驱动器用户手册

Version: V1.0

Data: 2020/5/8

修订历史

版本	日期	内容
V1.0	2010/5/8	初版

目录

一.使用前准备

1.1 使用注意事项

1) 接线过程中，**要求**断开电源连接。**严禁**带电拔插所有接口。

2) **请勿**超出电压范围使用，额定定压：24V，工作电压范围：7-33V。

3) 接线时，**注意**分辨接口功能和兼容的逻辑电平，上电前**检查**接线是否正确。（定义见章节 2.3 接口说明）。未按要求标准接线可能会烧毁接口，导致驱动无法正常工作，不作保修。

4) CAN 接口默认驱动器是不带 120Ω终端电阻，使用时**注意**在总线的两个终端上接上 120Ω电阻。可以要求发货焊接好电阻。建议 CAN 总线使用双绞线并使用 GND 包围屏蔽，可以提高稳定性，减少干扰。

5) 编码器 CHA、CHB 两个通道电平为兼容 3.3V/5V，**请勿**接入过高电压；注意编码器引线尽量不要过长，小于 20cm 为最佳。

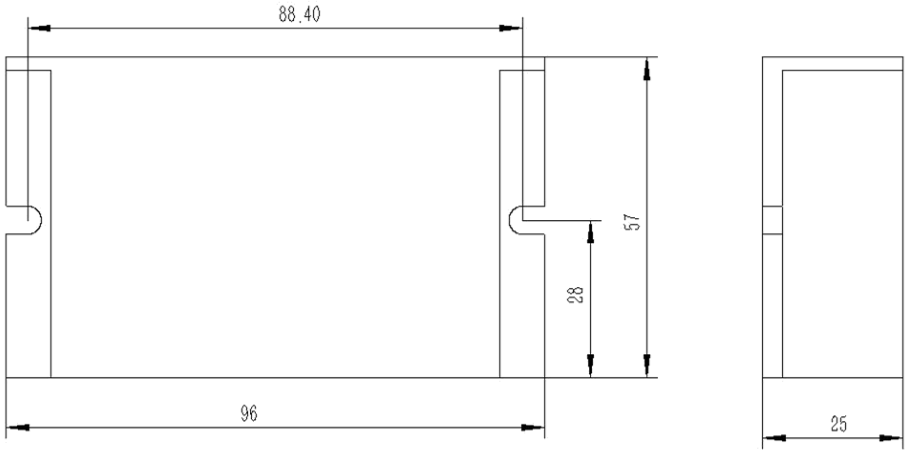
6) 串口电平为 RS232 电平，电平范围为±15V，非 TTL 电平，TTL 电平范围为 0~3.3/5V，两者不兼容。

7) 驱动器供电**要求**用电池直接供电，**禁止**使用小型 DC-DC 模块。因为电机在加减速的时候会大的电流波动，产生浪涌电压，使用电池能提供足够的电流，同时可以吸收掉浪涌电压。

二.产品性能参数

2.1 产品尺寸

项目	参数
外壳材料	铝合金
外形尺寸	96mm×57mm×25mm
重量	约 120g



2.2 技术参数

项目	参数	备注
工作电压	DC 7-33V	额定 24V
额定电流	10A	
峰值电流	15A	
编码器电压	5V	
控制输入电压	5V	

使用单位：

转向定义：默认正视电机转轴，逆时针转动为正转，顺时针转动为反转。

速度的单位为：RPM。（RPM 为转每分钟的意思，国际通用单位）。

电流的单位为：mA。（mA 为毫安， 1A = 1000mA）

位置的单位为：qc。（qc 意思是四倍频线数，比如编码器精度是 500 线每圈，那么一圈就是 2000qc， 1qc 代表角度 360° /2000 = 0.18° ）

2.3 接口定义

接口		说明	备注
电源接口	+24	电源正极输入	范围 DC 12-40V
	GND	电源负极输入	
电机接口	M1	电机输出接口 1	
	M2	电机输出接口 2	
编码器接口	5V	编码器 5V 电源	
	GND	编码器电源地	
	A	编码器输出 A 相	
	B	编码器输出 B 相	
控制接口	D/CTL2	复用控制接口	
	A/CTL1	复用控制接口	
	CAN H	CAN 通讯接口	
	CAN L	CAN 通讯接口	

