免责声明

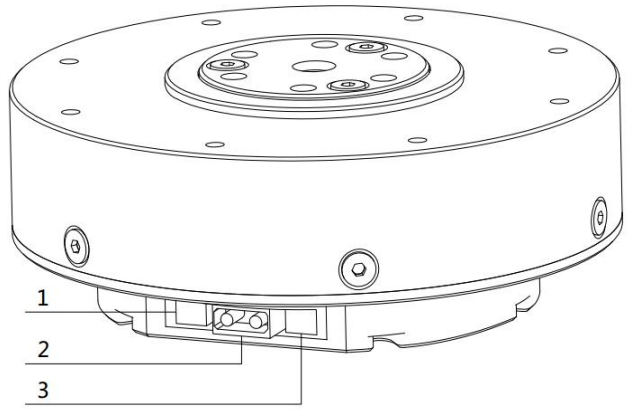
使用注意事项

1.请严格按照本文档规定的工作环境以及温度范围使用电机。
2.避免异物进入电机内部，导致转子运行异常。
3.使用前确保接线正常、稳固。
4.用户请勿私自拆卸电机，否则会影响电机的控制精度，甚至导致电机运行异常。

简介

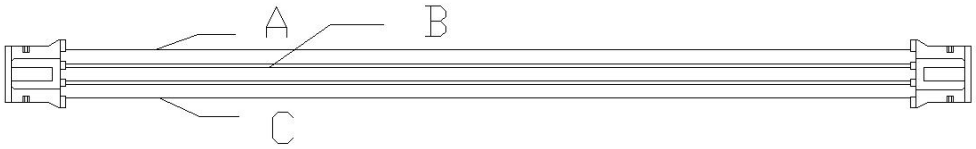
AK80-6 电机是一款集成减速组、驱动器高可靠性的无刷直流电机。可广泛应用于机械臂，自动化设备，科研教育等领域。驱动器采用磁场定向控制（FOC）算法，配合高精度的角度传感器，能实现精准的位置、力矩控制。多槽极数的设计、稀土材料磁铁和高精度的减速组为电机能够输出更稳定、更大的扭矩。AK80-6 电机支持多种通信协议，通过与 PC 通信，提供人机交互界面，使用户更快、更精准的控制电机。

接口与线序说明



1.串口信号端口

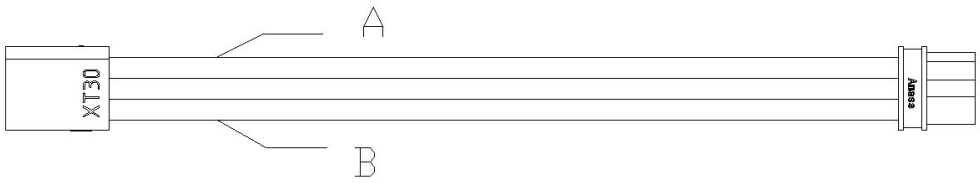
通过 USART 连接 PC 端，设置 AK80-6 电机的零点位置，校准编码器，设置驱动器 ID 号，最大电流等参数。



线序：从上到下分别是 A(GND)、B(RX)、C(TX)。线长 200mm。

2.电源接口

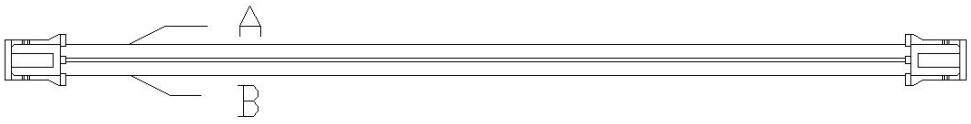
通过 XT30 接头连接电源（额定电压 24V）为 AK80-6 电机供电。



线序：从上到下分别是 A(电源负极)、B(电源正极)。

3.can 信号接口

外部设备可以通过 can 信号线发送控制指令，反馈电机状态信息。can 总线比特率为 1Mbps，主机 ID 号默认为 0x00。



线序：从上到下分别是 A（CAN_H）、B(CAN_L)。线长：200mm。

CAN 通信协议

电机接收报文格式。

用于向电机发送控制指令，控制电机的位置，转速，电流。

特殊 can 代码

进入电机控制模式 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,0XFC }

退出电机控制模式 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0XFD }

设置电机当前位置为零点 {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0XFE }

注意：使用 CAN 通信控制电机时必须先进入电机控制模式！

标识符： 设置的电机 ID 号（默认为 1） 帧类型：标准帧

帧格式： DATA DLC: 8 字节

数据域	DATA[0]	DATA[1]	DATA[2]	DATA[3]	
数据位	7-0	7-0	7-0	7-4	3-0
数据内容	电机位置高 8 位	电机位置低 8 位	电机速度高 8 位	电机速度低 4 位	KP 值高 4 位

