

VM4000Mini、VM4000、VM4000Pro

硬件软件简明手册 V1.0



版权所有 不得翻印

长沙精眸科技有限公司保留该文件所有权

目录

一、 概述	4
1. 功能及特点	4
2. 主要用途	4
3. 接线管型端子针型接线端子	4
二、 控制器类型	5
1. 电脑 IP 设置	7
2. 光源控制	8
三、 通用输入和通用输出示意图	9
1. 通用输入信号	9
2. 通用输出信号	9
3. 飞拍输出接线(VM4000、VM4000Pro)	9
4. 连接步进电机示意图	10
5. 连接伺服电机示意图(VM4000、VM4000Mini)	10
6. X/Y/Z 编码器连接示意图(VM4000)	11
四、 api 函数说明 (C++,三个类型控制器通用)	12
1. MotionV4000_API int __stdcall MotionV4000_initial()	12
2. MotionV4000_API int __stdcall MotionV4000_is_connect()	12
3. MotionV4000_API int __stdcall MotionV4000_get_Type()	12
4. MotionV4000_API int __stdcall MotionV4000_close()	12
5. MotionV4000_API int __stdcall MotionV4000_get_input_bit(int id,bool &status)	12
6. MotionV4000_API int __stdcall MotionV4000_get_input_bit(bool *status)	12
7. MotionV4000_API int __stdcall MotionV4000_set_output(int id, bool status)	12
8. MotionV4000_API int __stdcall MotionV4000_set_output(int id, int status)	13
9. MotionV4000_API int __stdcall MotionV4000_set_outputAll(bool status)	13
10. MotionV4000_API int __stdcall MotionV4000_get_move_state(int iAixs, bool &status)	13
11. MotionV4000_API int __stdcall MotionV4000_get_home_state(int iAixs, bool &bHome)	13
12. MotionV4000_API int __stdcall MotionV4000_get_leftRight_softLimit(int iAixs, double &leftLimit, double &rightLimit)	13
13. MotionV4000_API int __stdcall MotionV4000_go_home(int iAixs, float speed=0)	13
14. MotionV4000_API int __stdcall MotionV4000_set_speed(int iAixs, float speed)	14
15. MotionV4000_API int __stdcall MotionV4000_get_max_speed(int iAixs, int &speed)	14
16. MotionV4000_API int __stdcall MotionV4000_get_acc_time(int iAixs, double &time)	14
17. MotionV4000_API int __stdcall MotionV4000_single_move_to(int iAixs, double pos)	14
18. MotionV4000_API int __stdcall MotionV4000_xyz_move_to(double posX, double posY, double posz)	14
19. MotionV4000_API int __stdcall MotionV4000_jog_move(int iAixs, bool dir)	14
20. MotionV4000_API int __stdcall MotionV4000_stop(int iAixs, int iMode=0)	15
21. MotionV4000_API int __stdcall MotionV4000_zero(int iAixs)	15
22. MotionV4000_API int __stdcall MotionV4000_get_pos(double *Pos)	15
23. MotionV4000_API int __stdcall MotionV4000_get_aixs_rotation(int iAixs, bool &bRotation)	15
24. MotionV4000_API int __stdcall MotionV4000_set_fly(int iMode,int prePulse,int exposureTime)	15
25. MotionV4000_API int __stdcall MotionV4000_get_fly(int &iMode, int &prePulse, int &exposureTime)	16
26. MotionV4000_API int __stdcall MotionV4000_start_fly(int iAixs, int iTimes)	16
27. MotionV4000_API int __stdcall MotionV4000_get_fly_pos(double *pos)	16
28. MotionV4000_API int __stdcall MotionV4000_trim_fly(int iAixs)	16
五、 功能演示软件	19
1. 轴状态	19

2. 输入状态	19
3. 信息提示	19
4. 运动控制	20
5. IO 控制(光源、继电器输出).....	20
6. 飞拍控制(VM4000Mini 无此功能).....	20
7. 运动轴参数设置	20
8. 灯光等级设置	21
六、变更记录	22

一、概述

VM4000 系列控制器专注于闪测仪行业，具备 3-6 轴运动控制、10-16 路输入、8-19 路输出（光源输出）的专用控制器，配置 12V 与 5V 输出，主要用于机器视觉项目，使用简单可靠。

1. 功能及特点

- 先进的上下位机结构形式，高性能 32 位浮点 ARM 双核处理器
- 采用 24VDC 电源输入，12V 输出为 CCD 供电，5V 输出为小型光源供电与光栅尺供电
- 8-16 路通用输入，可连接各种开关按钮、传感器，与 3-6 轴运动控制限位共用
- 8-19 路通用输出，输出具备输出电流大小可调节功能，可用于 LED 光源亮度调节,可用作 IO 输出
- 光源控制具备硬件亮度校准功能（亮度等级分 1-16 可设置，用于补偿光衰等光源特性差异），支持多通道并联为更大功率光源供电
- 支持 XP/win7/win10/win11 操作系统，32/64 位均可，使用网口直接通讯，软件采用最先进的通讯方法，更加稳定可靠，具备断线自动重连功能，单个 API 函数执行速度快（0.1-1ms）
- 具备金属外壳保护，强固可靠，抗干扰性强
- 具备飞拍功能，可设置飞拍脉冲补偿或根据相机曝光时间自动补偿
- 具备回原点偏移，运动轴软限位功能
- 速度最低为 0.1mm/s
- 控制器级全闭环控制功能，根据光学尺位置进行定位控制，采用更先进的 PID 算法，比开环、半闭环控制定位精度更高、速度更快
- Pro 的控制器光学尺接口 R/S 两种类型可选，无需转接线

2. 主要用途

- 闪测仪、一键测量仪、影像仪、机械手、全闭环控制系统、自动化、运动控制与光源控制
- 建议立式闪测、卧式闪测用 VM4000Mini,立卧一体，小型拼接用 VM4000，大型拼接用 VM4000Pro（差分长线距离抗干扰更强、更稳定，不容易丢脉冲）

