THE UPPER COMPUTER'S PROGRAM:



SCREENSHOT OF CAN SENDING MESSAGE DATA:

序号	系统时间	时间标识	CAN通道	传输方向	帧ID	帧类型	帧格式	长度	数据
01929	17:36:28.598	0xF5AF10	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01930	17:36:28.718	0xF5B2EC	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01931	17:36:28.809	0xF5B6C9	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01932	17:36:28.898	0xF5BAA6	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01933	17:36:29.018	0xF5BE82	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01934	17:36:29.107	0xF5C25F	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01935	17:36:29.198	0xF5C63C	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01936	17:36:29.319	0xF5CA18	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01937	17:36:29.408	0xF5CDF5	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01938	17:36:29.498	0xF5D1D1	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01939	17:36:29.588	0xF5D5AE	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01940	17:36:29.708	0xF5D98B	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01941	17:36:29.799	0xF5DD67	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01942	17:36:29.888	0xF5E144	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF
01943	17:36:30.009	0xF5E520	ch1	接收	0x0200	数据帧	标准帧	0x08	x| 7F FF 7F F0 00 00 07 FF

Note* If the message set by the upper computer does not match with the received message, please set and modify some basic parameters of the upper computer.

3.6 CONFIGURATION SOFTWARE

Use the USB to serial tool to connect the motor to the computer and set the parameters for the motor.

1、Using the matching signal cable, connect the motor and USB to serial tool, 2p interface for CAN, 3p interface for serial. Then connect the USB to serial tool to the computer.

2、Turn on the power supply for the motor, do not cut off the power or connection before setting is completed.

3、Run the configuration software. After selecting the serial port in the software interface, click CONNECT. After selecting the motor, click ENTER_MA_MODE to enter the motor mode for debugging.

Actuator	AK60-6	AK80-6	AK80-9	AK10-9		AK70-10		AK80-80	
Maximum no-load speed rpm	400	365	485	24V	48V	24V	48V	24V	48V
				222	445	200	400	28	57
No-load current A	1								
Rated torque Nm	3	6	9	18		8.3		48	
Positioning accuracy bit	12								
Rated current A	7.4	12	12	22		8.8		13	
Maximum efficiency	80%								
Stall torque 14Nm	9	12	18	54		24.8		144	
Stall current 24A	22	24	24	68		26.1		40	
Number of Pole-Pair 21	14	21							

Note: Due to different programming environments, the above routines can be compared to the protocol if there are errors, and if there are differences, everything is based on the protocol.

The above example environment is keil5 version 5.33, and the library used is the hal library STM32Cube FW_F4 V1.25.2.

4. Package List