leetro
专业运动控制

P2 系列　独立脉冲式运动控制器
编程手册

目录

版权申明

	通电或正在工作中的机器有危险！ 使用者有责任在及其中设计应对型的出错处理及安全保护机制，乐创技术没有义务和责任对由此造成的、附带的或相应产生的损失负责。

联系方式

官方网站：**http://www.leetro.com**

微信公众号：cdleetro

服务热线：400-990-0289

技术支持：support**@leetro.com**

总部研发：**成都市高新区科园南二路 1 号大一孵化园 8 幢 B 座**

东莞销售：东莞市松山湖园区科技四路 2 号御豪轩大厦 1 栋 610

苏州销售：苏州市高新区狮山路 28 号苏州高新广场 1102

更新记录

版本号	修订日期	更新要点
V1.0	2022 年 11 月 29 日	新建

第 1 章 指令列表

- 本手册中所有字体为蓝色的下划线字体（如 init_api）均带有链接功能，点击可跳转至具体说明。
- 本手册的指令全部为单卡库的指令说明，当工控机连接多张控制卡时，请调用多卡库指令，多卡库指令相比单卡库指令仅多出一个参数 idMc（控制卡卡号），用于控制相应的控制卡。控制卡卡号确定方法：参见第 8 章 8.2.1 多卡连接。
- 本手册中所有调用示例代码均在 C++环境下调试通过

第 2 章 系统配置	
2.2.1 控制器初始化	
init_api	初始化运动控制器
2.2.2 轴配置	
set_channel_work_axis_group	配置工作轴组
set_axis_private_output_level	配置轴专用输出口
set_axis_private_input_level	配置轴专用输入口
set_axis_pulse_per_mm	配置轴脉冲当量
set_axis_max_speed	配置轴最大速度
set_axis_jerk_speed	配置轴许可跳变速度
enable_axis_servo_on	配置轴伺服使能
set_axis_encode_param	配置轴编码器反馈参数
set_pulse_abs_pos	配置轴脉冲位置
set_encode_abs_pos	配置编码器反馈位置
2.2.3 拓展轴配置	
enable_exaxis	配置拓展轴使能
set_exaxis_channel	配置拓展轴到关联通道中
第 3 章 状态检测	
get_mc_all_states	获取控制器所有状态
api_get_lib_version	获取当前动态链接库的版本
open_api_record	生成文本获取调用函数的记录
第 4 章 I/O 控制	
4.2.1 通用输出口控制	
outport_bit	控制单路输出口输出
outport_byte	控制多路输出口输出
4.2.2 通用输入口有效电平设置	
set_input_level	配置通用输入口的有效电平
第 5 章 轴运动控制	
5.2.1 运动指令立即模式&批处理模式	
switch_cmd_type	切换通道指令输出模式
begin_package_bat_cmd	批处理打包指令开始