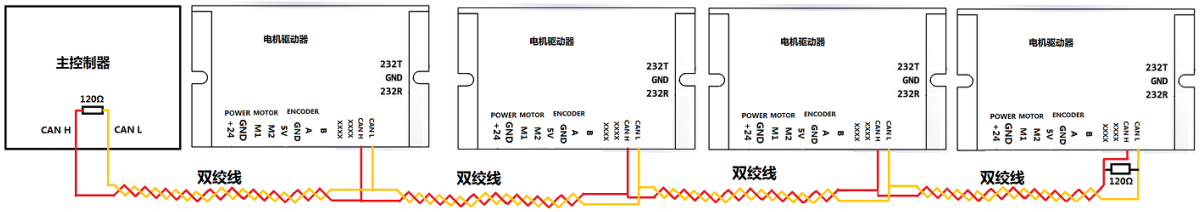
串口接口：串口电平为 RS232 电平，电平范围为±15V（非 TTL 电平，电平范围为 0~3.3/5V，两者不兼容）

编码器接口：编码器 CHA、CHB 两个通道电平为兼容 3.3V/5V（注意编码器引线尽量不要过长，小于 20cm 为最佳）。

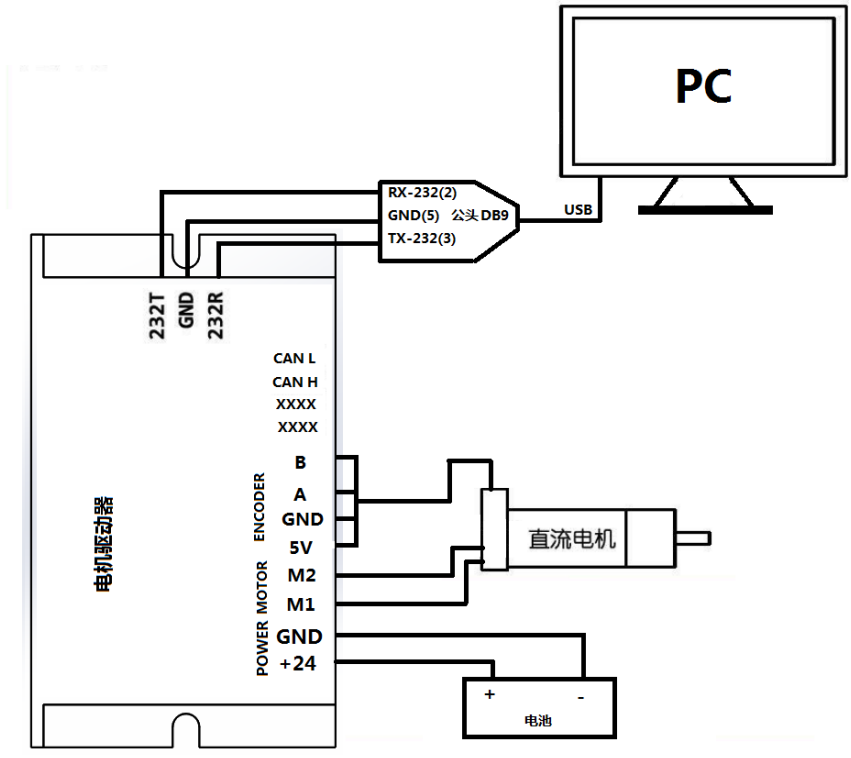
CAN 接口：CAN 为标准 CAN 总线接口。

2.4 接线说明

使用 CAN 通讯进行控制时，接线方式如下。使用双绞线连接所有 CAN H，CAN L；驱动器并联在一条总线上。并在首尾端分别接入 120Ω电阻，以提高通讯稳定性。



使用串口通讯时，连接驱动器右侧的串口接口（注意串口电平为 RS232 电平，非 TTL 电平）。一般 TX 接其他设备的 RX 接口，RX 接其他设备的 TX 接口。接法如下，与其通讯的设备是可以其他串口设备或者 PC。