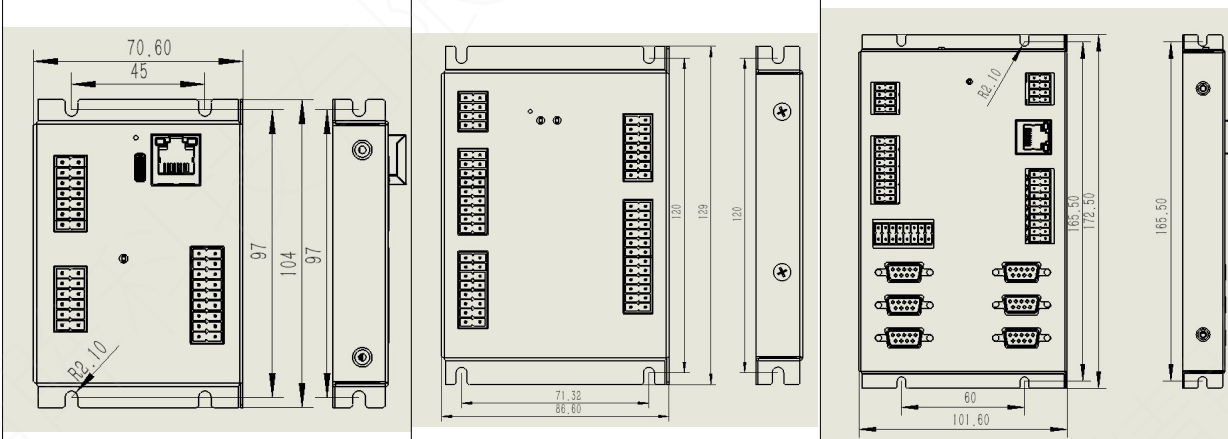
3. 接线管型端子针型接线端子

- 0.3mm² 以下的线用 E0310,两根 0.3mm² 以下的并线用 E0512
- 0.3-0.5mm² 的线用 E0512
- 插入不用按压，接线便捷

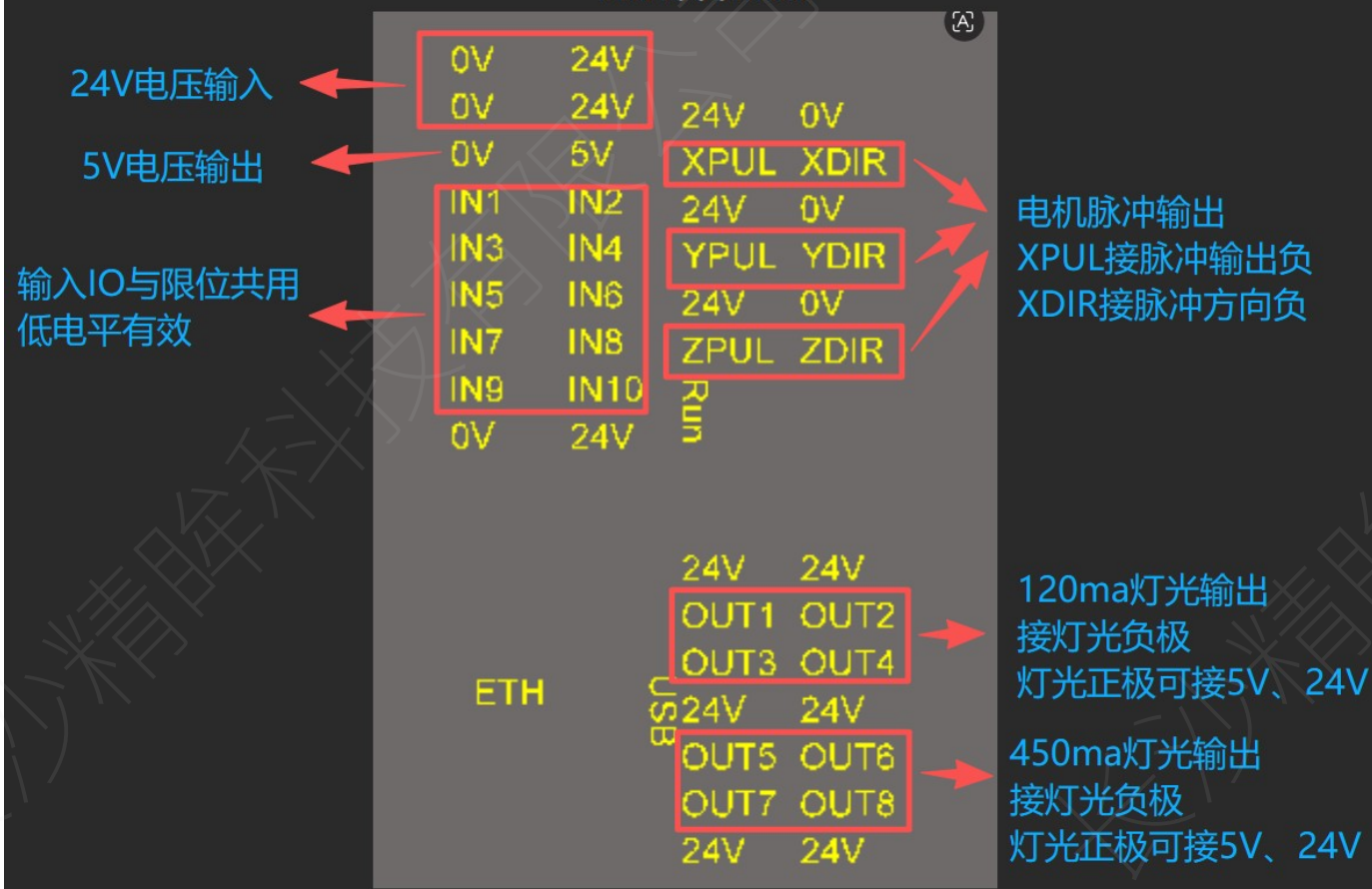
如下：



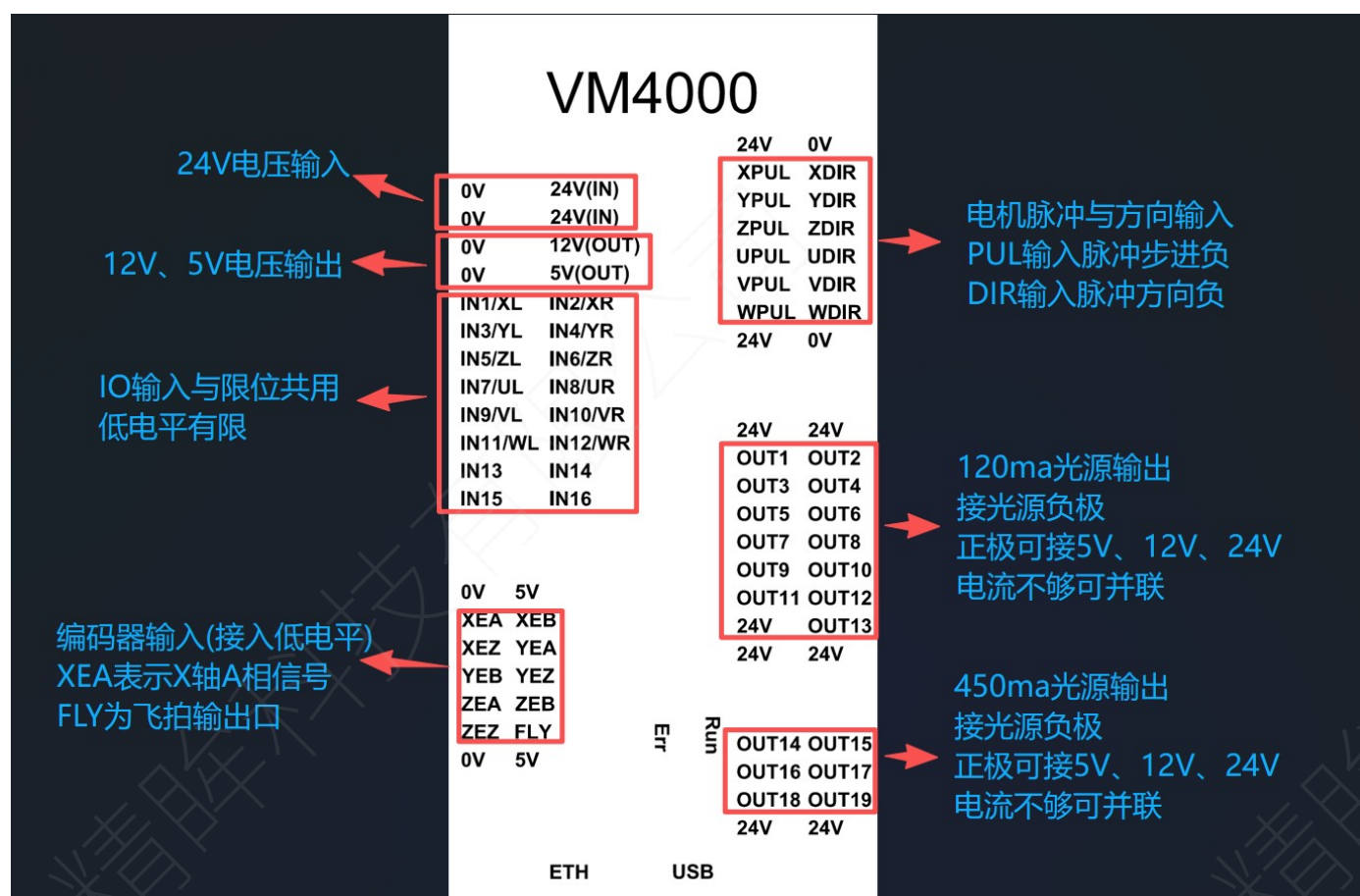
二、控制器类型

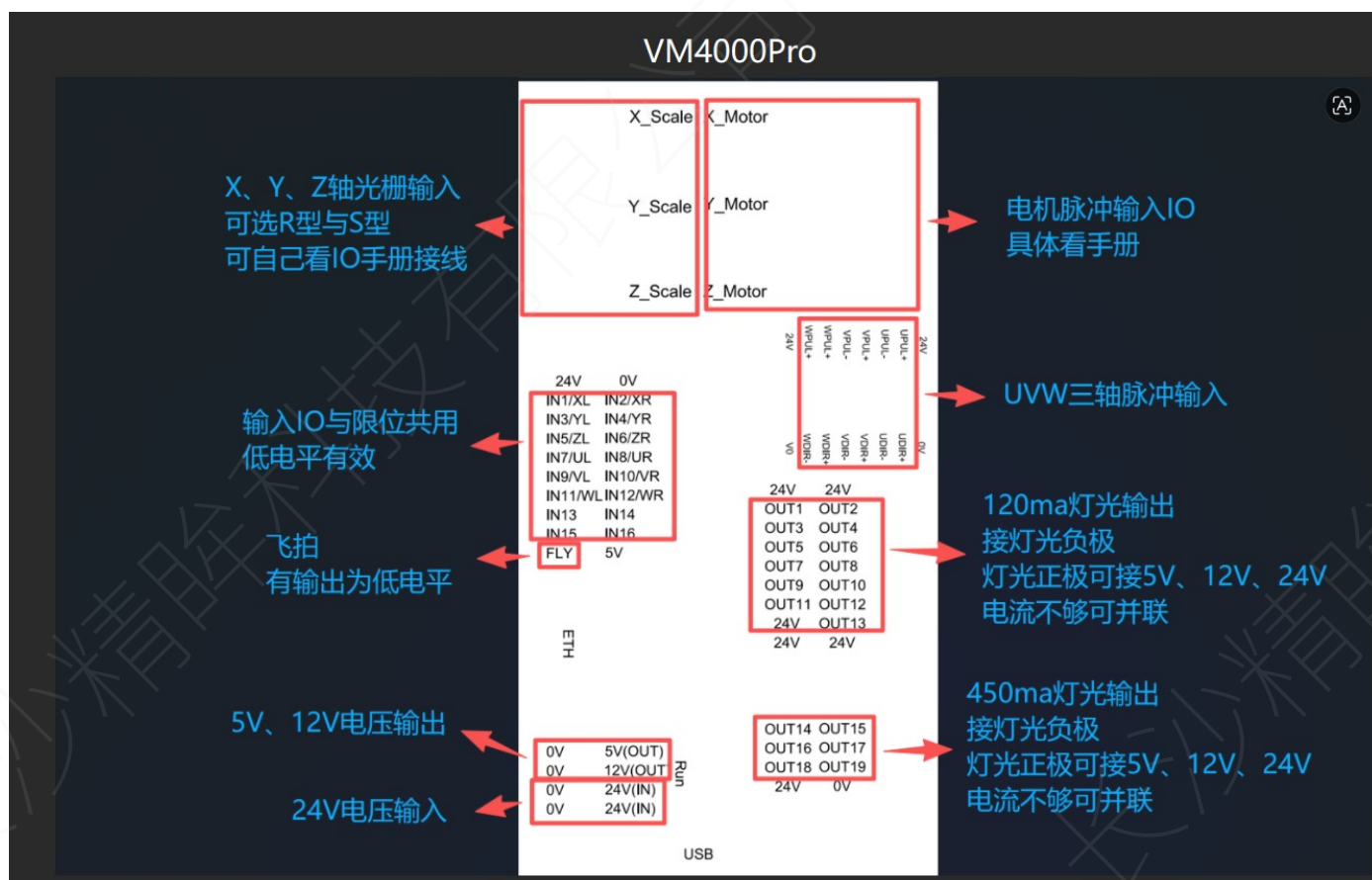
	VM4000Mini	VM4000	VM4000Pro
输入电源	+24VDC±5%	+24VDC±5%	+24VDC±5%
输出电源	+5VDC±5% @2.0A(max)	+12VDC±5%@2.0A(max) +5VDC±5% @2.0A(max)	+12VDC±5%@2.0A(max) +5VDC±5% @2.0A(max)
通讯	网口(0.5-1ms)	网口(0.5-1ms)	网口(0.5-1ms)
调用 API 时间	获取、输出 IO 调用无限制，光源更新（5ms），其它指令调用（1ms），无线程限制		
轴数	3 轴	6 轴	6 轴
闭环方式	无	全闭环(PID)	全闭环(PID)
飞拍	不支持	支持（单轴）	支持（单轴）
光栅接口	0 路	3 路（ABZ）	3 路（ABZ）
光栅频率	0 路	3 路(0-5MHz)	3 路(0-20MHz)(差分)
脉冲频率	400K(Max)	400K (Max)	400K(Max)
脉冲输出方式	脉冲+方向	脉冲+方向	脉冲+方向(差分)
输入	10 路(包括 3 轴限位)	16 路(包括 6 轴限位)	16 路(包括 6 轴限位)
输出	8 路(4 路 120ma,4 路 450ma)	19 路(13 路 120ma,6 路 450ma)	19 路(13 路 120ma,6 路 450ma)
工作条件	0-40℃ 无凝露、无油污		
存储环境	-20~80℃ 无凝露、无油污		
尺寸	 The technical drawings show the physical dimensions of the three controller models. VM4000Mini has a width of 70.60mm and a height of 104mm. VM4000 has a width of 71.32mm and a height of 120mm. VM4000Pro has a width of 101.60mm and a height of 165.50mm. All models have a depth of 45mm. The drawings also show the location of various ports and connectors on the front panel.		

