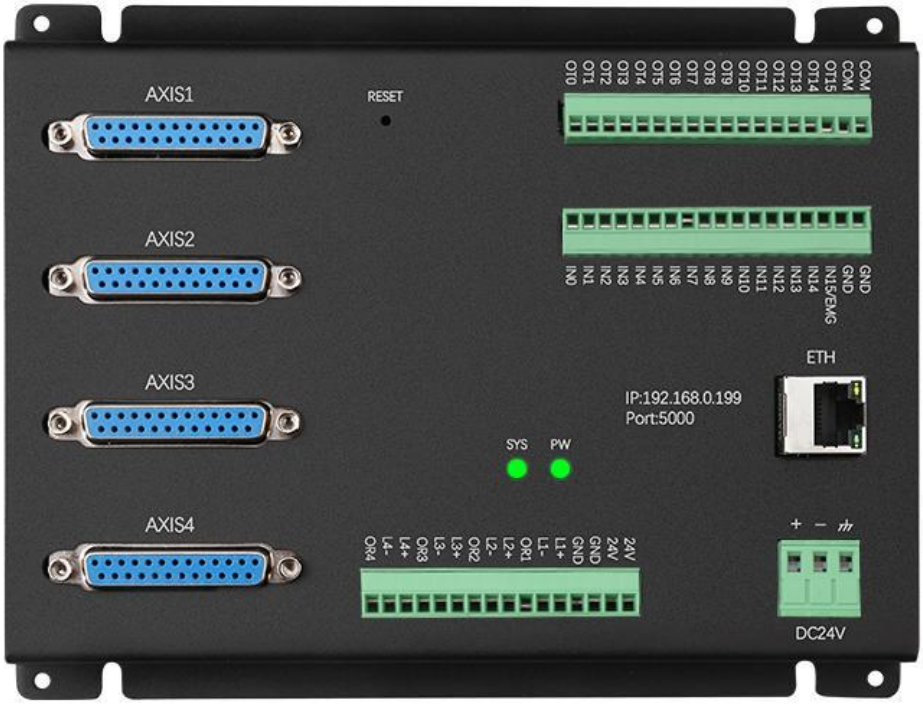
以太网四轴脉冲型运动控制器使用手册



V01	初版	20221112
V02	增加飞拍，脉冲跟随，螺旋插补，读取指令缓存等功能。	20240329

本手册仅供参考，由于产品改进设计和功能等原因，恕不另行通知。

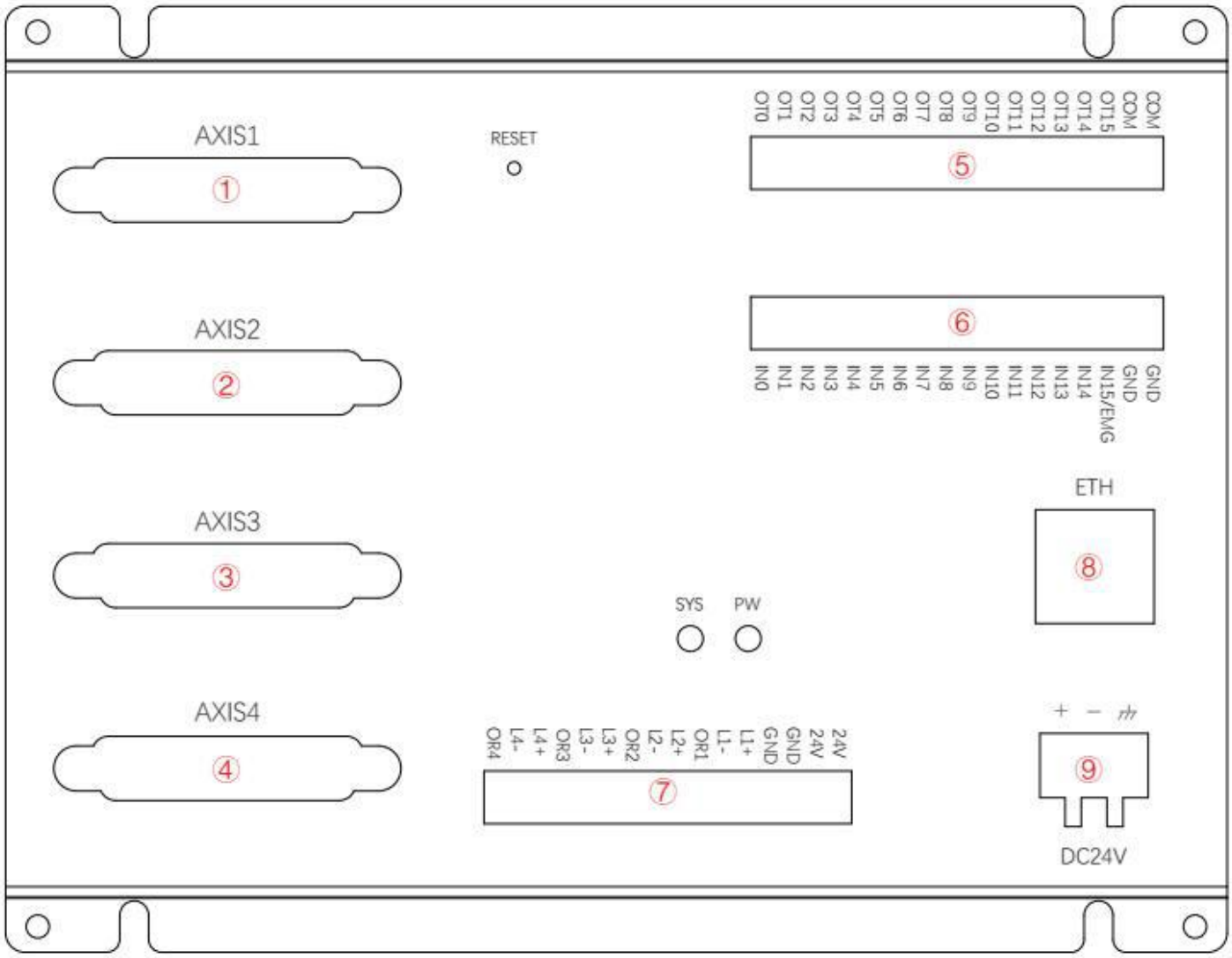
一、产品特点

- 支持单轴运动和单轴停止
- S 型加减速
- 支持直线插补，圆弧插补和 螺旋插补
- 支持连续插补,支持脉冲跟随,支持飞拍
- 支持运动中改变速度
- 支持 1000 条运动指令缓存
- 32 组光电隔离通用数字输入/输出 IO
- 4 组编码器输入
- 各轴均带原点限位，正负限位
- 带伺服电机驱动器专用接口

二、产品参数

网络标准	符合 Ethernet 标准
网络传输速率	100Mbps 全双工
网络接口	1 个自适应 RJ45 端口
指示灯	电源,系统状态,Link/act
电机轴数	4
电机脉冲频率范围	插补：1HZ-2MHZ，单轴：1HZ-500KHZ
脉冲输出最大电流	20mA（吸入）
脉冲信号长度	-2147483648~+2147483647
编码器输入数	4 组
编码器输入频率	最大 1Mhz
编码器计数器长度	-2147483648~+2147483647
正负限位输入数量	4
零限位输入数量	4
伺服到位输入数量	4
伺服报警输入数量	4
伺服准备好输入数量	4
伺服使能输出数量	4
伺服误差清除输出数量	4
默认 IP 地址	192.168.0.199，端口：5000
输入电压	DC24V
通用数字 IO 输入数量	16 个，光电隔离，电流 5-10mA
通用数字 IO 输出数量	16 个，光电隔离，开漏输出，每组最大 500mA
运动控制 函数库	支持 VC, C#, LabVIEW
恢复默认设置	支持
功耗	<1W
工作温度	-10-50°C
储存温度	-20-70°C
湿度	10%-90%，无凝结
尺寸	190*147mm