三.使用说明

3.1 快速入门

- 连接电机和编码器

  a.将电机线连接到驱动器电机接口（MOTOR）M1，M2，无强制区分，一般红接 M1，黑接 M2，保持同种电机一致接线顺序即可，后续可以通过试运行调整。

  b.编码器按定义连接到驱动器编码器接口（ENCODER）的 5V，GND，A，B。

- 连接通讯线

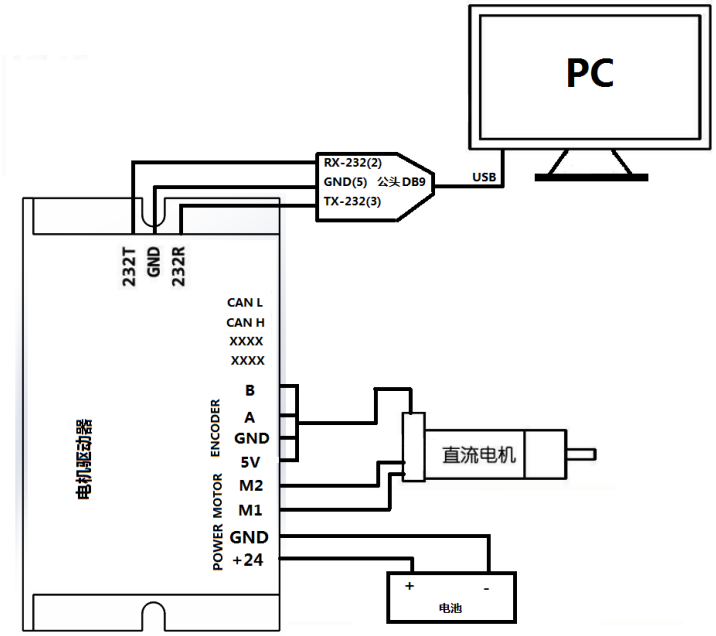
  将 USB 转串口线连接到电脑。安装 USB 转串口线驱动。

  将串口线连接到驱动器串口接口（RS232）。

- 连接电源线

按正负定义连接电源线。

总体连接方式如下：

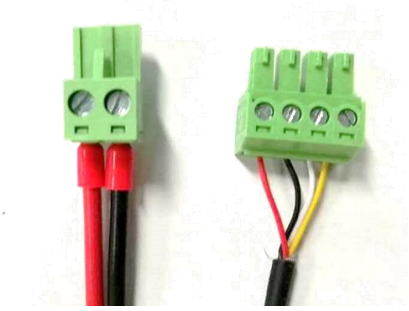


若使用推荐电机，则接线如下：

推荐电机编码器接口定义如下：

红色	黑色	白色	黄色
+5V	GND 地	A 相	B 相

电机线与编码器线接法如下：

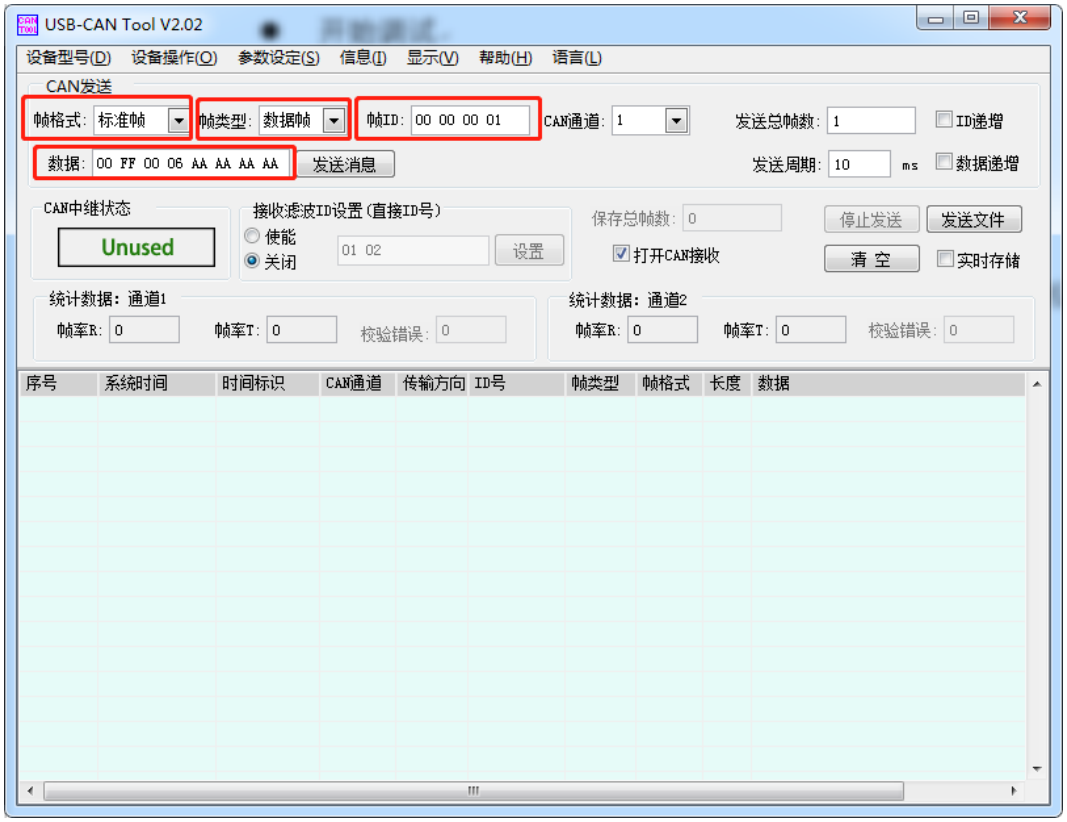


- CAN 快速测试

使用 CAN 分析仪/工具快速测试，通过 CAN 接口进行电机控制。

CAN 控制的启动流程参考 4.1 章。

以 USB-CAN Tool 工具为例，如下图：



设置，帧格式：标准帧；帧类型：数据帧；按下发内容设置帧 ID，数据。

1） 复位一号驱动器，等待驱动器上线。

CAN_ID：0x01　　数据：0xAA，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

2） 切换控制模式，切换为开环模式 01。

CAN_ID：0x11　　数据：0x01，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

3） 下发开环控制指令，20%的 PWM 占空比，让电机正转。

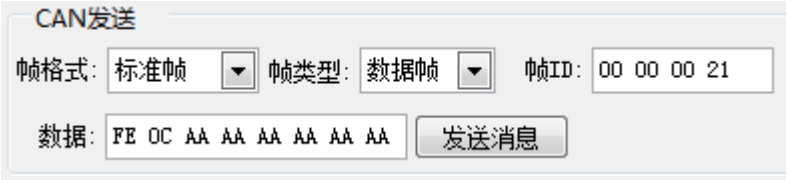
CAN_ID：0x21　　数据：0x03，0xE8，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

4） 下发开环控制指令，让电机停止。

CAN_ID：0x21　　数据：0x00，0x00，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

5） 下发开环控制指令，10%的 PWM 占空比，让电机反转。

CAN_ID：0x21　　数据：0xFE，0x0C，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA



6） 下发开环控制指令，让电机停止。

CAN_ID：0x21　　数据：0x00，0x00，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

3.2 工作模式

本驱动器可以一共支持 8 种工作模式：开环模式、力矩模式（电流模式）、速度模式、位置模式、速度位置模式、电流速度模式、电流位置模式、电流速度位置模式。