end_package_bat_cmd	结束指令打包，将缓冲中的数据发送到控制器中
5.2.2 回零运动	
set_profile	配置轴回零运动和点位运动的速度
set_profile_s	配置轴回零运动和点位运动的 S 型速度速度
enable_axis_pulse_out	配置轴输出脉冲
fast_hmove	控制单轴回零运动
fast_hmove_multi	控制多轴回零运动
fast_hmove_by_z_encoder	控制单轴 Z 脉冲回零
fast_hmove_multi_by_z_encoder	控制多轴 Z 脉冲回零
5.2.3 点位运动	
fast_pmove	相对位置模式下的单轴点位运动（T 型速度曲线）
fast_pmove2	相对位置模式下的两轴点位运动（T 型速度曲线）
fast_pmove3	相对位置模式下的三轴点位运动（T 型速度曲线）
fast_pmove4	相对位置模式下的四轴点位运动（T 型速度曲线）
fast_pmove5	相对位置模式下的五轴点位运动（T 型速度曲线）
fast_pmove_s	相对位置模式下的单轴点位运动（S 型速度曲线）
fast_pmove2_s	相对位置模式下的两轴点位运动（S 型速度曲线）
fast_pmove3_s	相对位置模式下的三轴点位运动（S 型速度曲线）
fast_pmove4_s	相对位置模式下的四轴点位运动（S 型速度曲线）
fast_pmove5_s	相对位置模式下的五轴点位运动（S 型速度曲线）
fast_abs_pmove	绝对位置模式下的单轴点位运动（T 型速度曲线）
fast_abs_pmove2	绝对位置模式下的两轴点位运动（T 型速度曲线）
fast_abs_pmove3	绝对位置模式下的三轴点位运动（T 型速度曲线）
fast_abs_pmove4	绝对位置模式下的四轴点位运动（T 型速度曲线）
fast_abs_pmove5	绝对位置模式下的五轴点位运动（T 型速度曲线）
fast_abs_pmove_s	绝对位置模式下的单轴点位运动（S 型速度曲线）
fast_abs_pmove2_s	绝对位置模式下的两轴点位运动（S 型速度曲线）
fast_abs_pmove3_s	绝对位置模式下的三轴点位运动（S 型速度曲线）
fast_abs_pmove4_s	绝对位置模式下的四轴点位运动（S 型速度曲线）
fast_abs_pmove5_s	绝对位置模式下的五轴点位运动（S 型速度曲线）
axis_decel_stop	控制轴缓停
axis_sudden_stop	控制轴急停
5.2.4 插补运动	
set_vector_profile	配置插补运动矢量 T 型速度曲线
set_vector_profile_s	配置插补运动矢量 S 型速度曲线
fast_line2	相对位置模式下的两轴直线插补运动（T 型速度曲线）
fast_line3	相对位置模式下的三轴直线插补运动（T 型速度曲线）
fast_line4	相对位置模式下的四轴直线插补运动（T 型速度曲线）
fast_line5	相对位置模式下的五轴直线插补运动（T 型速度曲线）
fast_line6	相对位置模式下的六轴直线插补运动（T 型速度曲线）
fast_line2_s	相对位置模式下的两轴直线插补运动（S 型速度曲线）
fast_line3_s	相对位置模式下的三轴直线插补运动（S 型速度曲线）
fast_line4_s	相对位置模式下的四轴直线插补运动（S 型速度曲线）

fast_line5_s	相对位置模式下的五轴直线插补运动（S 型速度曲线）
fast_abs_line2	绝对位置模式下的两轴直线插补运动（T 型速度曲线）
fast_abs_line3	绝对位置模式下的三轴直线插补运动（T 型速度曲线）
fast_abs_line4	绝对位置模式下的四轴直线插补运动（T 型速度曲线）
fast_abs_line5	绝对位置模式下的五轴直线插补运动（T 型速度曲线）
fast_abs_line2_s	绝对位置模式下的两轴直线插补运动（S 型速度曲线）
fast_abs_line3_s	绝对位置模式下的三轴直线插补运动（S 型速度曲线）
fast_abs_line4_s	绝对位置模式下的四轴直线插补运动（S 型速度曲线）
fast_abs_line5_s	绝对位置模式下的五轴直线插补运动（S 型速度曲线）
5.2.5 速度前瞻运动	
curve_begin	速度前瞻运动开始
curve_end	速度前瞻运动结束
set_arc_segment_dis	配置圆弧和整圆拆分精度
c_set_track_speed	设置前瞻运动矢量速度
c_change_track_speed	前瞻运动中轨迹速度修改
fast_arc_center	三维空间圆弧插补运动
fast_arc_circle	三维空间整圆插补运动
第 6 章 位置硬件捕捉	
enable_axis_capture	使能或者禁止硬件位置捕捉功能
set_axis_capture_source	设置轴的位置捕捉源
clear_axis_capture_pos	清除轴的位置锁存值
第 7 章 位置比较输出	
config_position_compare_output	配置位置比较输出功能参数
set_compare_link_output	配置位置比较输出功能输出口
write_compare_position	设置非等间距的比较输出位置
set_compare_counter	设置总计的位置比较输出数量
enable_postion_compare	使能或者禁止位置比较输出功
第八章 常见应用	
8.2.1 多卡连接	
command_set_mc_ip_addr	设置控制器 IP 地址以及控制器 ID
8.2.2 喷射阀控制	
config_jet_ports	配置喷射阀输出端口
set_jet_track_params	配置喷射阀控制参数
config_track_jet	配置喷射阀轨迹位置比较输出参数
set_compare_datas	配置比较输出点绝对坐标写入硬件 fifo
reset_compare_datas	清除比较输出点硬件 fifo
start_track_jet	启动喷射阀矢量位置比较输出
stop_jet	停止喷射阀输出
change_jet_track_dis	轨迹运动中修改喷射阀打点间距
start_points_jet	启动喷射阀按固定频率输出
wait_jet_done	等待 start_points_jet 指令执行完毕后，再执行后续指令
8.2.3 四轴/五轴轨迹运动配合喷射阀矢量位置比较输出	
fast_line4	轨迹运动 4 轴插补