VM4000Mini



VM4000





1. 电脑 IP 设置:

☒ 使用下面的 IP 地址(S):

IP 地址(I):

子网掩码(U):

默认网关(D):

☐ 自动获得 DNS 服务器地址(B)

☒ 使用下面的 DNS 服务器地址(E):

首选 DNS 服务器(P):

备用 DNS 服务器(A):

- 控制器 IP: 10.10.0.254
- 电脑建议 IP: 10.10.0.65 (65 可改为 1~250)
- 子网掩码: 255.0.0.0
- 上位机最多支持 8 个同时连接
- 如果连接不上, 用控制台 cmd 命令可以 ping 10.10.0.254 看是否连通, 如果一直没有连通, 可更改网卡设置, 如下:



2. 光源控制:

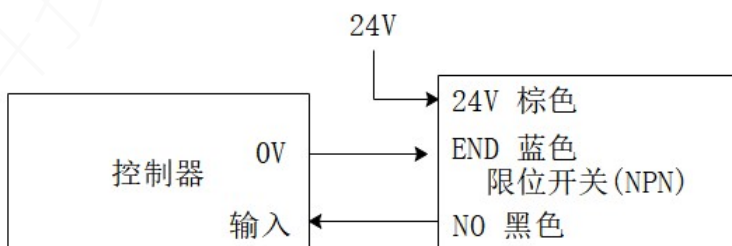
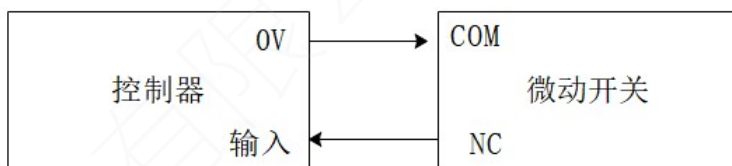
- 采用恒流驱动方式，所有通道均为集电极开路恒流源输出；如果使用大功率光源，则可以将同等 电流通道的 Light 输出口并联使用。
- 电压可选择 5V、12V、24V，只需要光源正极接 5V、12V、24V 正极，负极根据电流大小接光源输出接口即可。
- 并联接法，可将光源输出可并到一起，再接光源负极（功率计算如下）

如：使用 24V 光源，1A 光源

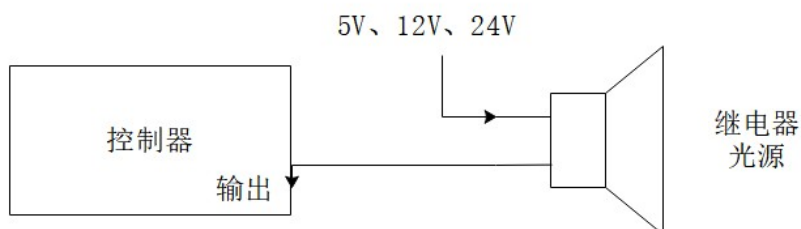
可以接正极接到 24V，将 Out18、Out19、Out1 用线连接后接光源负极（功率为： $450\text{mA}+450\text{mA}+120\text{mA}=1020\text{mA}$ ）

三、通用输入和通用输出示意图

1. 通用输入信号

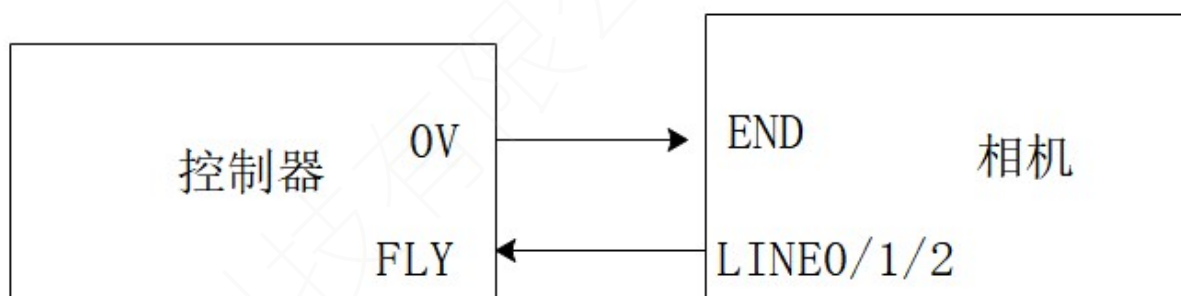


2. 通用输出信号

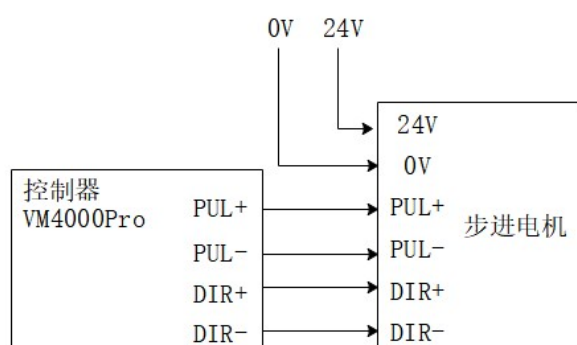
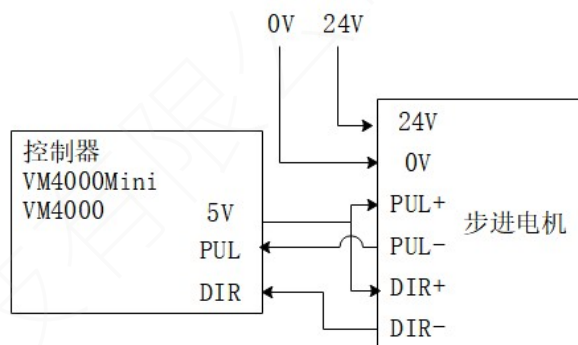


3. 飞拍输出接线(VM4000、VM4000Pro)