三、接口图：



① -④：电机驱动器 DB25 连接器

⑤：通用 IO 输出口，COM 为输出公共端

⑥：通用 IO 输入口，其中 IN15 可以通过设置改为紧急停止口

⑦：正负和零限位接口

⑧：RJ45 网络接口

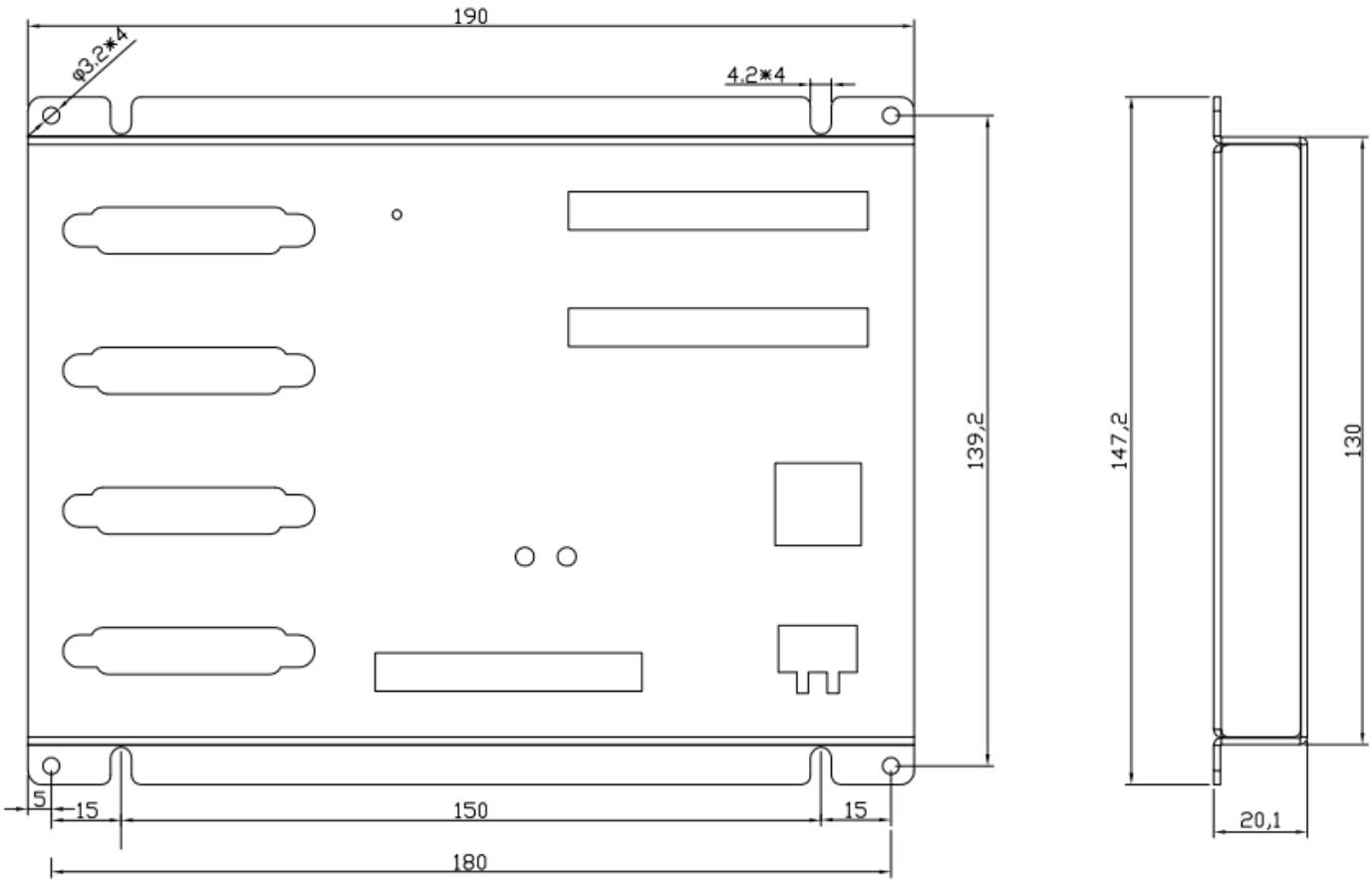
⑨：电源输入接口

RESET：长按 3 秒以上，恢复出厂设定

SYS：系统状态灯，慢速闪烁：空闲，快速闪烁：轴忙碌

PW：电源指示灯

四、安装尺寸图



五、AXIS1-2 接口引脚定义

序号	名称	1/O	说明	序号	名称	1/O	说明
1	OGND	O	24V 电源地	14	24V	O	+24V 输出
2	ALM	I	驱动报警	15	ERC	O	驱动报警复位
3	SEVON	O	驱动允许	16	INP	I	到位信号
4	EA-	I	编码器输入	17	EA+	I	编码器输入
5	EB-	I	编码器输入	18	EB+	I	编码器输入
6	EZ-	I	编码器输入	19	EZ+	I	编码器输入
7	+5V	O	内部 5V	20	GND	O	内部 5V 地
8	NC	-	NC	21	GND	O	内部 5V 地
9	DIR+	O	方向输出	22	DIR-	O	方向输出
10	GND	O	内部 5V 地	23	PUL+	O	脉冲输出
11	PUL-	O	脉冲输出	24	GND	O	内部 5V 地
12	RDY	I	伺服准备完成	25	NC	NC	
13	GND	O	内部 5V 地				

六、通用 IO 输入引脚定义

序号	名称	1/O	说明
1	IN0	I	输入 0 口
2	IN1	I	输入 1 口
3	IN2	I	输入 2 口
4	IN3	I	输入 3 口
5	IN4	I	输入 4 口
6	IN5	I	输入 5 口
7	IN6	I	输入 6 口
8	IN7	I	输入 7 口
9	IN8	I	输入 8 口
10	IN9	I	输入 9 口
11	IN10	I	输入 10 口
12	IN11	I	输入 11 口
13	IN12	I	输入 12 口
14	IN13	I	输入 13 口
15	IN14	I	输入 14 口
16	IN15/EMG	I	输入 15 口或急停
17	OGND	I	24V 电源地
18	OGND	I	24V 电源地

七、通用 IO 输出引脚定义

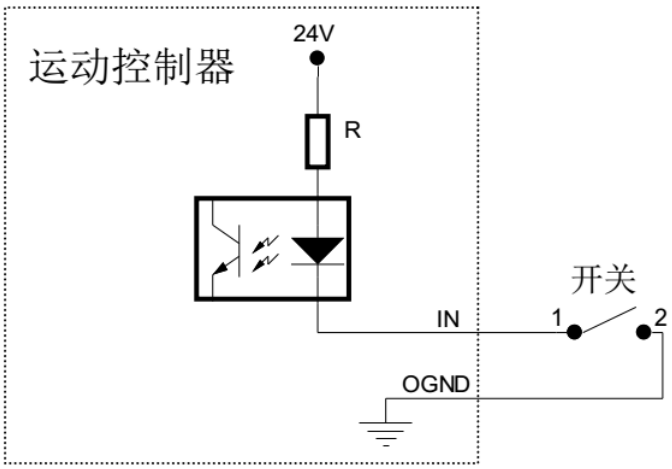
序号	名称	1/O	说明
1	OT0	O	输出 0 口
2	OT1	O	输出 1 口
3	OT2	O	输出 2 口
4	OT3	O	输出 3 口
5	OT4	O	输出 4 口
6	OT5	O	输出 5 口
7	OT6	O	输出 6 口
8	OT7	O	输出 7 口
9	OT8	O	输出 8 口
10	OT9	O	输出 9 口
11	OT10	O	输出 10 口
12	OT11	O	输出 11 口
13	OT12	O	输出 12 口
14	OT13	O	输出 13 口
15	OT14	O	输出 14 口
16	OT15	O	输出 15 口
17	COM	O	公共端
18	COM	O	公共端

八、限位引脚定义

序号	名称	1/O	说明
1	OR4	I	4 轴原点限位
2	L4-	I	4 轴负限位
3	L4+	I	4 轴正限位
4	OR3	I	3 轴原点限位
5	L3-	I	3 轴负限位
6	L3+	I	3 轴正限位
7	OR2	I	2 轴原点限位
8	L2-	I	2 轴负限位
9	L2+	I	2 轴正限位
10	OR1	I	1 轴原点限位
11	L1-	I	1 轴负限位
12	L1+	I	1 轴正限位
13	OGND	O	24V 地
14	OGND	O	24V 地
15	24V	O	24V 输出
16	24V	O	24V 输出