fast_line5	轨迹运动 5 轴插补
8.2.4 遇限位缓停	
set_axis_stop_mode_el	设置轴遇限位停止模式
8.2.5 轨迹运动中的辅助轴控制	
change_axis_speed	轴运动变速
8.2.6 轨迹运动暂停/恢复/急停/缓停	
motion_pause	加工暂停
motion_resume	加工恢复
motion_sudden_stop	加工停止
motion_decel_stop	加工缓停
set_pause_outports_ctrl	配置输出口暂停时状态
8.2.7 条件暂停	
wait_inport_condition	等待输入口条件有效后再执行后续指令
set_mc_error_code	设置控制器的错误码
8.2.8 延时功能	
delay_time	延时
8.2.9 轴硬限位&轴报警使能与禁止	
enable_axis_el	使能轴硬限位功能
enable_axis_alarm	使能轴报警功能
8.2.10 输出口开启后自动延时关闭	
lag_outport_bit	控制单路输出口状态，滞后控制，滞后过程中，后续指令继续输出
lag_outport_byte	控制多路输出口状态，滞后控制，滞后过程中，后续指令继续输出
8.2.11 立即指令	
imm_set_profile	立即配置轴回零运动和点位运动的 T 型速度曲线
imm_set_profile_s	立即配置轴回零运动和点位运动的 S 型速度曲线
imm_fast_pmove	立即相对位置模式下的单轴点位运动（T 型速度曲线）
imm_fast_pmove_s	立即相对位置模式下的单轴点位运动（S 型速度曲线）
imm_outport_byte	立即控制多路输出口输出
imm_outport_bit	立即控制单路输出口输出
imm_config_jet_ports	立即配置喷射阀轨迹位置比较输出参数
imm_start_points_jet	立即配置比较输出点绝对坐标写入硬件 fifo
imm_stop_jet	立即清除比较输出点硬件 fifo

第 2 章 系统配置

在使用运动控制器进行各种操作之前，需要对运动控制器进行配置，使运动控制器的状态和各种工作模式能够满足客户的要求。这个过程叫做系统配置。

系统配置的内容包含：控制器初始化；工作轴组信息；各个轴的脉冲模式、脉冲当量、专用输入/输出口配置、最大速度/加速度、许可跳变速度、编码器参数、伺服使能模式。

2.1 指令列表

2.2.1 控制器初始化	
init_api	初始化运动控制器
2.2.2 轴配置	
set_channel_work_axis_group	配置工作轴组
set_axis_private_output_level	配置轴专用输出口
set_axis_private_input_level	配置轴专用输入口
set_axis_pulse_per_mm	配置轴脉冲当量
set_axis_max_speed	配置轴最大速度
set_axis_jerk_speed	配置轴许可跳变速度
enable_axis_servo_on	配置轴伺服使能
set_axis_encode_param	配置轴编码器反馈参数
set_pulse_abs_pos	配置轴脉冲位置
set_encode_abs_pos	配置编码器反馈位置
2.2.3 拓展轴配置	
enable_exaxis	配置拓展轴使能
set_exaxis_channel	配置拓展轴到关联通道中

2.2 重点说明

2.2.1 控制器初始化

控制器初始化是正常调用其他指令的基础，本文后续的所有指令都需要在初始化成功后才能正常调用。

调用示例

```
if (0 == init_api())//初始化运动控制器
{
    AfxMessageBox(_T("控制器初始化成功!"));
}
```