a) 开环模式

开环模式用于控制无编码器反馈电机，通过给定 PWM 的值（取值范围：-5000~+5000，正负号代表方向）来控制电机输出。

例如： 给定 PWM 值为+5000 时，电机正转，100%输出。

给定 PWM 值为 0 时，停止输出。

给定 PWM 值为-2500 时，电机反转，50%输出。

开环模式还可以用于试运行，测试编码器和电机线方向是否对应；测量电机最高转速。测试编码器和电机线方向是否对应：给定一个正数 PWM 值，电机正转，编码器反馈的数值为正数；给定一个负数 PWM 值，电机反转，编码器反馈的数值为负数；则方向正确，两者对应。反之，给定正数 PWM 值，编码器反馈为负值，则方向不对应，需要在“编码器设置”中将编码器方向取反，或者将两根电机线对调（建议：按默认转向定义为前提，确定电机线接线，然后通过调整编码器设置来标定编码器和电机线方向。）

转向定义：默认正视电机转轴，逆时针转动为正转，顺时针转动为反转

测量电机最高转速：给定 PWM 值为+5000，此时反馈的转速为该电机的最高转速。

b）力矩模式（电流模式）

让电机以一个恒定的电流工作，相当于电机输出一个恒定的力矩。

控制量：目标电流，单位：mA

PWM 限幅，单位：占空比

c）速度模式

让电机以一个恒定的速度工作，电机的转速不因负载的变化而变化。

该模式为常用的工作模式，一般应用于对速度有要求的场景，例如小车的行进速度等。可以保证小车在上坡下坡的时候依然保持原有的速度。

控制量：目标转速，单位：RPM

d）位置模式

给定一个目标位置值（相当于需要转过多少个脉冲），电机以最快速度转到给定位置并停下来，同时锁定在该位置上。

控制量：目标位置，单位：qc

e）速度位置模式

给定一个限定最高转速和目标位置值，电机在限定转速内以最快速度到达给定的目标位置，停下来并锁定。

控制量：目标位置，单位：qc

限定转速，单位：RPM

f）电流速度模式

让电机以一个恒定转速工作，同时运动过程中保持电流小于给定值。

该模式可以让电机在工作过程中，输出电流（力矩）小于限定值，起保护电机或者外部机构的作用。

通过限定电流，也可以有防止堵转的作用。

控制量：目标转速，单位：RPM

限定电流，单位：mA

g）电流位置模式

给定一个目标位置值（相当于需要转过多少个脉冲），电机以最快速度转到给定位置并

停下来，同时整个过程中限制电流的大小。

控制量：目标位置，单位：qc

限制电流，单位：mA

h）电流速度位置模式

给定一个目标位置值，电机在限定转速和限定电流内以最快速度到达给定的目标位置，停下来并锁定。

控制量：目标位置，单位：qc

限定速度，单位：RPM

限制电流，单位：mA

四.控制接口

4.1 CAN 通讯说明

驱动器的 CAN 总线支持不同波特率（出厂默认 1000Kbps），可以通过软件进行设置。CAN 波特率以下数值：1000kbps，500kbps，250kbps，125kbps 等。