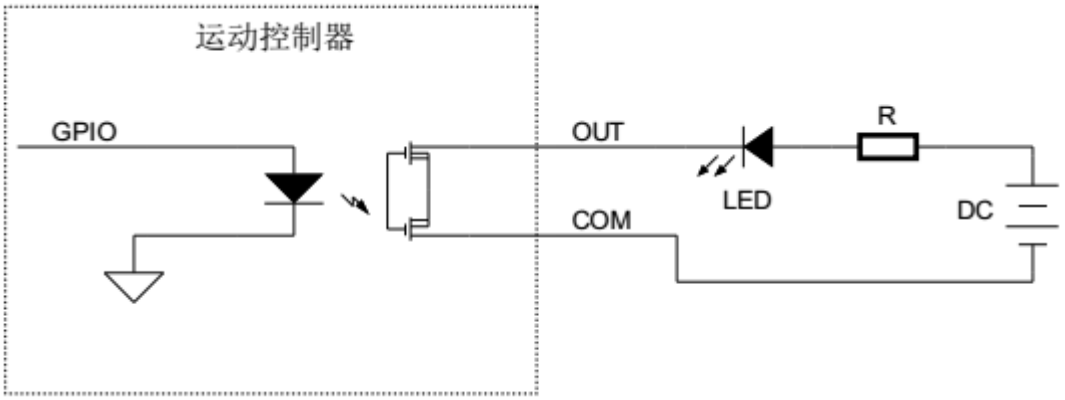
九、通用输入接口说明

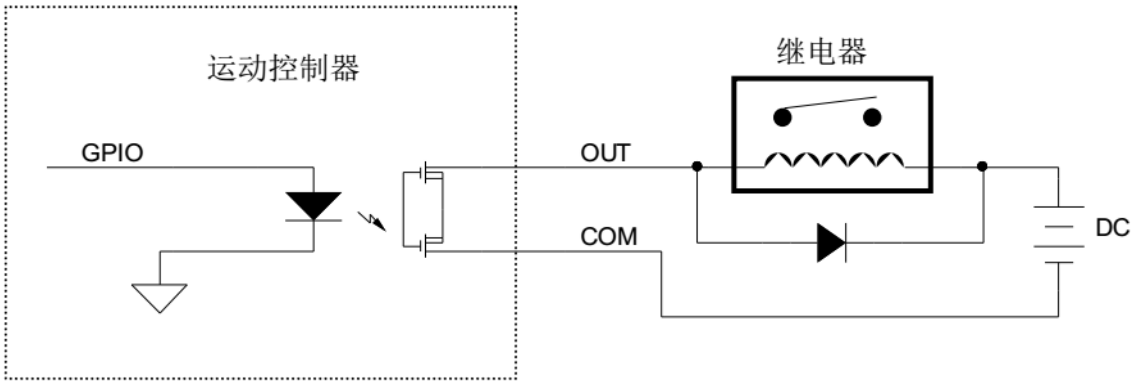
控制器共提供 **16** 组通用数字 IO 输入，每组均带有光电隔离，接口类型为开关量输入，接口原理如下：



十、通用输出接口说明

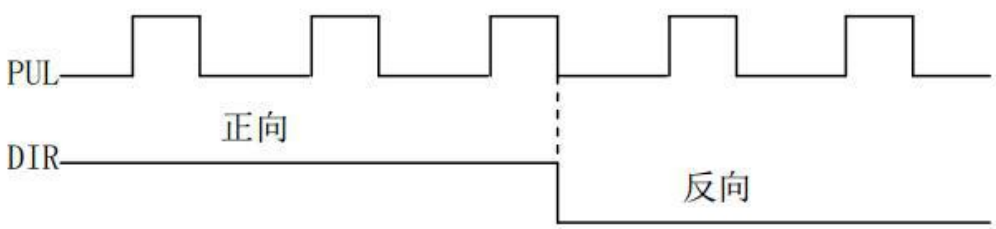
控制器共提供 **16** 组通用数字 IO 输出，各输出端口为光电隔离，开漏输出，下图分别是应用连接 **LED** 和继电器应用示意图，请注意输出端口必须串接负载，再连接到电源，不能直接连接到电源，否则会烧坏设备。



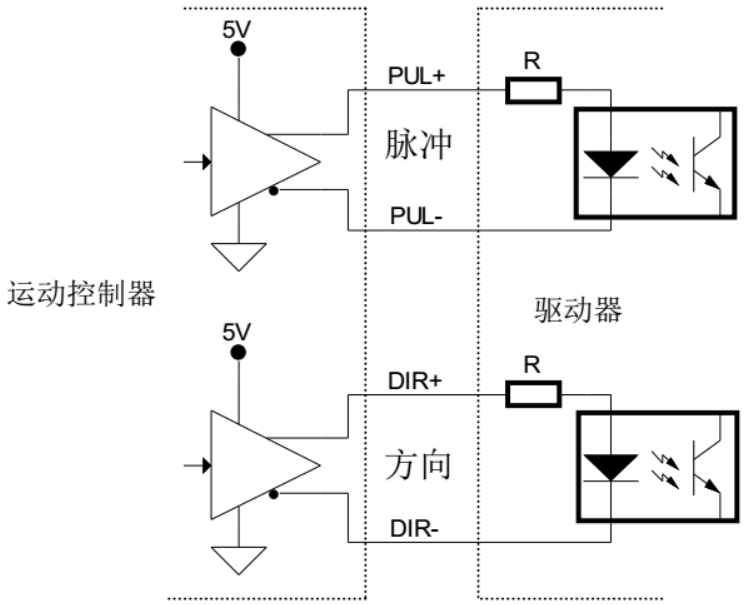


十一、电机控制信号接口

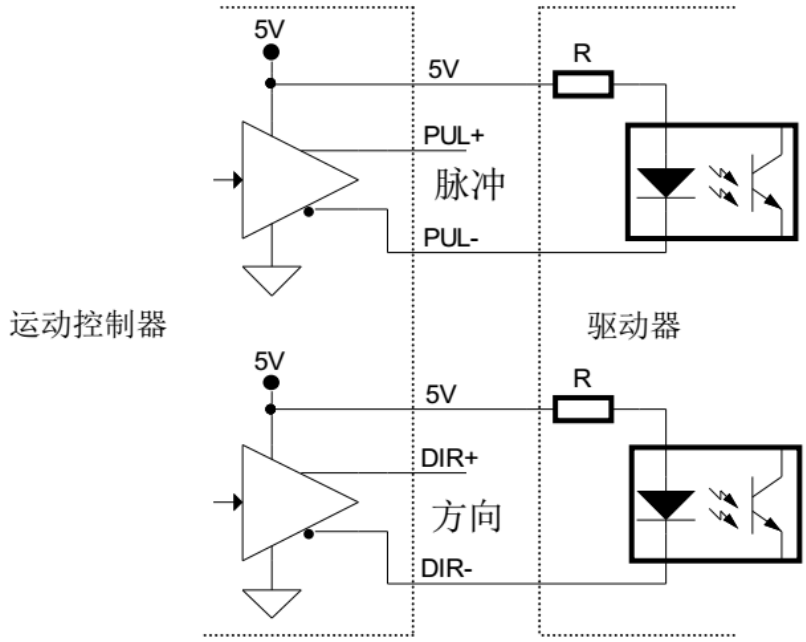
控制器提供脉冲+方向信号模式驱动电机，如下图所示。



与驱动器的差分和单端连接方式如下图：



差分接法

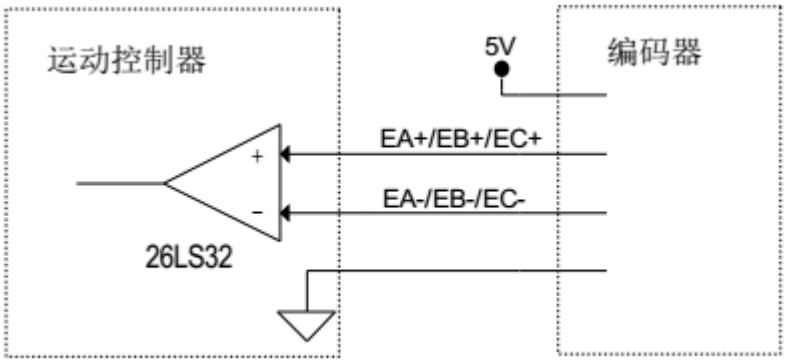


单端接法

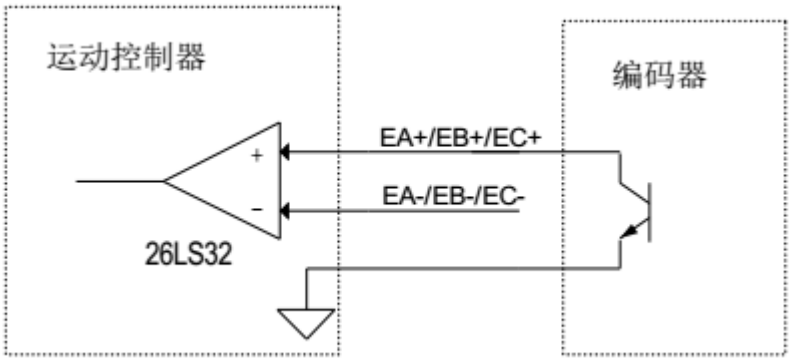
十二、编码器信号输入接口说明

控制器支持 2 种类型的编码器信号输入：非 AB 相脉冲输入和 A/B 相正交信号。目前只支持 1 倍频计数方式，如果使用的编码器为 1000 线，即电机转一圈反馈计数 脉冲数为 1000 个。