- 相机与控制器必须共地线，具体看相机接口定义



4. 连接步进电机示意图



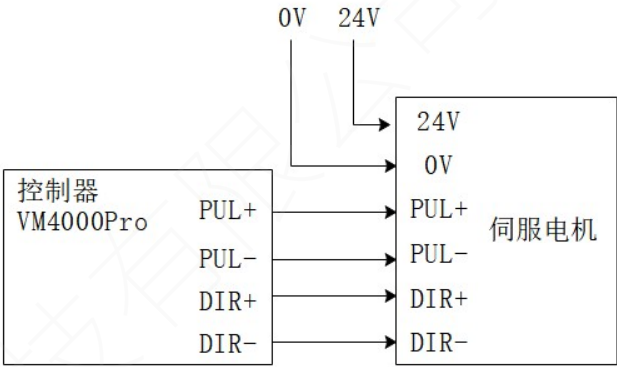
5. 连接伺服电机示意图(VM4000、VM4000Mini)

- 注 1：请查看伺服手册，共阳极接法（与步进电机接法不同）
- 注 2：控制器如果跟步进电机一样接可能烧坏伺服驱动器

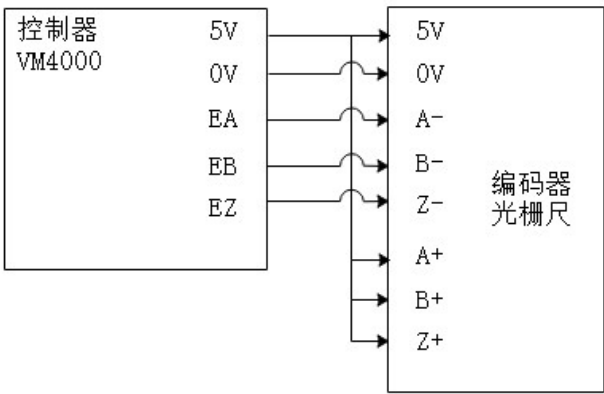
连接伺服电机示意图(VM4000Pro)

X/Y/Z Motor 接口引脚定义		
引脚编号	引脚定义	备注
1	PUL+	脉冲输出+
2	PUL-	脉冲输出-
3	DIR+	方向输出+
4	DIR-	方向输出-
5	0V	电源 0V
6	空	空脚
7	空	空脚
8	空	空脚
9	24V	电源 24V

U/V/W 轴接口示意图



6. X/Y/Z 编码器连接示意图(VM4000)



X/Y/Z 编码器接口定义(VM4000Pro)

X/Y/Z Scale 光栅尺 S 型引脚定义 万濠、长春七海光电 TTL 电平 或者 RS422 电平		
引脚编号	引脚定义	备注
1	A-	A 相负
2	0V	0V
3	B-	B 相负
4	空	
5	RI-	RI 相负
6	A+	A 相正
7	+5V 输出	5V 电源
8	B+	B 相正
9	RI+	RI 相正

X/Y/Z Scale 光栅尺 R 型引脚定义 新天、信和 TTL 电平或者 RS422 电平		
引脚编号	引脚定义	备注
1	+5V 输出	5V 电源
2	0V	0V
3	A+	A 相正
4	B+	B 相正
5	RI+	RI 相正
6	RI-	RI 相负
7	A-	A 相负
8	B-	B 相负
9	空	

四、api 函数说明 (C++,三个类型控制器通用)

```
#include "MotionV4000.h"
```

```
#pragma comment(lib, "MotionV4000.lib")
```

MotionV4000.lib 与 MotionV4000.dll 放到 exe 运行目录

1. MotionV4000_API int __stdcall MotionV4000_initial()

功能: 初始化网口并连接到 VM4000 控制器; 该函数必须先于其他函数调用, 否则其他函数得不到成功执行, 掉线软件具有自动连接功能, 无须外部再初始化

返回参数: 1 为连接成功, 其它值连接失败

2. MotionV4000_API int __stdcall MotionV4000_is_connect()

功能: 是否控制器已经连接

返回参数: 1 为连接, 其它值断开

3. MotionV4000_API int __stdcall MotionV4000_get_Type()

功能: 得到控制器的类型

返回参数: 0-失败 1-VM4000Mini 2-VM4000 3-VM4000Pro

4. MotionV4000_API int __stdcall MotionV4000_close()

功能: 关闭控制器连接, 电源没有关闭, 输出端口状态以及光源亮度保持不变

返回参数: 0-失败 1-成功

5. MotionV4000_API int __stdcall MotionV4000_get_input_bit(int id,bool &status)

功能: 按输入端口编号读取输入口状态

入口参数:

id: 输入口编号:1-24 表示 In1~In24(与轴限位共用)

Status: 1-接通, 0-断开

返回参数: 1 为成功, 其它值失败

6. MotionV4000_API int __stdcall MotionV4000_get_input_bit(bool *status)

功能: 一次读取所有输入口状态

入口参数:

Status: 传入数组 status[24], 1-接通, 0-断开

返回参数: 1 为成功, 其它值失败

如: bool status[24];MotionV4000_get_input_bit(status); //0-23 对应 In1~In24 的状态

7. MotionV4000_API int __stdcall MotionV4000_set_output(int id, bool status)

功能: 按通用输出端口编号设置输出口的状态, 通用输出口用于开/关指示灯、继电器线圈等负载 控制器通电后, 默认输出口无效 (对 0V 断开)

入口参数:

id: 输出口编号:1-19;表示 Out1~Out19

Status: 1-接通,0-断开

返回参数: 1 为成功, 其它值失败

8. MotionV4000_API int __stdcall MotionV4000_set_output(int id, int status)

功能: 按通用输出端口编号设置输出口的状态, 通用输出口用于调节灯光

入口参数:

id: 输出口编号:1-19;表示 Out1~Out19

Status: 设置为 0-255 灯光亮度(等级可以通过 Demo 设置)

返回参数: 1 为成功, 其它值失败

9. MotionV4000_API int __stdcall MotionV4000_set_outputAll(bool status)

功能: 输出端口全部打开或关闭

入口参数:

Status: 1-接通,0-断开

返回参数: 1 为成功, 其它值失败

10. MotionV4000_API int __stdcall MotionV4000_get_move_state(int iAixs, bool &status)

功能: 得到轴是否移动结束

入口参数:

iAixs: 0-5,0 表示第一轴

Status: 1-停止, 0-运动

返回参数: 1 为成功, 其它值失败

11. MotionV4000_API int __stdcall MotionV4000_get_home_state(int iAixs, bool &bHome)

功能: 得到轴是否已经回原点

入口参数:

iAixs: 0-5,0 表示第一轴

bHome: 1-回原点完成, 0-没有回原点

返回参数: 1 为成功, 其它值失败

12. MotionV4000_API int __stdcall MotionV4000_get_leftRight_softLimit(int iAixs, double &leftLimit, double &rightLimit)

功能: 得到指定轴的软限位

入口参数:

iAixs: 0-5,0 表示第一轴

leftLimit: 得到负限位位置(包括软限位减速距离)

rightLimit: 得到正限位位置(包括软限位减速距离)