指令说明

init_api

```
int init_api(const char * ipaddr = NULL, int port = 502, int timeout = 3000, int idMc = 0);
//-----------------------------------------------
// 输入参数:
```

```
//        ipaddr          -    目标MC控制器IP地址。NULL表示连接IP为192.168.192.34的控制器
//        port            -    modbus_tcp通讯端口号。保持默认值，禁止修改
//        timeout         -    设置超时时间,单位ms。保持默认值，不建议修改
//        idMC            -    控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：指令接口封装模块初始化
//-------------------------------------------------
```

注意：P2系列运动控制器默认IP为192.168.192.34，与之连接的电脑网卡IP需要在同一网段

2.2.2 轴配置

调用示例

```
set_channel_work_axis_group(0x3FF, 0x07, 1, 0); //将1~10都配置为0号通道的工作轴，其中1~3轴为联动轴
for (i = 0; i < 10; i++)
{//循环设置10个轴的具体配置
    set_axis_private_input_level(i+1,0,0,0,0); //设置轴的正/负硬限位、原点为低电平有效
    set_axis_private_output_level(i+1,0,1,0,0); //设置轴电机方向不反向、电机使能为高电平、脉冲模式为脉冲+方向、脉冲计数为下降沿计数
    set_axis_encode_param(i+1, 4, 1); //设置编码器为4倍频、编码器方向反向
    set_axis_pulse_per_mm(i+1, 1000); //设置轴脉冲当量为1000p/mm
    set_axis_max_speed(i+1, 2000, 20000); //设置轴最大速度为2000mm/s、最大加速度为20000mm/s2
    set_axis_jerk_speed(i+1, 50); //设置轴许可跳变速度为50mm/s
    enable_axis_servo_on(i+1, 1); //轴伺服使能开启
}
```

说明

set_channel_work_axis_group

```
int set_channel_work_axis_group(unsigned int workAxis, unsigned int linkAxis, int axisX, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：workAxis  -配置的工作轴，BIT0~BIT9分别表示1~10轴，1-表示配置，0-表示不配置
//         : linkAxis  -配置的联动轴，BIT0~BIT9分别表示1~10轴，1-表示配置，0-表示不配置
//         : axisX     -X轴轴号(该参数无意义)
//         : uCmdCh    -通道号,取值：0、1、2、3
//         : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置通道的工作轴和联动轴，注意，联动轴必须也要配置为工作轴，否则该条指令会调用失败
//-------------------------------------------------
```

注意：P2系列为4通道运动控制器，在各个通道无共用轴的前提下可以支持4个通道同时运行。指令set_channel_work_axis_group中描述的工作轴必须包含联动轴，联动轴可以参与插补运动，工作轴可以被暂停/恢复。后续介绍的绝大部分指令都带有“通道号”参数，用以控制将指定发送对应的通道进行执行。

set_axis_private_input_level

```
int set_axis_private_input_level(unsigned char ch, int posElLevel, int negElLevel, int orgLevel, int
```

```
alarmLevel, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：ch-轴号，1~10号轴；
//         : posElLevel - 正限位有效电平，0-低电平有效，1-高电平有效
//         : negElLevel - 负限位有效电平，0-低电平有效，1-高电平有效
//         : orgLevel   - 原点有效电平，0-低电平有效，1-高电平有效
//         : alarmLevel - 报警信号有效电平，0-低电平有效，1-高电平有效
//         : uCmdCh     -通道号，取值：0、1、2、3
//         : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置轴的专用输入口的有效电平
//--------------------------------------------------
```

set_axis_private_output_level

```
int set_axis_private_output_level(unsigned char ch, int dirMotor, int enableMotor, int modePulse, int
logicEdgePulse, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：ch-轴号，1~10号轴；
//         : dirMotor - 电机方向，0-不反向，1-反向
//         : enableMotor - 电机使能有效电平，0-低电平有效，1-高电平有效
//         : modePulse - 脉冲模式，0-脉冲方向模式，1-双脉冲模式
//         : logicEdgePulse - 脉冲边沿有效电平，0-下降沿，1-上升沿
//         : uCmdCh     -通道号，取值：0、1、2、3
//         : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置轴的专用输出口的有效电平
//--------------------------------------------------
```