数据域	DATA[4]	DATA[5]	DATA[6]		DATA[7]
数据位	7-0	7-0	7-4	3-0	0-7
数据内容	KP 值低 8 位	KD 值高 8 位	KD 值低 4 位	电流值高 4 位	电流值低 8 位

电机发送报文格式

标识符： 0X00+驱动器 ID 帧类型：标准帧

帧格式： DATA DLC: 8 字节

数据域	DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[4]
数据位	7-0	7-0	7-0	7-0	7-4
数据内容	驱动器 ID 号	电机位置高 8 位	电机速度低 8 位	电机速度高 8 位	电机速度低 4 位

数据域	DATA[4]	DATA[5]
数据位	3-0	7-0
数据内容	电流值高 4 位	电流值低 8 位

发送频率：1 MHZ

参数范围

电机位置：-95.5f ~ 95.5f 单位 rad

电机速度：-30 ~ 30 单位 rad/S

电机扭矩：-18 ~ 18 单位 Nm

Kp 的范围：0~500

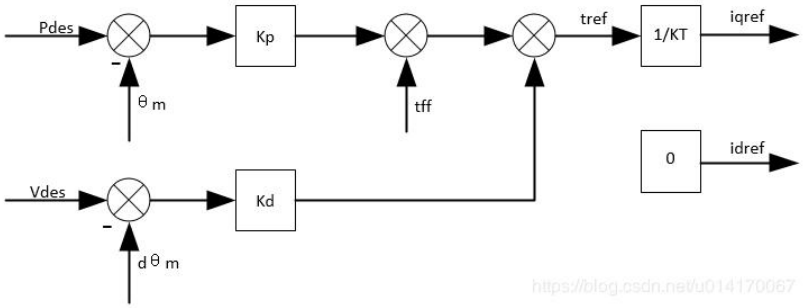
Kd 的范围：0~5

内部控制器 pid 示意图

若想实现纯位置、纯速度、纯转矩控制，仅需给对应的变量赋值其余量赋 0 即可。比如若想实现位置控制，则在发包的时候将电机位置赋值，转矩和速度则发 0。

Kp 控制的是位置环的参数，Kd 控制的是速度环的参数，所以理论来说在纯位置模式下 Kd 应该赋 0.纯速度模式下 Kp 应该赋 0.

通过CAN输入的控制命令包含了：期望位置Pdes、期望速度Vdes、前馈转矩tff、控制参数Kp、Kd。由代码可知，这些控制命令按如下控制框图的方式组合成闭环。其中，θm表示机械角度、dθm表示机械角速度、tref表示参考转矩、KT是转矩常数、iqref和idref分别表示q轴和d轴的参考电流，作为FOC控制的输入。
可以看出，这样的控制方式很灵活，既可以是纯位置控制、纯速度控制、纯转矩控制，也可以位置加转矩前馈、速度加转矩前馈等方式控制电机。



CAN 通信收发代码例程

发送例程代码

```
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;                    //位置高 8
        msg->data[1] = p_int&0xFF;                  //位置低 8
msg->data[2] = v_int>>4;                          //速度高 8 位
        msg->data[3] = ((v_int&0xF)<<4)|(kp_int>>8);  //速度低 4 位     KP 高 4 位
        msg->data[4] = kp_int&0xFF;                 //KP 低 8 位
        msg->data[5] = kd_int>>4;                   //Kd 高 8 位
        msg->data[6] = ((kd_int&0xF)<<4)|(t_int>>8);  //KP 低 4 位扭矩高 4 位
        msg->data[7] = t_int&0xff;                  //扭矩低 8 位
        }
```