驱动器编号出厂默认为 1，也可以通过调试软件进行修改，驱动器编号范围为 1-15（对应 16 进制 1-0xF）。

所有 CAN 消息均为标准帧，数据帧，帧长度为 8 位，这些不可修改。

CAN_ID 遵循以下规则，

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	范围：0~F	范围：1~F

标准帧 ID 长度为 11 位，十六进制表示，以 0x 开头，最高 3 位保留功能，固定为 0，中间 4 位为功能位，代表不同功能，最后 4 位为编号位，表示不同驱动器的标号。

功能序号定义如下：

0：驱动器复位指令；

1：工作模式切换指令；

2：开环模式下，驱动器控制指令；

3：闭环模式下，驱动器控制指令；

4~9：保留；

A：配置指令；

B：驱动器电流、 速度、 位置反馈；

C：驱动器 CTL1、 CTL2 、DIN、AIN 反馈；

D：保留；

E：错误状态反馈；

F：上线上报/在线检测；

使用 CAN 通讯进行控制时，流程如下：

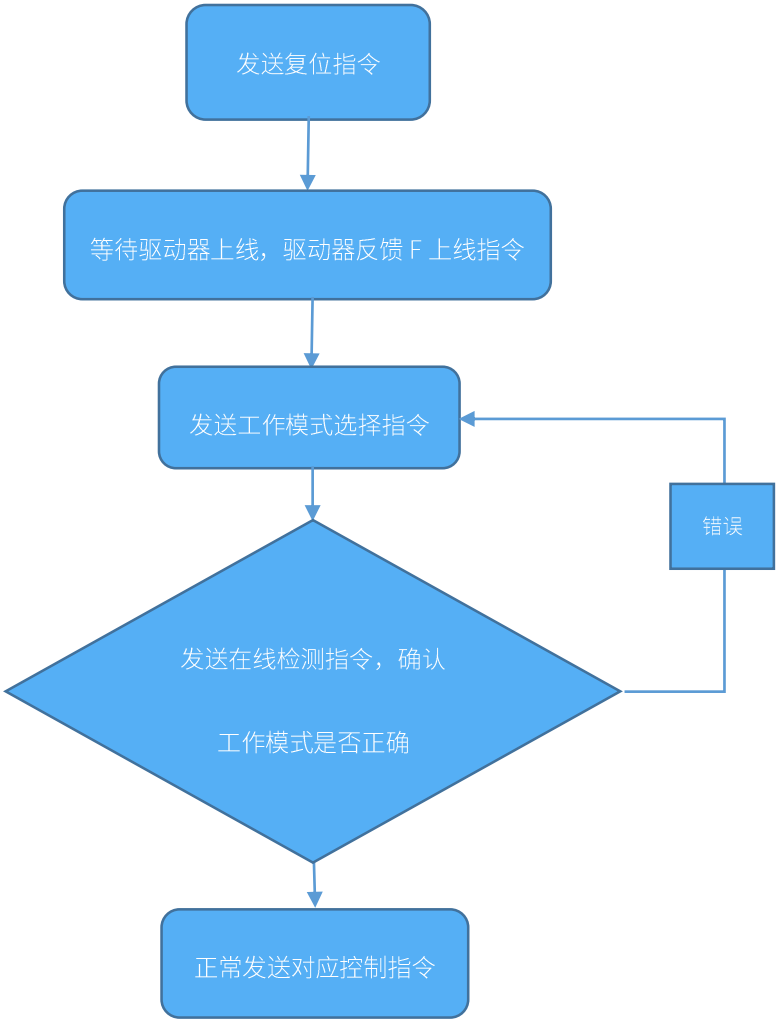
1．发送复位指令；

2．等待驱动器上线反馈；

3．发送工作模式选择指令，使驱动器进入目标模式；

4．等待 500ms，发送在线检测指令，确认驱动器正常进入目标工作模式；

5. 在已经进入的模式下，发送控制指令，正常使用。



4.2 CAN 通讯协议说明

0：驱动器复位指令

该指令 CAN_ID：

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	0	范围：1~F

主控发送指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA

说明：该指令在任何状态下都会被响应，响应后，驱动器复位，相当于断电重启。复位后，需要重新切换工作模式。

使用示例：

希望一号驱动器复位重启，则下发以下指令：

CAN_ID=0x001，标准帧,，数据帧，长度为 8，

内容为：0xAA，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

希望三号驱动器复位重启，则下发以下指令：

CAN_ID=0x003，标准帧,，数据帧，

内容为：0xAA，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

1：工作模式切换指令

该指令 CAN_ID：

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	1	范围：1~F

主控发送指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	待选择	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA

说明：重启上电后，驱动器工作模式为 0（待机模式），该指令只有工作模式为 0 时才会被响应。Data[0]的值对应不同的工作模式，如下：

工作模式	Data[0]的值
开环模式	0x01
电流模式	0x02
速度模式	0x03
位置模式	0x04
电流速度模式	0x05
电流位置模式	0x06
速度位置模式	0x07
电流速度位置模式	0x08