set_axis_pulse_per_mm

```
int set_axis_pulse_per_mm(unsigned char ch, float pulsePermm, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：ch-轴号，1~10号轴；
//         : pulsePermm - 当量脉冲
//         : uCmdCh     -通道号，取值：0、1、2、3
//         : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置轴的当量脉冲
//--------------------------------------------------
```

set_axis_max_speed

```
int set_axis_max_speed(unsigned char ch,float maxSpeedMM, float maxAccSpeedMM, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：ch-轴号，1~10号轴；
//         : maxSpeedMM - 最大速度，单位毫米/秒
```

```
//          : maxAccSpeedMM  -最大加速度，单位毫米/秒
//          : uCmdCh     -通道号，取值：0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置轴的最大速度
//-------------------------------------------------
```

set_axis_jerk_speed

```
int set_axis_jerk_speed(unsigned char ch, float jerkSpeedMM, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 函数名称：set_axis_jerk_speed
// 输入参数：ch-轴号，1~10号轴；
//          : jerkSpeedMM - 每个轴的许可跳变速度，单位毫米/秒
//          : uCmdCh     -通道号，取值：0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置每个轴的许可跳变速度
//-------------------------------------------------
```

set_axis_encode_param

```
int set_axis_encode_param(unsigned char ch, int frequency, int encodeDir, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：ch-轴号，1~10号轴；
//          : frequency - 编码器倍频设置，1-1倍频，4-4倍频
//          : encodeDir - 编码器计数方向设置，0-不取反，1-取反
//          : uCmdCh     -通道号，取值：0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置轴的编码器参数
//-------------------------------------------------
```

enable_axis_servo_on

```
int enable_axis_servo_on(unsigned char ch, int enable, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：ch-轴号，1~10号轴；
//          : enable - 使能控制符，0-禁止，1-使能
//          : uCmdCh     -通道号，取值：0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：使能轴的伺服使能，该指令配合set_axis_private_output_level使用。
//-------------------------------------------------
```

set_pulse_abs_pos

```
int set_pulse_abs_pos(unsigned char ch, int absPos, int uCmdCh = 0, int idMc = 0);
```

```
//-------------------------------------------------
// 输入参数：ch-轴号，1~10号轴；
//         : abspos -  配置的轴指令位置值
//         : uCmdCh     -通道号，取值：0、1、2、3
//         : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：配置轴的指令位置值
//-------------------------------------------------
```

set_encode_abs_pos

```
int set_encode_abs_pos(unsigned char ch, int encodeAbsPos, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：ch-轴号，1~10号轴；
//         : encodeAbsPos - 配置的轴编码器位置值
//         : uCmdCh     -通道号，取值：0、1、2、3
//         : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：配置轴编码器位置值
//-------------------------------------------------
```

2.2.3 拓展轴配置

P2 系列可以将通用 IO 配置为拓展轴，相对于非拓展轴输出频率调整为 500khz，且仅具备 T 型速度曲线的点位运动、回零运动（MPC24A2 不具备）等功能。启动该功能可由 enable_exaxis 设置轴号，其限位触发的有效电平可由 set_input_level 进行设置，脉冲当量可由 set_axis_pulse_per_mm 进行设置。

启用拓展轴时，控制器上部分通用数字输入会作为限位，部分通用输出会作为脉冲+方向的输出口。详见下列信息：

拓展轴 IO 说明

MPC24A2：可拓展 2 轴。无限位。
MPC24A0：无拓展轴。
MPC2460：可拓展 4 轴。
MPC2260：可拓展 4 轴。
MPC2250：可拓展 4 轴。
MPC2240：可拓展 4 轴。
备注：拓展轴的专用输出口从 Out5 开始占用，拓展轴的专用输入口从 In1 开始占用。

通用输出（Out）	映射为拓展轴专用输出	通用输入（In）	映射为拓展轴专用输入
5	轴 1 脉冲输出（PUL1）	1	轴 1 负限位（EL-）
6	轴 1 脉冲输出（DIR1）	2	轴 1 正限位（EL+）
7	轴 2 脉冲输出（PUL2）	3	轴 2 负限位（EL-）
8	轴 2 脉冲输出（DIR2）	4	轴 2 正限位（EL+）