返回参数: 1 为成功, 其它值失败

13. MotionV4000_API int __stdcall MotionV4000_go_home(int iAixs, float speed=0)

功能: 指定轴回原点

入口参数: iAixs: 0-5, 0 表示第一轴

speed: 回零速度设置 例: speed=0 以 Demo 软件设置的速度回原点; speed=10mm/s; 以 10mm/s 的速度回原点 (原点偏移功能: 回原点后会移动到偏移位置再次归 0, 偏移为正就往右限位移动后归 0, 偏移为负就往负限位移动后归 0)

返回参数: 1 为成功, 其它值失败

14. MotionV4000_API int __stdcall MotionV4000_set_speed(int iAixs, float speed)

功能: 得到指定轴回原点速度

入口参数: iAixs: 0-5, 0 表示第一轴

speed: 设置轴的速度 mm/s (最小 0.1mm/s), 大于最大速度以最大速度运行

返回参数: 1 为成功, 其它值失败

15. MotionV4000_API int __stdcall MotionV4000_get_max_speed(int iAixs, int &speed)

功能: 得到指定轴回原点速度

入口参数: iAixs: 0-5, 0 表示第一轴

speed: 得到轴的最大速度 mm/s (Demo 设置的最大速度)

返回参数: 1 为成功, 其它值失败

16. MotionV4000_API int __stdcall MotionV4000_get_acc_time(int iAixs, double &time)

功能: 得到指定轴运动加减速时间

入口参数: iAixs: 0-5, 0 表示第一轴

time: 得到加减速时间, 以秒为单位 如: 0.1 表示加减速时间为 0.1 秒

返回参数: 1 为成功, 其它值失败

17. MotionV4000_API int __stdcall MotionV4000_single_move_to(int iAixs, double pos)

功能: 指定轴按照指定的速度移动到指定位置 (速度可以通过 MotionV4000_set_speed 函数设置速度, 没回原点最大速度 100mm/s)

入口参数: iAixs: 0-5, 0 表示第一轴

pos: 目标位置

返回参数: 1 为成功, 其失败

18. MotionV4000_API int __stdcall MotionV4000_xyz_move_to(double posX, double posY, double posz)

功能: 指定轴按照各轴的位置直线插补运行到目标位置 (速度可以通过 MotionV4000_set_speed 函数设置速度, 以距离最远轴速度为最大速度)

入口参数:

posx: X 轴目标位置

posy: Y 轴目标位置

posz: Z 轴目标位置

返回参数: 1 为成功, 其它值失败

19. MotionV4000_API int __stdcall MotionV4000_jog_move(int iAixs, bool dir)

功能: JOG 运动是一种用于指定运动方向和运动速度但不指定目标位置的运动方式(到达软限位减速停止, 到达限位立即停止, 没回原点最大速度 100mm/s)

入口参数:

iAixs: 0-5, 0 表示第一轴

dir: 0-表示负限位方向运动, 1-正限位方向运动 (以当前运动速度运行)

返回参数: 1 为成功, 其它值失败

20. MotionV4000_API int __stdcall MotionV4000_stop(int iAixs, int iMode=0)

功能: 停止所有轴运动

入口参数:

iAixs: 0-5, 0 表示第一轴

iMode: 0-减速停止, 1-立即停止

返回参数: 1 为成功, 其它值失败

21. MotionV4000_API int __stdcall MotionV4000_zero(int iAixs)

功能: 指定轴编码器和指令计数清零

入口参数:

iAixs: 0-5, 0 表示第一轴

返回参数: 1 为成功, 其它值失败

22. MotionV4000_API int __stdcall MotionV4000_get_pos(double *Pos)

功能: 获取指定轴读数, 开环以脉冲计算, 闭环以光栅尺当量计算

入口参数:

Pos: 得到 6 个轴当前位置

返回参数: 1 为成功, 其它值失败

如 double posTarget[6], MotionV4000_get_pos(posTarget), 返回对应轴的坐标

23. MotionV4000_API int __stdcall MotionV4000_get_aixs_rotation(int iAixs, bool &bRotation)

功能: 判断指定轴是否为旋转轴 (旋转轴可以以限位回原点, 移动的时候不受限位控制)

入口参数:

iAixs: 0-5, 0 表示第一轴

bRotation: 1-旋转轴 0-不是旋转轴

返回参数: 1 为成功, 其它值失败

24. MotionV4000_API int __stdcall MotionV4000_set_fly(int iMode, int prePulse, int exposureTime)

功能: 设置飞拍补偿参数, 改变的时候只要设置一次, 可以 Demo 设置

入口参数:

iMode: 0-不使用 1-提前补偿 2-曝光补偿

prePulse: 提前多少个当量触发 (可为正负)

exposureTime: 设置相机曝光时间, 软件自动计算

返回参数: 1 为成功, 其它值失败

25. MotionV4000_API int __stdcall MotionV4000_get_fly(int &iMode, int &prePulse, int &exposureTime)

功能：得到飞拍补偿参数

入口参数：

iMode：得到模式 0-不使用 1-提前补偿 2-曝光补偿

prePulse：得到提前多少个当量触发（可为正负）

exposureTime:得到设置相机曝光时间，软件自动计算

返回参数：1 为成功，其它值失败

26. MotionV4000_API int __stdcall MotionV4000_start_fly(int iAixs, int iTimes)

功能：开始飞拍，然后运动到目标位置，就会触发 iTimes 次

入口参数：

iAixs：0-5，0 表示 X 轴

iTimes：运动到目标位置触发次数, 最小为 2（包括开始与结束 2 次）

返回参数：1 为成功，其它值失败

27. MotionV4000_API int __stdcall MotionV4000_get_fly_pos(double *pos)

功能：开得到飞拍结束后的坐标（运动完成后才会有数据）

入口参数：

pos：得到飞拍位置，传入数组（最多 100 次触发）

返回参数：大于 0，表示触发次数

如：

```
double posTrim[100];
```

```
int iTimes=MotionV4000_get_pos(posTarget);
```