使用示例：

希望一号驱动器进入速度模式，则下发以下指令：

CAN_ID=0x011，标准帧,，数据帧，长度为 8，

内容为：0x03，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

2：开环模式下，驱动器控制指令

该指令 CAN_ID：

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	2	范围：1~F

主控发送指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	待计算	待计算	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA

说明：　该指令只有工作模式为 1（开环模式）时才会被响应。

控制量为 PWM 占空比，取值范围：-5000~+5000，正负号代表方向（正反转），

0~5000 对应 0~100%PWM 占空比。

数据类型为：signed int16；Data[0]为高八位，Data[1]为低八位。

使用示例：

希望一号驱动器在开环模式下，以 60%的 PWM 控制正转，则下发以下指令：

CAN_ID=0x021，标准帧,，数据帧，长度为 8，

内容为：0x0B，0xB8，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

STM32 C 语言示例，

```
signed int16 temp_pwm = 3000;                    //定义控制变量
TxMessage.StdId = 0x021;
TxMessage.IDE = CAN_ID_STD;                      //标准帧
TxMessage.RTR = CAN_RTR_DATA;                    //数据帧
TxMessage.DLC = 8;                               //数据长度为 8 字节

TxMessage.Data[0] = (unsigned char)(temp_pwm>>8);
TxMessage.Data[1] = (unsigned char) temp_pwm;
TxMessage.Data[2] = 0xAA;
TxMessage.Data[3] = 0xAA;
```

其他数据位也均为 0xAA。

3：闭环模式下，驱动器控制指令

该指令 CAN_ID：

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	3	范围：1~F

主控发送指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	待计算	待计算	待计算	待计算	待计算	待计算	待计算	待计算

说明：　该指令在工作模式 2~8 时会被响应。在不同模式下，分别响应不同位的控制量。

在模式 3（速度模式）下，驱动器只响应该指令的 Data[2]和 Data[3]数据位。

在模式 7（速度位置模式）下，驱动器只响应该指令的 Data[2]~ Data[7]数据位。

其他模式同理。数据位对应的控制量如下：

数据位	控制量	数据类型	说明
Data[0]	目标电流	signed int16	单位：mA 取值范围：-32768~+32767 正负号代表方向（正反转）
Data[1]			
Data[2]	目标转速	signed int16	单位：rpm 取值范围：-32768~+32767 正负号代表方向（正反转）
Data[3]			
Data[4]	目标位置	signed int32	单位：qc 取值范围：int32 数据范围 正负号代表方向（正反转）
Data[5]			
Data[6]			

Data[7]			

所有数据发送时，均为高位在前，低位在后。

使用示例：

希望一号驱动器在速度模式下，以 200RPM 正转，则下发以下指令：

CAN_ID=0x031，标准帧,，数据帧，长度为 8，

内容为：0xAA，0xAA，0x00，0xC8，0xAA，0xAA，0xAA，0xAA

STM32 C 语言示例，

```
signed int16 temp_rpm = 200;                    //定义控制变量
TxMessage.StdId = 0x031;
TxMessage.IDE = CAN_ID_STD;                     //标准帧
TxMessage.RTR = CAN_RTR_DATA;                   //数据帧
TxMessage.DLC = 8;                              //数据长度为 8 字节

TxMessage.Data[0] = 0xAA;
TxMessage.Data[1] = 0xAA;
TxMessage.Data[2] = (unsigned char)(temp_rpm>>8);
TxMessage.Data[3] = (unsigned char) temp_rpm;
TxMessage.Data[4] = 0xAA;
TxMessage.Data[5] = 0xAA;
TxMessage.Data[6] = 0xAA;
TxMessage.Data[7] = 0xAA;
```

希望三号驱动器在电流速度位置模式下，进行控制，则下发以下指令：

STM32 C 语言示例，

```
signed int16 temp_current;                      //定义电流控制变量
signed int16 temp_rpm;                          //定义速度控制变量
signed int16 temp_position;                   //定义位置控制变量
TxMessage.StdId = 0x033;
TxMessage.IDE = CAN_ID_STD;                     //标准帧
TxMessage.RTR = CAN_RTR_DATA;                   //数据帧
TxMessage.DLC = 8;                              //数据长度为 8 字节

TxMessage.Data[0] = (unsigned char)(temp_current >>8);
TxMessage.Data[1] = (unsigned char)temp_current;
TxMessage.Data[2] = (unsigned char)(temp_rpm >>8);
TxMessage.Data[3] = (unsigned char)temp_rpm;
```

```
TxMessage.Data[4] = (unsigned char)(temp_position >>24);
TxMessage.Data[5] = (unsigned char)(temp_position >>16);
TxMessage.Data[6] = (unsigned char)(temp_position >>8);
TxMessage.Data[7] = (unsigned char)temp_position;
```