9	轴 3 脉冲输出（PUL3）	5	轴 3 负限位（EL-）
10	轴 3 脉冲输出（DIR3）	7	轴 3 正限位（EL+）
11	轴 4 脉冲输出（PUL4）	8	轴 4 负限位（EL-）
12	轴 4 脉冲输出（DIR4）	9	轴 4 正限位（EL+）

调用示例

```
enable_exaxis(15, 0,  0);//将11~14作为拓展轴进行使能，并且配置给0号控制器的0号通道。
for (i = 0; i < 4; i++)
{//循环设置4个拓展轴的具体配置
    set_axis_private_input_level(i+11,0,0,0,0); //设置轴的正/负硬限位、原点为低电平有效
    set_axis_pulse_per_mm(i+11, 1000); //设置轴脉冲当量为1000p/mm
    set_axis_max_speed(i+11, 200, 2000); //设置轴最大速度为2000mm/s、最大加速度为20000mm/s2
}
```

指令说明

enable_exaxis

```
int enable_exaxis(unsigned int exaxis, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：exaxis    -配置使能拓展轴。-D0~D3对应扩展轴11、12、13、14, 0-禁止，1-使能
//         : uCmdCh   -通道号,取值：0、1、2、3
//         : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：将通用IO配置为拓展轴并使能。
// 备注：功能开启后对应的输出输入口无发通过OUT口控制指令控制
//-------------------------------------------------
```

set_exaxis_channel

```
int set_exaxis_channel(int ch, int channel, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：ch    -扩展轴号，取值11、12、13、14    。
//         : channel  - 关联通道
//         : uCmdCh   -通道号,取值：0、1、2、3
//         : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：配置拓展轴到关联通道中。可受该通道调用的暂停，恢复指令影响。
// 备注：功能开启后对应的输出输入口无发通过OUT口控制指令控制
//-------------------------------------------------
```

第 3 章 状态检测

本章介绍如何通过指令调用获取控制器的所有状态信息。包含：控制器版本信息、I/O 状态信息、轴位

置信息、轴运动状态信息。

3.1 指令列表

控制器状态获取	
get_mc_all_states	获取控制器所有状态
api_get_lib_version	获取当前动态链接库的版本
open_api_record	生成文本获取调用函数的记录

3.2 重点说明

P2 系列运动控制器的所有状态信息都通过一条指令进行读取。该指令通过数据结构体完成数据传递，指令参数详细说明如下：

指令说明

get_mc_all_states

```
int get_mc_all_states(McAllStates* mcStatus, int idMc = 0);
//-------------------------------------------------
// 输入参数：mcStatus  -输出参数，控制卡所有状态
//         : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：获取控制所有状态信息
//-------------------------------------------------
```

McAllStates 结构体说明

```
typedef struct MC_ALL_STATES
{
    UINT32   VersionARM1APP;       //arm1版本号
    UINT32   VersionARM2APP;       //arm2版本号
    UINT32   VersionFpga;          //fpga版本号
    UINT32   VersionLinuxOs;       //linux版本号
    UINT32   VersionIopNet;
    UCHAR8   ProductSNID[12];      //控制器SN号
    UINT32   TimerAvailableDays;   //控制器允许使用时间
    UINT32   PrivateAxesStatus;    //轴专用输入口状态
    UINT32   MainInputState;       //主卡通用输入状态
    UINT32   ExtendInputState[2];  //扩展通用输入口状态
    UINT32   MainOutputState;      //主卡通用输出口状态
    UINT32   ExtendOutputState[2]; //扩展通用输出口状态
     INT32   AxisPosPlus[10];      //1~10轴位置信息（脉冲单位）
     INT32   AxisPosFeedback[10];  //1~10轴编码器位置信息（脉冲单位）
    UINT32   ErrorCodeArm0;
    UINT32   ErrorCodeArm1;
    UINT32   ProgressUpdateArm0;
    UINT32   ProgressUpdateArm1;
     INT32   AxisCaptureLock[10];
```

```
    WcStatesWords arm2Status;       //控制器状态
    UCHAR8  ProductRegSerialNo[20];
     UINT32   EncoderStatus;
     UINT32   ServoOutputStatus;     //伺服使能输出，bit0~bit9，对应1~10轴的使能输出
}McAllStates;
```