发包时所有的数都要经以下函数转化成整型数之后再发给电机。

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

接收例程代码

```
void unpack_reply(CANMessage msg){
    /// unpack ints from can buffer ///
    int id = msg.data[0];                                   //驱动 ID 号
    int p_int = (msg.data[1]<<8)|msg.data[2];               //电机位置数据
    int v_int = (msg.data[3]<<4)|(msg.data[4]>>4);          //电机速度数据
    int i_int = ((msg.data[4]&0xF)<<8)|msg.data[5];         //电机扭矩数据
    /// 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, -I_MAX, I_MAX, 12);
            if(id == 1){
postion = p;                                        //根据 ID 号读取对应数据
speed = v;
torque   = i;
          }
   }
```

收包时所有的数都要经以下函数转化成浮点型。

```
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;
}
```

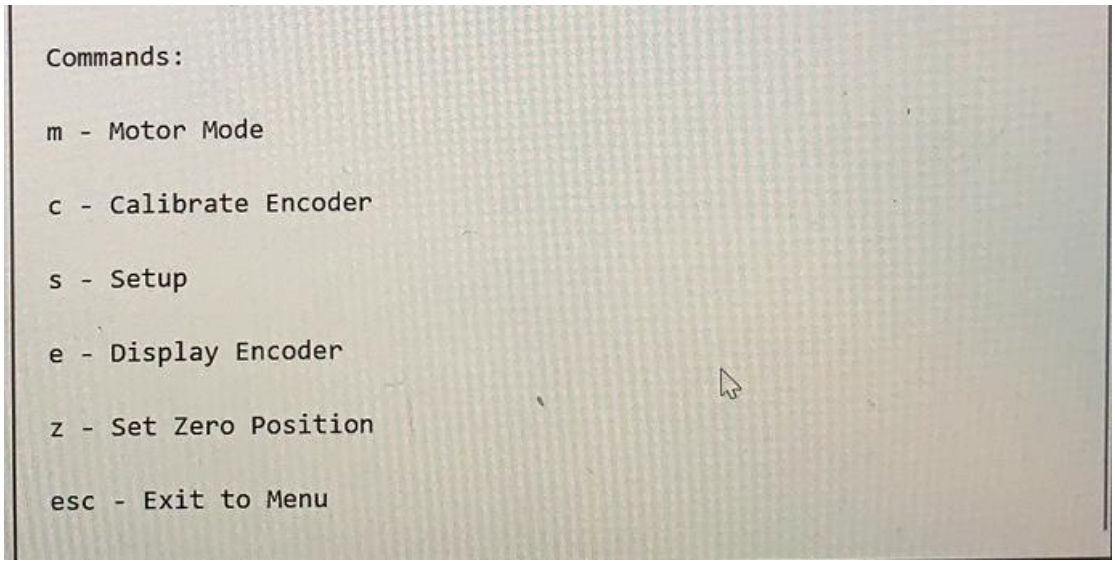
调参软件

使用 USB 转串口工具，将电机连接至计算机，对电机进行参数设置。

1、使用配套信号线，连接电机和 USB 转串口工具，2p 接口为 CAN，3p 接口为串口信号。然后将 USB 转串口工具接入计算机。

2、接通电源为电机供电，设置完成前请勿切断电源或连接。

3、运行调参软件（此处以 sscom 串口调试工具为例）。波特率为 921600。在电机上电时会向连接的串口发数。显示出以下文字：



4、m 即进入电机模式，此时电机中的红灯会变为绿灯。

5、c 即标定编码器，发送 c 时电机会开始转动，并向串口不断发送当前位置信息。

6、s 即为设置界面里面，内部可调整控制器 can id。主机 can id 等一系列参数。在发送数据时一定要将 sscom 内“加回车换行”勾选。

之后在按照内部提示调整即可。如希望将 can id 号改成 2 则需发送 i2+回车键，如果此时没有修改成功则需一次性多发几个 i2.

7、z 即设置当前位置为电机编码器零点。

特征参数

额定电压下的电机参数/24V

一般流程：
1.先不供电源打开上位机，选择串口，连接。
2.到configuration界面，此时供电会显示一串字符，而后输入c整定，输入s设置CAN ID（这个数据将会长期存储于驱动器中，就像大疆一样），输入m检查点击状态是否良好。
3.结束，该上位机只起到整定和改变CAN ID的作用，其他一概不用处理，直接跑程序。

最大空载转速	427rpm
空载电流	1A
额定扭矩	4.8Nm
定位精度	12bit
额定电流	8.1A
最大效率	80%
堵转扭矩	14Nm
堵转电流	24A
电机极对数	21