4~9：保留；

A：配置指令

该指令 CAN_ID：

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	A	范围：1~F

主控发送指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待选择	待计算	待计算	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA

说明：　该指令在任何状态下都会被响应。

用于开启/关闭驱动器反馈。

数据位	数据类型	说明
Data[0]	unsigned int8	0：关闭反馈指令 B 1~255：反馈的周期，单位：ms
Data[1]	unsigned int8	0：关闭反馈指令 C 1~255：反馈的周期，单位：ms

使用示例：

希望一号驱动器以 10ms 定时反馈 B，关闭反馈 C，则下发以下指令：

CAN_ID=0x0A1，标准帧,，数据帧，长度为 8，

内容为：0x0A，0x00，0xAA，0xAA，0xAA，0xAA，0xAA，0xAA

B：驱动器电流、 速度、 位置反馈

该指令 CAN_ID：

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	B	范围：1~F

驱动器反馈指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待接收	待接收	待接收	待接收	待接收	待接收	待接收	待接收	待接收

说明： 该指令反馈电机的电流，转速，编码器位置。

具体数据位含义如下：

数据位	反馈值	数据类型	说明
Data[0]	当前电流	signed int16	单位：mA 正负号代表方向（正反转）
Data[1]			
Data[2]	当前转速	signed int16	单位：rpm 正负号代表方向（正反转）
Data[3]			
Data[4]	当前位置	signed int32	单位：qc 正负号代表方向（正反转）
Data[5]			
Data[6]			
Data[7]			

所有数据均高位在前，低位在后。

例如，一号驱动器周期往外反馈 current_now（当前电流），rpm_now（当前转速），position_now（当前位置），对外 CAN 发送如下：

```
TxMessage.StdId = 0x0B1;
```

```
TxMessage.IDE = CAN_ID_STD;                    //标准帧
TxMessage.RTR = CAN_RTR_DATA;                  //数据帧
TxMessage.DLC = 8;                             //数据长度为 8 字节

TxMessage.Data[0] = (unsigned char)(current_now >>8);
TxMessage.Data[1] = (unsigned char)current_now;
TxMessage.Data[2] = (unsigned char)(rpm_now >>8);
TxMessage.Data[3] = (unsigned char)rpm_now;
TxMessage.Data[4] = (unsigned char)(position_now >>24);
TxMessage.Data[5] = (unsigned char)(position_now >>16);
TxMessage.Data[6] = (unsigned char)(position_now >>8);
TxMessage.Data[7] = (unsigned char)position_now;
```

主控制器接收到该 CAN_ID 信息后，可以用以下方式进行接收处理。

```
signed int16 real_current = (Data[0]<<8)|Data[1];
signed int16 real_velocity = (Data[2]<<8)|Data[3];
signed int32 real_position = (Data[4]<<24)| (Data[5]<<16)| (Data[6]<<8)|
Data[7];
```

注意：电流和速度的数据类型必须为有符号 16 位整型，位置的数据类型必须为有符号 32 位整型。

C：驱动器 CTL1、 CTL2 、DIN、AIN 反馈

该指令 CAN_ID：

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	C	范围：1~F

驱动器反馈指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待接收	待接收	待接收	待接收	待接收	待接收	待接收	待接收	待接收

说明： 该指令反馈驱动 CTL1、CTL2、DIN、AIN、PWM 的状态。

具体数据位含义如下：

数据位	反馈值	数据类型	说明
Data[0]	CTL1	unsigned int8	状态反馈：0 或 1
Data[1]	CTL2	unsigned int8	状态反馈：0 或 1
Data[2]	DIN	unsigned int8	状态反馈：0 或 1
Data[3]	AIN	unsigned int16	电压值反馈 单位：mA
Data[4]			
Data[5]	PWM	signed int16	反馈范围：-5000~+5000
Data[6]			
Data[7]			

所有数据均高位在前，低位在后。

例如，一号驱动器周期往外反馈，对外 CAN 发送如下：

```
TxMessage.StdId = 0x0C1;
TxMessage.IDE = CAN_ID_STD;                    //标准帧
TxMessage.RTR = CAN_RTR_DATA;                  //数据帧
```

```
TxMessage.DLC = 8;                                    //数据长度为 8 字节

TxMessage.Data[0] = (unsigned char)CTL1;
TxMessage.Data[1] = (unsigned char)CTL2;
TxMessage.Data[2] = (unsigned char)DIN;
TxMessage.Data[3] = (unsigned char)(AIN >>8);
TxMessage.Data[4] = (unsigned char)AIN;
TxMessage.Data[5] = (unsigned char)(PWM >>8);
TxMessage.Data[6] = (unsigned char)(PWM);
TxMessage.Data[7] = 0xAA;
```