McAllStates.PrivateAxesStatus 轴专用输入口状态

bit 0	bit 1	bit 2	bit 3	bit 4	bit 5	bit 6	bit 7	bit 8	bit 9
1 轴负限位	2 轴负限位	3 轴负限位	4 轴负限位	5 轴负限位	6 轴负限位	7 轴负限位	8 轴负限位	9 轴负限位	10 轴负限位
bit 10	bit 11	bit 12	bit 13	bit 14	bit 15	bit 16	bit 17	bit 18	bit 19
1 轴正限位	2 轴正限位	3 轴正限位	4 轴正限位	5 轴正限位	6 轴正限位	7 轴正限位	8 轴正限位	9 轴正限位	10 轴正限位
bit 20	bit 21	bit 22	bit 23	bit 24	bit 25	bit 26	bit 27	bit 28	bit 29
1 轴报警	2 轴报警	3 轴报警	4 轴报警	5 轴报警	6 轴报警	7 轴报警	8 轴报警	9 轴报警	10 轴报警
bit 30	bit 31								

McAllStates. ExtendInputState 扩展板通用输入口状态

ExtendInputState[0]	I/O 扩展版 EA3232D 输入口状态。bit0~bit31 分别对应 In17~In48 端口
ExtendInputState[1]	I/O 扩展版 SMC3232D 输入口状态。bit0~bit31 分别对应 In1~In32 端口

McAllStates. MainOutputState 扩展板通用输出口状态

MainOutputState	P2 系列主卡输出口状态。bit0~bit15 分别对应 Out1~Out16 端口

McAllStates. ExtendOutputState 扩展板通用输出口状态

ExtendOutputState[0]	I/O 扩展版 EA3232D 输出口状态。bit0~bit31 分别对应 Out13~Out44 端口
ExtendOutputState[1]	I/O 扩展版 SMC3232D 输出口状态。bit0~bit31 分别对应 Out1~Out32 端口

McAllStates. WcStatesWords 控制器状态字

```
typedef struct WORK_STATUS_WORDS
{
    //加工通道1状态字
    UINT32 bWorkPauseProcess_1 : 1; //通道1暂停过程状态
    UINT32 bWorkPauseDone_1 : 1; //通道1暂停停止状态
    UINT32 bWorkResumeProcess_1 : 1; //通道1恢复过程状态
    UINT32 bCheckBatchDone1 : 1; //通道1指令执行状态
   //加工通道2状态字
    UINT32 bWorkPauseProcess_2 : 1; //通道2暂停过程状态
    UINT32 bWorkPauseDone_2 : 1; //通道2暂停停止状态
    UINT32 bWorkResumeProcess_2 : 1; //通道2恢复过程状态
    UINT32 bCheckBatchDone2 : 1; //通道2指令执行状态
    //加工通道3状态字
    UINT32 bWorkPauseProcess_3 : 1; //通道3暂停过程状态
    UINT32 bWorkPauseDone_3 : 1; //通道3暂停停止状态
    UINT32 bWorkResumeProcess_3 : 1; //通道3恢复过程状态
    UINT32 bCheckBatchDone3 : 1; //通道3指令执行状态
  //加工通道4状态字
```

```
    UINT32 bWorkPauseProcess_4 : 1; //通道4暂停过程状态
    UINT32 bWorkPauseDone_4 : 1; //通道4暂停停止状态
    UINT32 bWorkResumeProcess_4 : 1; //通道4恢复过程状态
    UINT32 bCheckBatchDone4 : 1; //通道4指令执行状态
  //轴实时运动状态
    UINT32 bAxis1RunStatus : 1; //物理轴1运动状态
    UINT32 bAxis2RunStatus : 1; //物理轴2运动状态
    UINT32 bAxis3RunStatus : 1; //物理轴3运动状态
    UINT32 bAxis4RunStatus : 1; //物理轴4运动状态
    UINT32 bAxis5RunStatus : 1; //物理轴5运动状态
    UINT32 bAxis6RunStatus : 1; //物理轴6运动状态
    UINT32 bAxis7RunStatus : 1; //物理轴7运动状态
    UINT32 bAxis8RunStatus : 1; //物理轴8运动状态
    UINT32 bAxis9RunStatus : 1; //物理轴9运动状态
    UINT32 bAxis10RunStatus : 1; //物理轴10运动状态
    UINT32 bReserved : 6;
}WorkStatesWords;

typedef union WC_STATUS_WORDS
{
    UINT32 all;
    WorkStatesWords bit;
}WcStatesWords;
```