posTarget 会返回 MotionV4000_start_fly 开始飞拍触发的坐标 iTimes 返回次数

28. MotionV4000_API int __stdcall MotionV4000_trim_fly(int iAixs)

功能：手动触发飞拍

入口参数：

iAixs：0-5，0 表示 X 轴

返回参数：1 为成功，其它值失败

例子:

```
//初始化
```

```
if (!MotionV4000_initial()) //初始化控制器, 如果控制器网线断开会自动连接
{
    printf("初始化控制器失败\n");
    return;
}
```

```
//得到输入状态
```

```
bool bInput[16] = { 0 };
for (int i = 0; i < 16; i++) //循环得到输入状态
    MotionV4000_get_input_bit(i, bInput[i]);
```

```
MotionV4000_get_input_bit(bInput); //一次得到所有输入状态
```

```
//设置输出状态
```

```
for (int i = 0; i < 24; i++)
    MotionV4000_set_output(i, false); //关闭所有输出
```

```
for (int i = 0; i < 24; i++)
    MotionV4000_set_output(i, true); //打开所有输出
```

```
MotionV4000_set_outputAll(false); //关闭所有输出
```

```
MotionV4000_set_outputAll(true); //打开所有输出
```

```
MotionV4000_set_output(1, 128); //PWM 输出 1 为 128, 灯光调节
```

```
//轴运动
```

```
MotionV4000_set_speed(0, 100); //设置 X 轴的速度为 100mm/s
MotionV4000_single_move_to(0, 100); //X 轴移动绝对位置 100mm
bool bRunState = 0;
MotionV4000_get_move_state(0, bRunState); //得到轴移动状态
while (!bRunState) //判断轴是否运动
    MotionV4000_get_move_state(0, bRunState);
```

```
MotionV4000_single_move_to(0, 200); //X 轴移动绝对位置 200mm
bRunState = 0;
MotionV4000_get_move_state(0, bRunState); //得到轴移动状态
while (!bRunState) //判断轴是否运动
    MotionV4000_get_move_state(0, bRunState);
```

```
//轴手动移动
MotionV4000_set_speed(0, 10); //设置 X 轴的速度为 10mm/s
MotionV4000_jog_move(0, 1); //鼠标按下后调用，轴往正限位以 10mm/s 的速度移动
MotionV4000_stop(0, 0); //鼠标松开，停止轴移动, 减速停止

//得到轴的坐标位置
double fPos[6];
MotionV4000_get_pos(fPos); //得到 6 个轴的坐标，开环以脉冲计算，闭环以光栅尺计算

//得到轴回原点状态
bool bHome[6] = { 0 };
for (int i = 0; i < 6; i++)
    MotionV4000_get_home_state(i, bHome[i]); //得到当前轴的回原点状态，1-回原点完成
0-没有回考点或回原点中

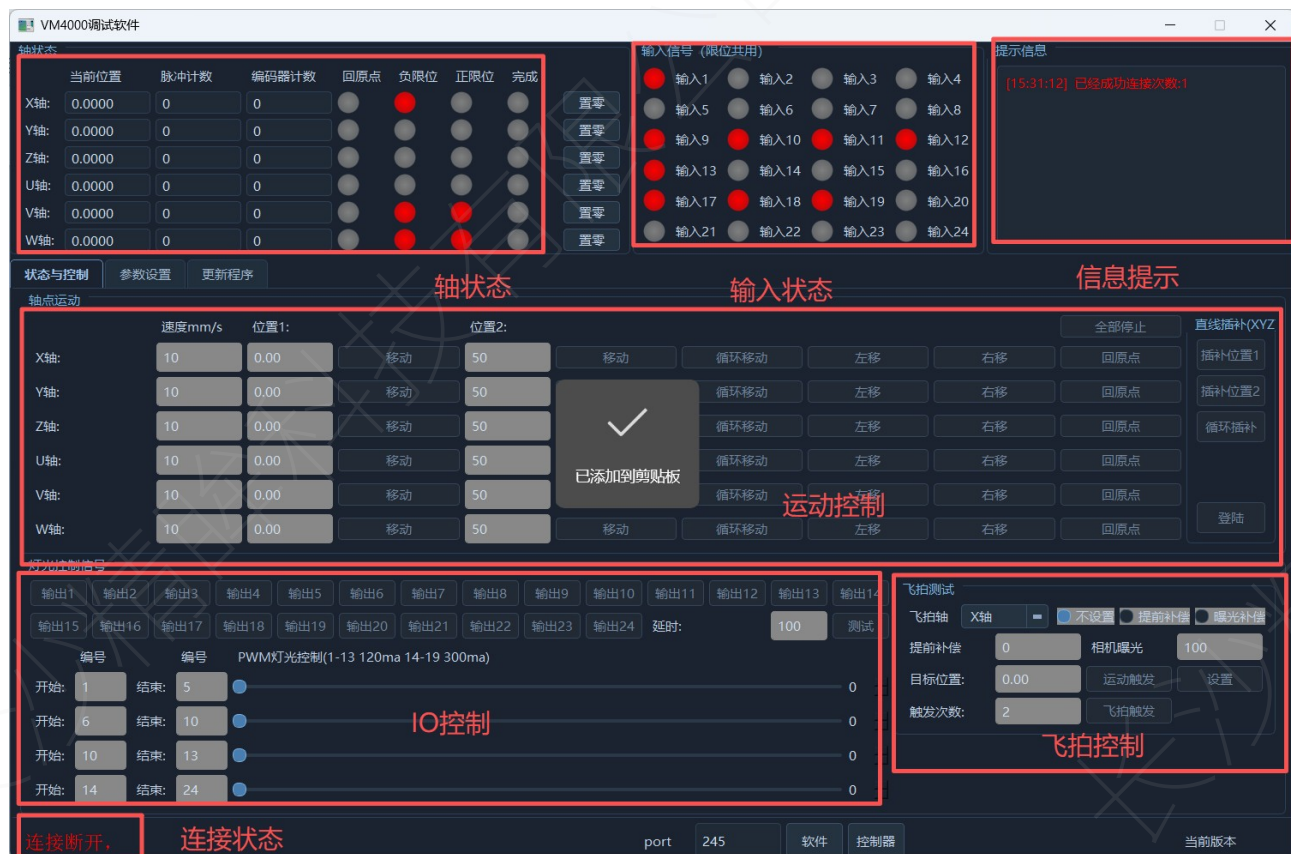
//轴飞逝
MotionV4000_start_fly(0, 3); //X 轴设置飞拍次数为 3，自动清除之前飞拍的数据

MotionV4000_set_speed(0, 100); //设置飞拍速度为 100mm/s
MotionV4000_single_move_to(0, 100); //X 轴移动绝对位置 100mm

/* 得到轴移动状态可以不用*/
bool bRunState = 0;
MotionV4000_get_move_state(0, bRunState); //得到轴移动状态
while (!bRunState) //判断轴是否运动
    MotionV4000_get_move_state(0, bRunState);

double fFlyPos[3]; //存放飞拍的坐标，开环为脉冲计算的坐标 闭环为光栅尺计算的坐标
```

五、功能演示软件



- 主界面有轴状态、输入状态、信息提示、运动控制、IO 控制、飞拍控制、连接状态、参数设置、程序更新

1. 轴状态：

分别显示当前位置，脉冲计数、编码器计数、回原点标记、负限位、正限位、运动状态、置零(标记亮起表示已经触发)

当前位置：mm 为单位，当前轴的位置（开环根据脉冲计算，闭环根据光栅尺计算）

回原点标记：回原点成功后该标记亮起，应用软件关闭不影响该标记，电源重新启动该标记自动清除。

运动状态：轴处于停止状态，该标记亮起；任何模式运动状态，该标记熄灭

正负限位：正、负限位的状态，限位开关触发时，该标记置位亮起

编码器计数与指令计数：脉冲当量与光栅尺最小分辨率

置零：脉冲计数、编码器计数重置为 0

2. 输入状态：

显示输入信号的状态与轴限位共用，限位开关或有输入信号触发时，该标记置位亮起。

3. 信息提示：

显示编码器 Z 触发信息、程序升级信息、程序报错信息、软件连接信息、飞拍触发次数等等。

4. 运动控制:

速度 : 当前轴移动的速度

移动 : 当前轴移动到位置 1 或位置 2

循环移动 : 当前轴在位置 1 与位置 2 来回移动

左移 : jog 往负限位方向移动, 到软限位减速停止, 到限位立即停止

右移 : jog 往正限位方向移动, 到软限位减速停止, 到限位立即停止

回原点 : 开环移动到限位后, 反向慢速离开限位, 然后停止, 置 0

闭环移动到限位后, 反向找光栅 RI 信号, 再反向慢速离开, 然后停止, 置 0

注: 原点偏移, 会再次移动到偏移距离, 再置 0

全部停止 : 会停止所有轴运动, 如果轴循环运动与输出测试, 也会停止

插补位置 1 : XYZ 轴会直线插补移动到位置 1

插补位置 2 : XYZ 轴会直线插补移动到位置 2

循环插补 : XYZ 轴会直线插补移动到位置 1, 然后插补到位置 2, 来回循环

登陆 : 只有登陆后才后设置控制器参数(密码:vm4000i)

注: 所有运动都可以通过(全部停止)取消运动

5. IO 控制(光源、继电器输出):

输出 x : 输出接通与断开

PWM 输出: 输出 IO 可以通过编号开始与结束设定, 编号间的输出光源亮度可通过滑块调节

(VM4000Mini:5-8 为大电流 其它:13-19 为大电流)

6. 飞拍控制(VM4000Mini 无此功能):

说明 : 此飞拍为单轴, 不能多轴飞拍

飞拍轴 : X/Y/Z/U/V/W 均可设置飞拍, 每次只能选择一个轴飞拍

不设置 : 不设置飞拍补偿

提前补偿: 可输入正负号, 为负时当没到达目标位置就会触发, 为正时到达目标位置后才触发

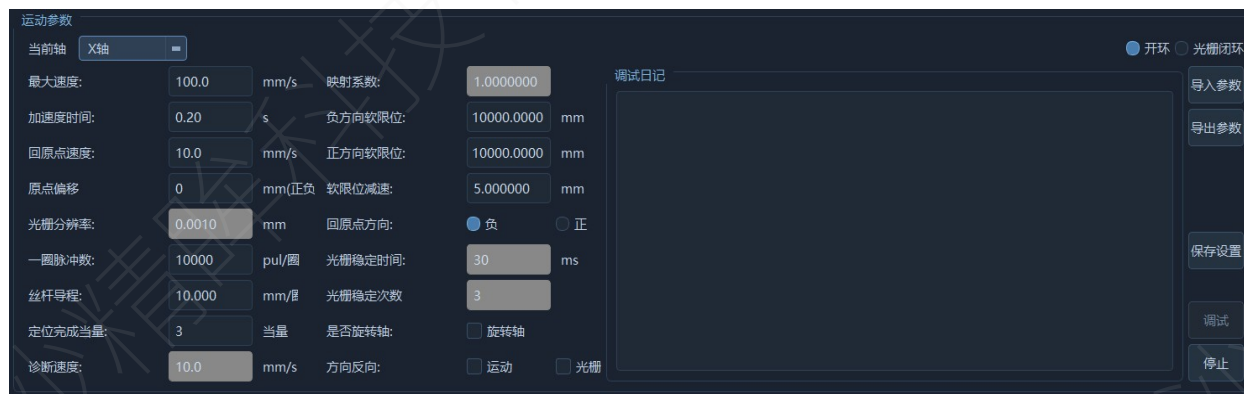
相机曝光: 可输入相机的曝光进行自动补偿, 由于相机曝光与速度的匹配会造成精度误差, 所以可以自动补偿

运动触发: 选择飞拍轴、触发次数、目标位置, 轴会运动到目标位置, 完成触发

飞拍触发: 手动触发

设置 : 补偿模式、提前补偿、相机曝光参数保存

7. 运动轴参数设置:



- 最大速度** : 设置当前轴运动的最大速度
- 加速度时间** : 设置当前轴加速度的时间
- 回原点速度** : 设置当前轴回原点的速度, API 回原点速度设置为 0 就以这个速度回原点
- 原点偏移** : 回原点后再次移动偏移距离, 轴置 0
- 光栅分辨率** : 光栅的最小当量, 1um 光栅就填写 0.001
- 一圈脉冲数** : 步进驱动器及伺服驱动器旋转一圈的脉冲数
- 丝杆导程** : 电机旋转一圈前进的距离 mm
- 定位完成当量** : 闭环运动中光栅尺停止的误差, 单位是光栅尺分辨率个数, 0.001 分辨率 3 表示 0.003mm
- 映射系数** : 闭环 调试后自动计算出来
- 负方向软限位** : 开环 可等待回原点后, 手动移动到负限位的位置
闭环 调试后自动计算出来
- 正方向软限位** : 开环 可等待回原点后, 手动移动到正限位的位置
闭环 调试后自动计算出来
- 注: 开环回原点, 请先将原点偏移设置成 0, 回原点后再次设置软限位, 最后再设置原点偏移**
- 软限位减速** : 软限位偏移, 移动到软限位偏移多少 mm 的时候减速停止
- 回原点方向** : 可以设置轴往负限位或正限位回原点
- 光栅稳定时间**: 闭环 控制器发完脉冲给伺服控制器后, 可能伺服还没有执行完, 等待执行完成的时间 (可调试伺服加快响应)
- 光栅稳定次数**: 等待光栅稳定时间后, 控制器判断是否到了目标位置, 没有到达就再次调节, 次数加 1, 到达设定次数就不在调整
- 是否旋转轴** : 驱动目标是否是 0-360 旋转运动, 没有转换成直线运动
- 方向反向** : 运动方向与光栅方向要一致, 如果不对要设置成一致
- 导入参数** : 从 ini 文件恢复参数到控制器
- 导出参数** : 把控制器参数保存到硬盘 ini 文件
- 保存设置** : 保存当前轴的参数到控制器
- 调试** : 闭环 调试当前轴
- 停止** : 停止当前轴调试

8. 灯光等级设置:

软件内部是 0-4096, 每一等级是 256, 16 级就是 4096, 等级 1 就是 0-255, 相当电流缩小到 1/16

输出亮度等级设置 (1到16个等级可调)															
输出1	16	输出2	16	输出3	16	输出4	16	输出5	16	输出6	16	输出7	16	输出8	16
输出9	16	输出10	16	输出11	16	输出12	16	输出13	16	输出14	16	输出15	16	输出16	16
输出17	16	输出18	16	输出19	16	输出20	16	输出21	16	输出22	16	输出23	16	输出24	16
															保存设置

保存设置: 修改当前输出的灯光等级(1-16), 保存到控制器

六、变更记录

变更记录	修改内容
2026061809	初始版本