如使用差分输出的编码器，输入信号的正端接 EA+(或 EB+， EC+)端，负端接 EA-(或 EB-，EC-)端，如下图所示。



如使用集电极开路输出的编码器，则编码器输出信号接 EA+（或 EB+， EC+）端， EA-（或 EB-，EC-）端悬空，如下图所示。



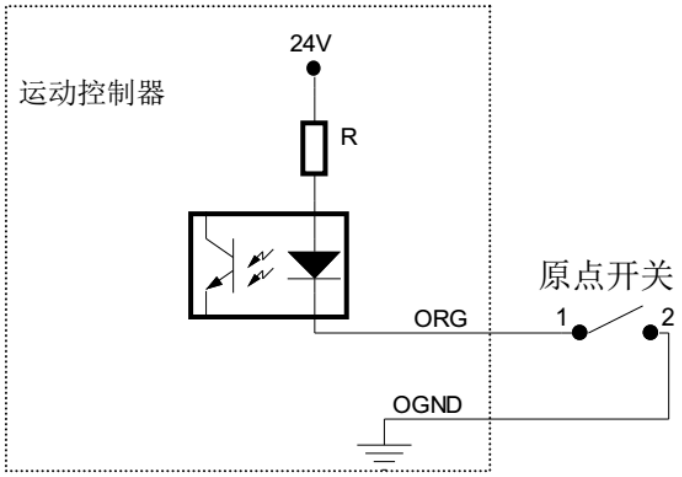
备注:

1) 编码器等脉冲输入信号的 EA+、 EA-、 EB+、 EB-和 EC+、 EC-的差分信号电压差必须高于 3V，小于 5.5V，且输出电流不应小于 5mA。

2) 需要将编码器的 GND 和控制卡的 GND 连接。

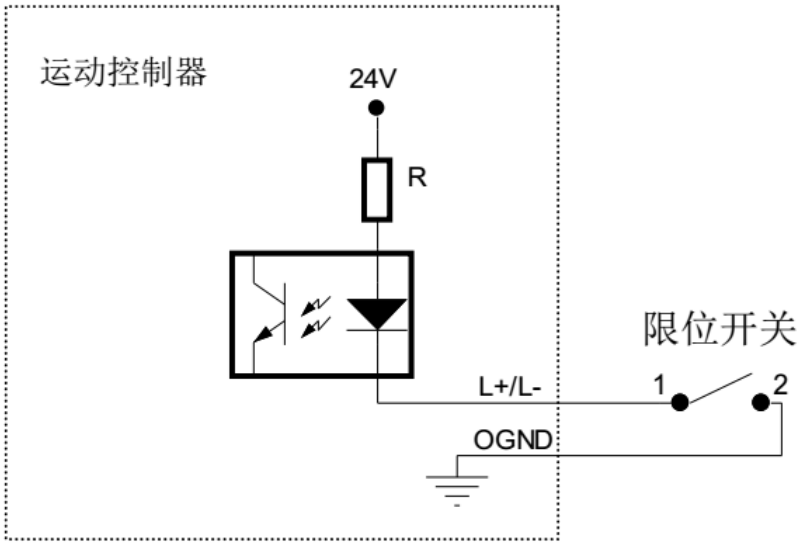
十三、原点信号接口

控制器每轴均配有 1 个 ORG 原点信号接口，接口电路如下图，ORG 信号经光电隔离后送至内部微处理。



十四、限位信号接口

控制器每轴均配有 1 个正限位和 1 个负限位开关接口，接口电路如下图，限位信号经光电隔离后送至内部微处理。注意，控制器只支持低电平有效限位，如选用接近开关，请选择 NPN 输出型。



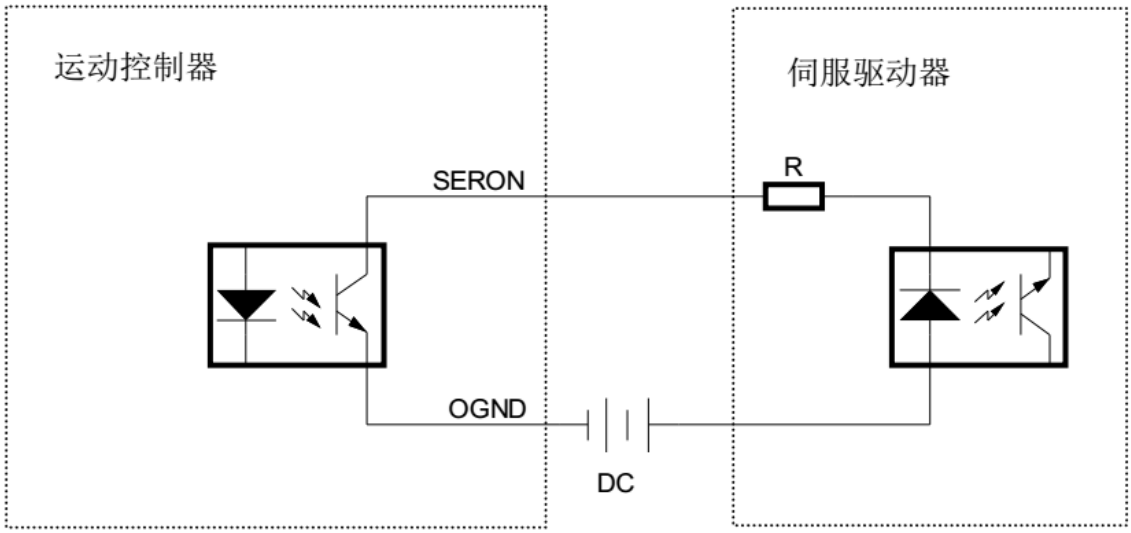
十五、伺服电机驱动器控制信号接口

控制器为每一轴均提供了伺服电机驱动器专用信号接口，其中信号 RDY、ALM 和 INP 用于监控伺服电机状态， 信号 SEVON 和 ERC 用于设置伺服电机状态。

15.1 驱动器使能信号

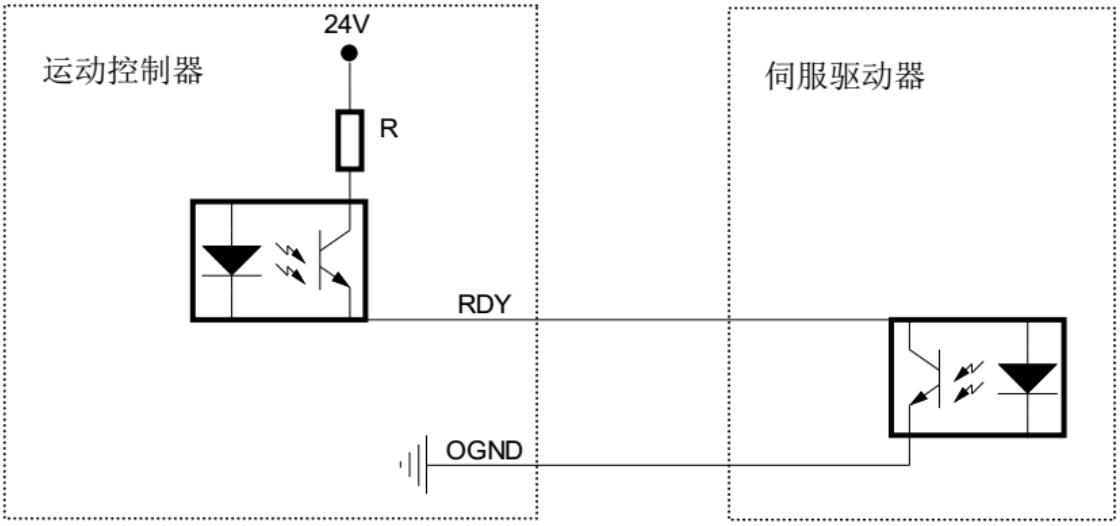
当伺服电机驱动器的 SEVON 信号为无效状态时，伺服电机驱动器不工作，电机处于自由状态；当 SEVON 信号有效时，伺服电机驱动器进入工作状态，电机锁紧。