D：保留；

E：错误状态反馈

该指令 CAN_ID：

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	E	范围：1~F

驱动器反馈指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待接收	待接收	待接收	待接收	待接收	待接收	待接收	待接收	待接收

说明：　该指令反馈驱动错误状态。

正常运行情况下，该帧不会发送反馈，当发生错误的时候，驱动器以 100ms 周期对外发送该帧。不同数据位表示不同错误类型。具体数据位含义如下：

数据位	反馈值	数据类型	说明
Data[0]	OC	unsigned int8	过流状态反馈： 0（正常）1（过流）
Data[1]	EE	unsigned int8	编码器状态反馈： 0（正常）1（异常）
Data[2]	保留	保留	保留
Data[3]	保留	保留	保留
Data[4]	保留	保留	保留
Data[5]	保留	保留	保留
Data[6]	保留	保留	保留
Data[7]	保留	保留	保留

例如，一号驱动器周期发生错误往外反馈，对外 CAN 发送如下：

```
TxMessage.StdId = 0x0E1;
TxMessage.IDE = CAN_ID_STD;                    //标准帧
TxMessage.RTR = CAN_RTR_DATA;                  //数据帧
TxMessage.DLC = 8;                             //数据长度为 8 字节

TxMessage.Data[0] = (unsigned char)0C;
TxMessage.Data[1] = (unsigned char)EE;
TxMessage.Data[2] = 0xAA;
TxMessage.Data[3] = 0xAA;
TxMessage.Data[4] = 0xAA;
TxMessage.Data[5] = 0xAA;
TxMessage.Data[6] = 0xAA;
TxMessage.Data[7] = 0xAA;
```

F：上线上报/在线检测；

该指令 CAN_ID：

十六进制开头	保留	功能序号	驱动器编号
0x	0（默认固定为 0）	F	范围：1~F

主控发送指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待计算	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA

驱动反馈指令：

CAN_ID	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
待接收	待接收	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA	0xAA

说明：　该指令检测设备是否在线，查看当前工作模式。

驱动器每次上电启动成功后，会发送一帧反馈指令，作为上线报道。

主控发送对应 ID 指令后，驱动器会响应指令，反馈发送当前工作模式。

反馈指令，不同数据位表示不同含义。具体数据位含义如下：

数据位	反馈值	数据类型	说明
Data[0]	MODE	unsigned int8	工作模式
Data[1]~Data[7]	保留	保留	保留

例如，查看一号驱动器是否在线，其工作模式是什么，则主控下发此 CAN 信息：

```
TxMessage.StdId = 0x0F1;
TxMessage.IDE = CAN_ID_STD;                    //标准帧
TxMessage.RTR = CAN_RTR_DATA;                  //数据帧
TxMessage.DLC = 8;                             //数据长度为 8 字节
```

```
TxMessage.Data[0] = 0xAA;
TxMessage.Data[1] = 0xAA;
TxMessage.Data[2] = 0xAA;
TxMessage.Data[3] = 0xAA;
TxMessage.Data[4] = 0xAA;
TxMessage.Data[5] = 0xAA;
TxMessage.Data[6] = 0xAA;
TxMessage.Data[7] = 0xAA;
```

一号驱动器收到后，响应反馈当前工作模式（假设为模式 3 速度模式）：

```
TxMessage.StdId = 0x0F1;
TxMessage.IDE = CAN_ID_STD;                    //标准帧
TxMessage.RTR = CAN_RTR_DATA;                  //数据帧
TxMessage.DLC = 8;                             //数据长度为 8 字节

TxMessage.Data[0] = 0x03;
TxMessage.Data[1] = 0xAA;
TxMessage.Data[2] = 0xAA;
TxMessage.Data[3] = 0xAA;
TxMessage.Data[4] = 0xAA;
TxMessage.Data[5] = 0xAA;
TxMessage.Data[6] = 0xAA;
TxMessage.Data[7] = 0xAA;
```

五.免责声明

5.1 保修说明

从发货时刻起，非人为损坏情况下，保修一年。

如非驱动器本身质量问题，返厂保修，产生的来回运费由客户承担。

返修时，附带纸条说明损坏过程，联系人、联系电话及联系地址。

返修费用，以收到驱动器后评估。

以下情况，需要付费返修：

a) 超过保修期一年

b) 外力/人为导致的机械性损坏。

c) 外部因素/人为导致损坏，例如电路板直接接触金属导致短路烧毁等。

d) 接口不按规范使用，接入高电压导致损坏。

e) 带电拔插驱动器，导致驱动器烧坏。