接口原理如下图：



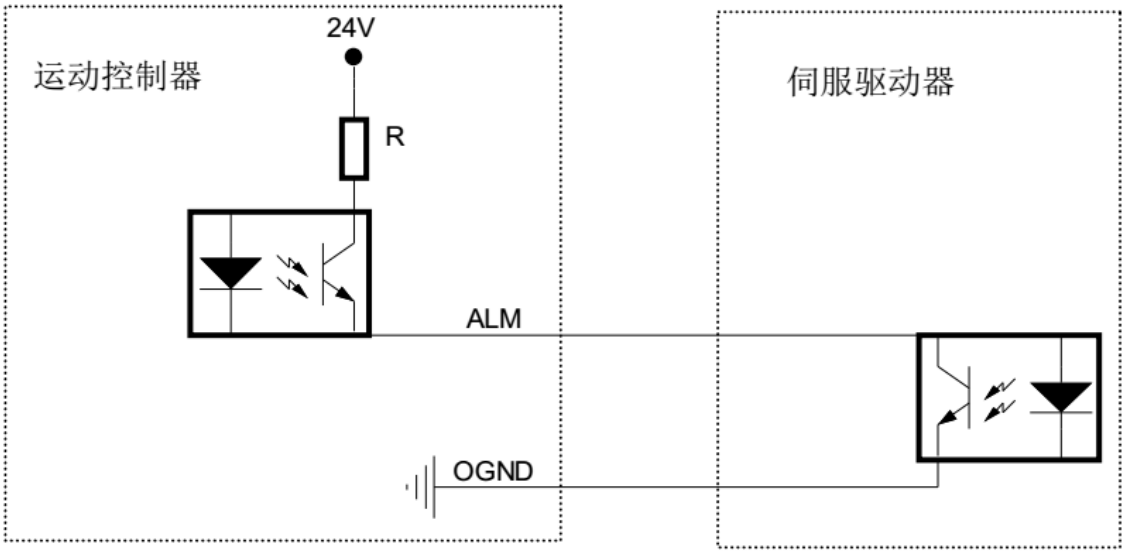
15.2 伺服准备好信号

当伺服电机驱动器处于准备好状态， RDY 信号就会自动将该信号置为有效。 此时， 控制器可以向伺服电机驱动器发出运动命令。 其接口电路原理图如下图 所示。



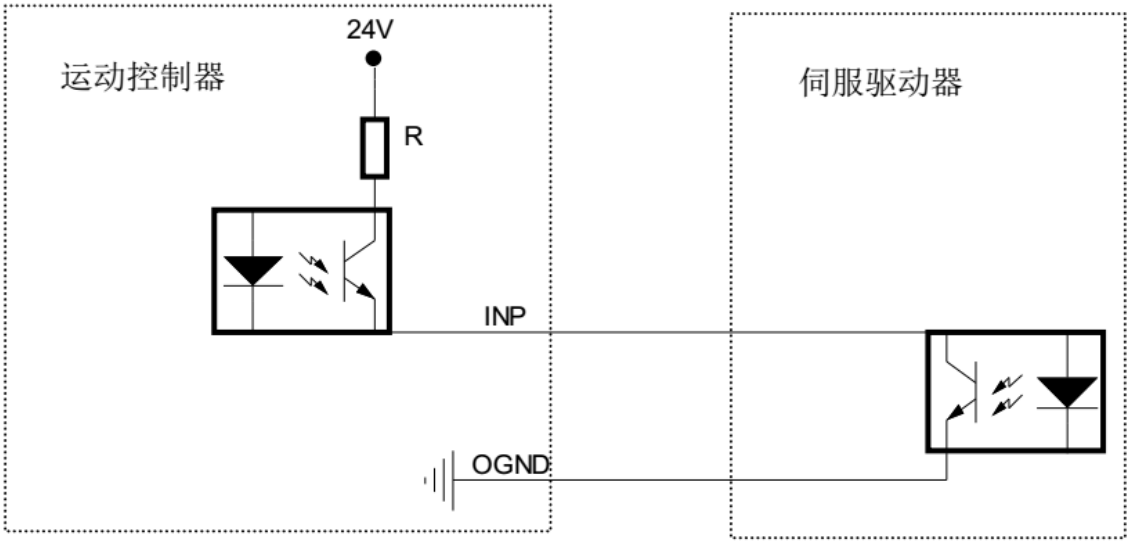
15.3 驱动器报警信号

ALM 信号是伺服电机驱动器发出的报警信号。当控制器接收到 ALM 信号后，将立即停止各轴运动。 其接口电路原理图如下图所示。



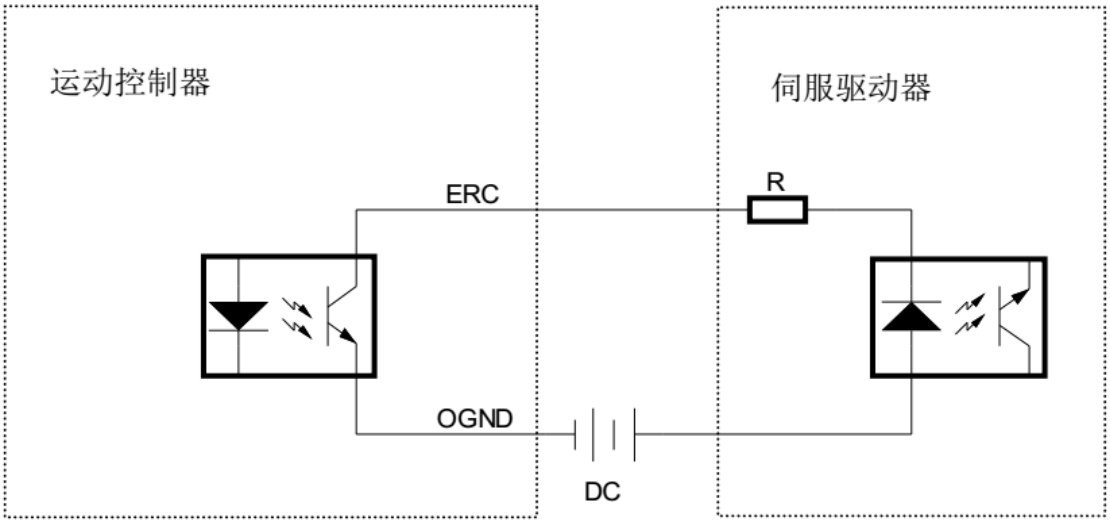
15.4 驱动器位置到达信号

控制器均为每一轴提供了用于监控伺服电机定位结果的 INP 信号接口。典型接口及接线原理如下图所示。**注意：** 当使能了 INP 信号功能时，只有在 INP 信号为有效状态下，对应的轴才能进行运动，否则该轴将不能运动。



15.5 驱动器位置偏差清除信号

控制器均为每一轴提供了用于清零伺服驱动位置偏差计数器的 ERC 信号输出接口。典型接口及接线原理如下图。



十六、恢复出厂设定按键

控制器提供一个硬件恢复出厂设定按键，当长按 **3** 秒以上，控制器将恢复默认的出厂设定值，包括 **IP** 地址和端口号，**INP,ALM** 使能和有效电平设置，编码器模式设定和 **EZ** 有效电平设定。

十七、函数库说明

17.1 连接控制器函数

string Connect()

功能：连接控制器

参数：无

成功返回值：Address +Connect Succeed

失败返回值：Address+ Connect Fail

17.2 断开连接函数

void Disconnect()

功能：断开与控制器的网络连接

参数：无

返回值：无

17.3 复位函数

int MotionCardReset()

功能：复位控制器

参数：无

成功返回值：0

失败返回值：1

提示：复位成功之后，网络连接会断开，需要重新连接控制器。

17.4 设置原点函数

int SetMoveHomeModel(int motorIndex, int direction, int mode)

功能：设置指定轴回到原点

参数 1：motorIndex，轴号 1~4

参数 2：direction，回原方向: 00-反向 01-正向(负限位不支持正向)

参数 3：model，回原点模式，支持如下 6 种:

0-一次性回原点
1-一次性回原点+回找
2-二次回找原点
3-负限位一次性回原点
4-负限位一次性回原点+回找
5-负限位二次回找原点

返回值：0

失败返回值：1

17.5 读取回原点状态函数

int ReadIsOrgRunig(int motorIndex)

功能：读取指定轴回原点执行状态

参数：motorIndex，轴号(1~4)

返回值：返回执行状态：1-已完成，0-未完成

提示：当进行回原点设置后，可通过这个函数，读取指定轴的回零执行状况，也可以读取该轴的坐标是否为 0 来判断。

17.6 设置轴运行速度函数

int SetSpeed(int motorIndex, int highSpeed, ushort lowSpeed, short acc, short dec)

功能：设置指定轴的速度

参数 1：Index，轴号(0~4)，0-插补速度，1~4 单轴速度

参数 2：highSpeed，运行速度:，范围：[单轴 1~320KHz]，[插补 1~800KHz]

参数 3： lowSpeed，加速开始速度和减速终止速度，范围：1000Hz~65535Hz

参数 4：acc，加速时间，范围：100~1000，单位毫秒

参数 5：dec，减速时间，范围：100~1000，单位毫秒

成功返回值：0

失败返回值：1

默认参数:

运行速度：单轴和运行均为 5KHZ
加速开始速度和减速终止速度：1KHZ
加速时间：1000 毫秒
减速时间：1000 毫秒

17.7 运动中改变速度函数

int ChangeSpeed(int motorIndex, int Speed)

功能：运动中改变指定轴的运行速度

参数 1：motorIndex，轴号(0~4)，0，插补速度，1~4，单轴速度

参数 2： Speed，速度值（速度范围:插补 1~800kHz，单轴 1-320KHZ）

成功返回值：0
失败返回值：1

注意：运动中变速，当从低速变为高速时，请注意速度调节不宜太高，应逐步提高速度，否则会出现电机卡死，如从 1KHZ 变速为 10KHZ，应先变速为 5KHZ，再提高到 10KHZ。具体根据实际情况调节。

17.8 单轴点位运动函数

int Move(int motorIndex, int pulse, int mode)

功能：指定轴做单轴点位运动

参数 1：motorIndex，轴号(1~4)

参数 2：pulse，脉冲数（范围：-2147483648~+2147483647）

参数 3：mode，模式，0 绝对位移，1 相对位移

成功返回值：0

失败返回值：1

17.9 四轴轴直线插补运动函数

int LineMotion(int xpulse, int ypulse, int zpulse, int upulse, int Mode)

功能：四轴直线插补运动

参数 1：xpulse，1 轴脉冲数,脉冲范围范围（-2147483648~+2147483647）

参数 2：ypulse，2 轴脉冲数,脉冲范围范围（-2147483648~+2147483647）

参数 3：zpulse，3 轴脉冲数,脉冲范围范围（-2147483648~+2147483647）

参数 4：upulse，4 轴脉冲数,脉冲范围范围（-2147483648~+2147483647）

参数 5：Mode，位移模式：0-绝对位移，1-相对位移

正确返回值：0

失败返回值：1

17.10 设置圆弧插补精度函数

int SetCircleAccuracy(int value)

功能：设置圆弧插补精度

参数：value，范围：1~8，设置圆弧精度，值越低精度越高，默认值为 3

成功返回值：0

失败返回值：1

17.11 二轴圆弧插补运动函数

int CircleMotion(int motorIndex0 ,int motorIndex1, int targetPos0, int targetPos1, int CentrePos0, int CentrePos1, int dir, int Mode)

功能：任意二轴圆弧插补运动

参数 1：motorIndex0，参与插补 X 轴的轴号

参数 2：motorIndex1，参与插补 Y 轴的轴号

参数 3： targetPos0，圆弧插补的终点 X 坐标，范围（-2147483648~+2147483647）

参数 4： targetPos1，圆弧插补的终点 Y 坐标，范围（-2147483648~+2147483647）

参数 5：CentrePos0，圆弧插补的圆心点 X 坐标，范围（-2147483648~+2147483647）

参数 6：CentrePos1，圆弧插补的圆心点 Y 坐标，范围（-2147483648~+2147483647）

参数 7：dir，圆弧插补模式：0-顺时针，1-逆时针

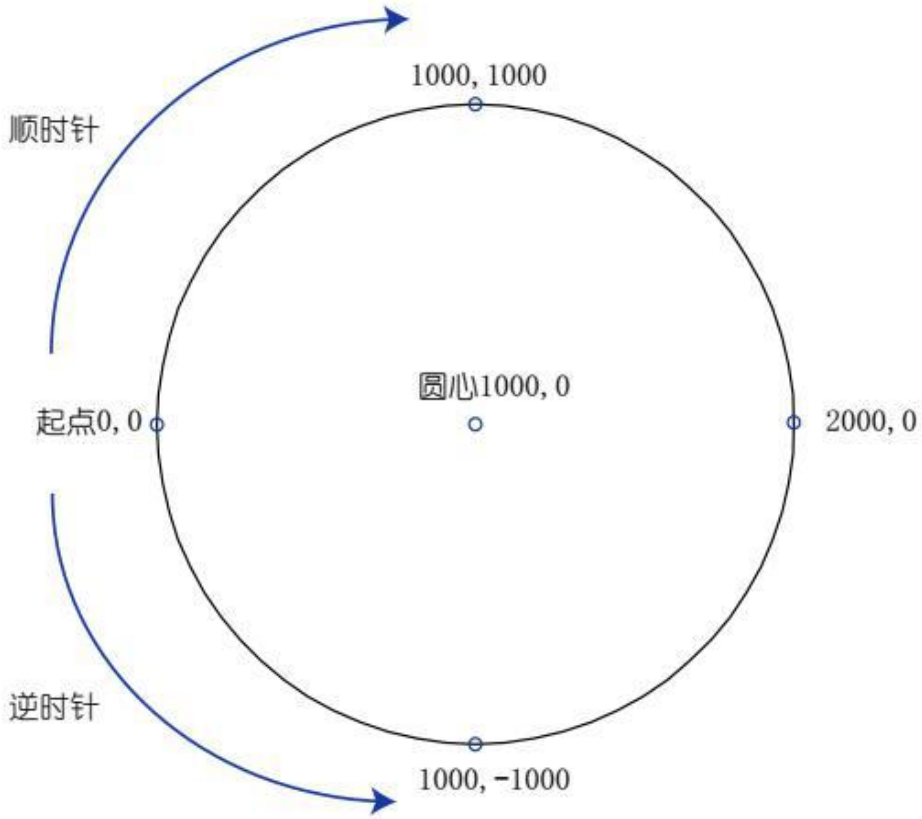
参数 8：Mode，圆弧插补模式：0-画圆，1-角度画圆弧(终点 X 坐标设置角度值)

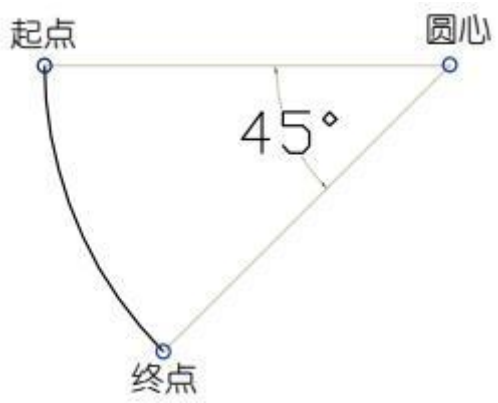
正确返回值：0

失败返回值：1

说明:

1. 终点 X 坐标和 Y 坐标，设置为当前坐标，会画一个 360 度的圆，如果终点 X 坐标和 Y 坐标设置在圆的任意坐标点，则当运行到该坐标点时就会停止，如果设置的终点 X 坐标和 Y 坐标不在圆的轨迹上，则视为无效。如下图，当设置终点 X 坐标和 Y 坐标为 2000,0 时，不管设置顺时针还是逆时针，运行到 2000,0 的位置时，就会停止，即完成一个 180 度半圆。
2. 模式为 0，画圆，模式为 1 时，终点 X 坐标设置角度值，如设置 45，逆时针则会画一个起点与圆心夹角为 45 度的圆弧，如下图。请注意角度值为 5 的倍数。





提示：因圆弧插补时，电机会瞬间反转，如果电机的扭矩不够，会造成轨迹偏移，因此圆弧插补时，运行速度不宜偏高，具体根据实际情况调整。

17.12 读取轴状态函数

int ReadMotorState(int motorIndex)

功能：读取当前轴的状态

参数：motorIndex，轴号，1~4

返回值：0 空闲，1 忙碌，-1 错误

17.13 设置编码器计数方式函数

int SetCountMode(int motorIndex, int countMode)

功能：设置指定轴编码器的计数方式

参数：motorIndex，轴号(1~4)

countMode，计数方式：00-非 AB 相，01-AB 相(默认)

成功返回值：0

失败返回值：1

17.14 读取编码器计数值函数

byte[] ReadEnCoderPulse(int motorIndex)

功能：读取指定轴编码器计数值

参数：motorIndex，轴号(1~4)

返回值: 计数值（范围：-2147483648~+2147483647）

错误返回： FF

17.15 读取编码器计数方式函数

int ReadCountMode(int motorIndex)

功能：读取指定轴编码器的计数方式

参数：motorIndex ， 轴号(1~4)

返回值： 00-非 AB 相, 01-AB 相(默认),-1 为错误

17.16 设置编码器脉冲计数值函数

int SetEnCoderPulse(int motorIndex, int pulse)

功能：设置指定轴编码器脉冲计数值

参数：motorIndex ， 轴号(1~4)

pulse 脉冲数（范围：-2147483648~+2147483647）

成功返回值：0

失败返回值：1

17.17 读取 EZ 信号有效电平设置函数

int ReadEZ(int motorIndex)

功能：读取指定轴的 EZ 信号有效电平设置

参数：motorIndex ， 轴号(1~4)

返回值：0-低电平 1-高电平 -1 为错误

17.18 设置 EZ 信号有效电平函数

int SetEZ(int index, int voltagelevel)

功能：设置指定轴的 EZ 信号有效电平

参数：Index， 轴号(1~4)

Voltagelevel， 有效电平，1 表示高电平有效，0 表示低电平有效

成功返回值：0

失败返回值：1

注意：如果没有编码器没有 EZ 信号，需设置有效电平为高

17.19 设置指令脉冲函数

int SetCommandPulse(int motorIndex, int pulse)

功能：设置指定轴的指令脉冲位置

参数：motorIndex， 轴号(1~4)

pulse，范围：-2147483648~+2147483647

成功返回值：0

失败返回值：1

说明：指令脉冲，即各轴的坐标。如果需将坐标归零，设置为脉冲数为 0 即可。

17.20 读取指令脉冲函数

byte[] ReadCommandPulse(int motorIndex)

功能：读取指定轴的指令脉冲位置

参数：motorIndex， 轴号(1~4)

返回值: 脉冲数

错误返回： FF

17.21 设置 ALM 信号使能和有效电平函数

int SetAlm(int index, int state, int voltagelevel)

功能：设置指定轴的 ALM 信号使能和有效电平

参数：index ， 轴号(1~4)

state ， ALM 使能设置，0-禁止，1-允许

voltagelevel， ALM 有效电平，1 表示高电平有效，0 表示低电平有效

成功返回值：0

失败返回值：1

17.22 读取 ALM 电平函数

int ReadALM(int motorIndex)

功能：读取指定轴的 ALM 电平

参数：motorIndex，轴号(1~4)

返回值：0 低电平，1 高电平，-1 错误

17.23 读取 ALM 信号使能和有效电平设置函数

dynamic ReadAlmSetting(int motorIndex)

功能：读取指定轴的 ALM 使能设置与 ALM 有效电平设置

参数：motorIndex，轴号(1~4)

返回值:dynamic =null 为错误指令

返回值:dynamic.Enable 为 ALM 使能 00-禁止，01-允许

返回值:dynamic.Level 为 ALM 电平 00-低电平 01-高电平

17.24 设置 INP 信号使能和有效电平函数

int SetINP(int index, int state, int voltagelevel)

功能：设置指定轴的 INP 使能与有效电平

参数：Index，轴号(1~4)

state，INP 使能设置，0-禁止，1-允许

voltagelevel，有效电平，1 表示高电平有效，0 表示低电平有效

成功返回值：0

失败返回值：1

17.25 读取 INP 信号使能和有效电平设置函数

dynamic ReadINPSetting(int motorIndex)

功能：读取指定轴的 INP 信号参数设置

参数：motorIndex，轴号(1~4)

返回值:dynamic =null 为错误指令

返回值:dynamic.Enable 为 INP 使能 00-禁止，01-允许

返回值:dynamic.Level 为 INP 电平 00-低电平 01-高电平

17.26 读取 INP 信号电平函数

int ReadINP(int motorIndex)

功能：读取指定轴的 INP 电平

参数：motorIndex，轴号(1~4)

返回值：0-低电平 1-高电平 -1 为错误

17.27 读取 RDY 信号电平函数

int ReadRDY(int motorIndex)

功能：读取指定轴的 RDY 端口的电平

参数：motorIndex，轴号(1~4)

返回值：0-低电平 1-高电平 -1 为错误

17.28 读取 Servo 信号电平函数

int ReadServoEnable(int motorIndex)

功能：读取指定轴的伺服使能端口的电平

参数：motorIndex，轴号(1~4)

返回值：0-低电平 1-高电平 -1 为错误

17.29 控制 Servo 信号输出函数

int SetServoEnable(int index, int voltagelevel)

功能：控制指定轴的伺服使能端口的输出

参数：Index，轴号(1~4)

Voltagelevel，输出电平 0-禁止，1-允许

成功返回值：0

失败返回值：1

17.30 读取 ERC 信号电平函数

int ReadERC(int motorIndex)

功能：读取指定轴的 ERC 端口电平

参数：motorIndex， 轴号(1~4)

返回值：0-低电平， 1-高电平 -1 为错误

17.31 控制 ERC 信号输出函数

int SetECR(int index, int voltagelevel)

功能：控制指定轴的 ERC 信号输出

参数：Index， 轴号(1~4)

voltagelevel， 输出电平： 0-低电平 1-高电平

成功返回值：0

失败返回值：1

17.32 读取原点限位状态函数

int ReadORG(int motorIndex)

功能：读取指定轴的原点开关限位状态

参数：motorIndex， 轴号(1~4)

返回值：0-已限位，1-未限位,-1 错误

17.33 读取正负限位状态函数

dynamic ReadPelNel(int motorIndex)

功能：读取指定轴的正负限位开关状态

参数：motorIndex， 轴号(1~4)

返回值：dynamic.Pel 为正限位，dynamic.Nel 为负限位，错误值 dynamic = null

00 已限位 01-未限位

17.34 读取输入 IO 电平函数

int ReadDigitalInput(int index)

功能：读取指定轴的某个输入端口的电平

参数：index， 通道编号(0~15)

返回值：返回值：0-低电平 1-高电平 -1 为错误

17.35 读取输出 IO 电平函数

int ReadDigitalOutput(int index)

功能：读取指定的某个输出端口的电平

参数：index，通道编号(0~15)

返回值：返回值：0-低电平 1-高电平 -1 为错误

17.36 控制输出 IO 电平函数

int SetDigitalOutput(int index, int voltagelevel)

功能：设置指定的某个输出端口的电平

参数：Index，输出端口号(0-15)

voltagelevel，0-低电平 1-高电平

成功返回值：0

失败返回值：1

17.37 设置急停函数

int SetStopPort(int level)

功能：设置急停端口

参数：使能 IN15 端口为急停，1-使能，0-禁用

成功返回值：0

失败返回值：1

17.38 停止轴运动函数

int StopMotor(int motorIndex)

功能：停止指定轴

参数：motorIndex，轴号，0-停止所有轴，1~4 停止单轴

成功返回值：0

失败返回值：1

请注意：轴号为 0 时，停止所有轴，并清除缓存运动指令。

17.39 修改 IP 地址函数

int SetAddress(string ip, int port)

功能：修改 IP 地址和端口号

参数：ip，IP 地址号

port，端口号

成功返回值：0

失败返回值：1

17.40 设置脉冲跟随

int Set_follow_move(int main_axis, int slave_axis,int state)

功能：设置脉冲跟随

参数：main_axis，主轴

slave_axis，从轴

state，状态，1-启用，0-禁用

成功返回值：0

失败返回值：1

说明：

1. 当启用某一个轴的脉冲跟随功能后，设备每 10ms 获取主轴的编码器值，并计算出主轴的运动速度和位置，以此驱动从轴运动，使从轴跟随主轴保持相同速度转动。
2. 主轴必须要连接编码器，否则从轴不能获取跟随数据。
3. 可任意设置主轴和从轴，从轴数量最多可设置 3 个。
4. 脉冲跟随功能，断电不保存设置，再次开机需要重新设置。

17.41 设置飞拍功能

int Set_feipai(int axis, int output_io_number, int delaytime, int setcnt)

功能：设置脉冲跟随

参数：axis，轴号

output_io_number， 指定输出的 IO 编号

delaytime，输出 IO 开通时间，到达设定时候后关断
setcnt，设置编码器的计数值（即到达改值后开通指定的输出 IO）

成功返回值：0

失败返回值：1

说明：

1. 每个轴可以设置 1 个飞拍点，当指定的轴编码器到达设定值后，指定的输出 IO 会开通，延时设定延时后，输出 IO 就关闭
2. 断电不保存设置，再次开机需要重新设置。

17.42 读取指令缓冲数量

int ReadBuffer()

功能：读取指令缓存

参数：无

返回值：当前剩余指令数量

17.43 螺旋插补运动函数

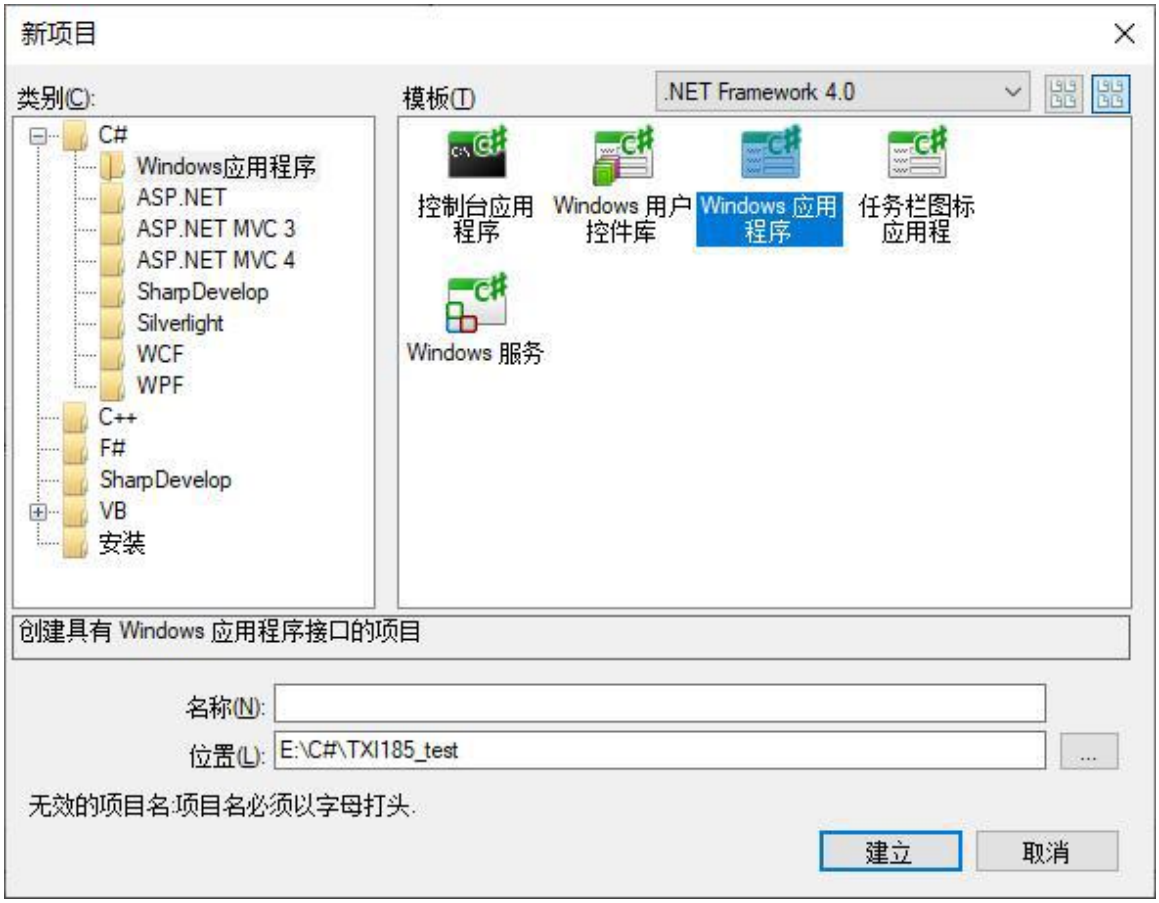
int SpiralMotion(byte x, byte y, byte z, int xTargetPos, int yTargetPos, int xCentrePos, int yCentrePos, int zTargetPos, byte dir, byte Mode)

功能：任意 3 轴螺旋插补

```
/// 参数 x 轴（1-4）
/// 参数 y 轴（1-4）
/// 参数 z 轴（1-4）
/// 参数 xTargetPos:终点 X 坐标,范围（-16777215~+16777215）
/// 参数 yTargetPos:终点 Y 坐标,范围（-16777215~+16777215）
/// 参数 xCentrePos:圆心点 X 坐标,范围（-16777215~+16777215）
/// 参数 yCentrePos:圆心点 Y 坐标,范围（-16777215~+16777215）
/// 参数 zTargetPos:z 轴的目标坐标
/// 参数 dir:方向:0-顺时针插补,1-逆时针插补
/// 参数 Mode:圆弧插补位移模式:0-画圆,1-角度画圆弧
/// 返回值：0 为成功 1,-1 为失败正确返回值：0
```

十八、函数库范例

以 C#为例，新建一个 Windows 应用程序



引用声明

```
using TXI183_DLL;
MotionCard MotionCard = new MotionCard ();
```

```
21      public partial class MainForm : Form
22      {
23          MotionCard MotionCard = new MotionCard ();
24          public MainForm()
25          {
26              InitializeComponent();
27          }
```

以下为函数调用范例：

```
void Button1Click(object sender, EventArgs e) //连接到控制器
{
  if (button1.Text == "关闭") {
    MotionCard.Disconnect();
```

```
        button1.Text = "连接";
    } else {
        string result;
        string rms;
        MotionCard.address = IP.Text; //IP 地址
        MotionCard.Port = int.Parse(Port.Text); //端口号
        result = MotionCard.Connect();
        log.Text = "" + result;     //显示结果
        rms = log.Text.Substring(log.TextLength - 7);  //判断是否连接成功
        if (rms == "Succeed") {
            button1.Text = "关闭";
        }
    }


void Button9Click(object sender, EventArgs e)//设置速度
{
    int axis = int.Parse(axis_no.Text); //轴号
    int runspeed = int.Parse(speed.Text);//运行速度
    ushort startspeed = ushort.Parse (lowspeed .Text); //加速起始和终止速度
    short acc = short.Parse(textBox2.Text);  //加速时间
    short dec = short.Parse(textBox3.Text);    //减速时间
    log.Text = "" + MotionCard.SetSpeed(axis, runspeed,startspeed, acc, dec);  //显示结果
}

void Button3Click(object sender, EventArgs e) //单轴运动
{
    int x = int.Parse(x_pulse.Text); //脉冲数
    int axis = int.Parse(axis_no.Text); //轴号
    log.Text = "" +MotionCard.Move(axis, x, 1);  //1,相对移动
}


void Button2Click(object sender, EventArgs e) //读坐标
{
    int axis = int.Parse(axis_no.Text);  //轴号
    log.Text = "" + MotionCard.ReadCommandPulse(axis); //显示结果
}


void Button4Click(object sender, EventArgs e) //设置坐标
{
    int axis = int.Parse(axis_no.Text); //轴号
    int position = int.Parse(x_pulse.Text); //脉冲值
    MotionCard.SetCommandPulse(axis, position);
}


void Button7Click(object sender, EventArgs e) //读正负限位
{
    int p_state, n_state;
    int axis = int.Parse(axis_no.Text); //轴号
```

```
        var nelpel = MotionCard.ReadPelNel(axis);
        bool pel = nelpel.Pel == 0 ? true : false;
        bool nel = nelpel.Nel == 0 ? true : false;
        if (pel == true)
            p_state = 0;
        else
            p_state = 1;
        if (nel == true)
            n_state = 0;
        else
            n_state = 1;
        log.Text = "" + p_state + n_state;  //显示结果

    }


    void Button10Click(object sender, EventArgs e) //直线插补运动
    {
        int x = int.Parse(x_pulse.Text); //1 轴脉冲
        int y = int.Parse(y_pulse.Text); //2 轴脉冲
        int z = int.Parse(z_pulse.Text); //3 轴脉冲
        int u = int.Parse(u_pulse.Text); //4 轴脉冲
        log.Text = "" +MotionCard.LineMotion(x, y, z, u, 1);//显示结果
    }



    void Button12Click(object sender, EventArgs e) //圆弧插补
    {
        int dir = int.Parse(dirBox.Text); //方向
        int mode = int.Parse(modeBox.Text); //模式
        int xcenter = int.Parse(x_pulse.Text); //x 圆心坐标
        int ycenter = int.Parse(y_pulse.Text); //y 圆心坐标
        log.Text = "" +MotionCard.CircleMotion(1, 2，  0, 0, xcenter, ycenter, dir, mode);//参与插补 1 轴和 2 轴，终
点坐标为 0,0
    }


    void StopClick(object sender, EventArgs e)//停止轴运行
    {
        int axis = int.Parse(axis_no.Text); //轴号
        log.Text = "" +MotionCard .StopMotor (axis); //显示结果
    }


    void Button13Click(object sender, EventArgs e)//设置急停端口
    {
      int mode = int.Parse(modeBox.Text);  //0 取消，1 允许 input15 为急停
     log.Text = "" + MotionCard .SetStopPort (mode);//显示结果
    }
```

```
void Button6Click(object sender, EventArgs e)//读编码器
{
   int axis = int.Parse(axis_no.Text); //轴号
   log.Text = "" + MotionCard.ReadEnCoderPulse(axis);//显示结果
}


void Button5Click(object sender, EventArgs e) //设置编码器归 0
{
   int axis = int.Parse(axis_no.Text); //轴号
   MotionCard.SetEnCoderPulse(axis, 0); //数值位 0
}
```