api_get_lib_version

```
int api_get_lib_version(unsigned int * version);
//-------------------------------------------------
// 输入参数：无。
// 输出参数：
// version:
// 返回值：0-成功 -1失败
// 功能描述：获取当前动态链接库的版本
//-------------------------------------------------
```

open_api_record

```
int open_api_record(int flag);
//-------------------------------------------------
// 输入参数：  flag：0：启动，1:不启动。
// 输出参数：无。
// 返回值：0-成功 -1失败
// 功能描述：生成文本获取调用函数的记录。在不调用该函数时该功能默认关闭。
//-------------------------------------------------
```

3.2.1 获取控制器版本信息

调用示例

```
McAllStates allStatus;
```

```
get_mc_all_states(&allStatus);//读取所有状态
CString ver;
ver.Format("Arm1:0x%08x    Arm2:0x%08x    Fpga:0x%08x",//按16进制转换
              allStatus.VersionARM1APP, allStatus.VersionARM2APP, allStatus.VersionFpga);
GetDlgItem(IDC_STATIC_VER)->SetWindowText(ver); //显示到UI
```

3.2.2 获取 I/O 状态信息

调用示例

```
McAllStates allStatus;
get_mc_all_states(&allStatus);//读取所有状态
CString str = "输出口状态:";
CString tmp = "";
int i = 0;
for (i = 0; i < 8; i++)//读取主卡上8路输出端口状态
{
    if (allStatus.MainOutputState & (0x01 << (i)))
    {
        tmp.Format("%d(1) ", i + 1); //输出口打开状态
    }
    else
    {
        tmp.Format("%d(0) ", i + 1); //输出口关闭状态
    }
    str += tmp;
}
GetDlgItem(IDC_STATIC_GENERAL_OUTPUT)->SetWindowText(str); //显示到UI

str = "输入口状态:";
for (i = 0; i < 10; i++)//读取I/O扩展版EA3232D上In17~In26输入端口状态
{
    if (g_cAllStatus.ExtendInputState[0]& (0x01 << (i)))
    {
        tmp.Format("%d(1) ", i + 1);
    }
    else
    {
        tmp.Format("%d(0) ", i + 1);
    }
    str += tmp;
}
GetDlgItem(IDC_STATIC_GENERAL_INPUT)->SetWindowText(str); //显示到UI
```

3.2.3 获取轴相关信息

调用示例

```
McAllStates allStatus;
get_mc_all_states(&allStatus);//读取所有状态
CString axisPos,feedbackPos,tempPos;
 feedbackPos = "编码器反馈位置(pluse):";
 axisPos = "轴脉冲位置(pluse):";
 for (int i = 0; i < 10; i++)
 {
      tempPos.Format("%d:%.3f  ", i + 1, allStatus.AxisPosPlus[i]); //读取轴绝对位置（脉冲）
      axisPos += tempPos;
      tempPos.Format("%d:%.3f  ", i + 1, allStatus.AxisPosFeedback[i]); //读取轴编码器位置（脉冲）
      feedbackPos += tempPos;
 }
GetDlgItem(IDC_STATIC_POS)->SetWindowText(axisPos); //显示到UI
GetDlgItem(IDC_STATIC_POS_FEEDBACK)->SetWindowText(feedbackPos);
```

3.2.4 获取运动状态信息

调用示例

```
McAllStates allStatus;
get_mc_all_states(&allStatus);//读取所有状态
CString str = "轴运动状态:";
 CString tmp = "";
 for (int i = 0; i < 10; i++)//检测10个轴的运行状态
 {
      if (g_cAllStatus.arm2Status.all & (0x01 << (16 + i)))
      {
           tmp.Format("%d(1) ", i + 1); //运动状态
      }
      else
      {
           tmp.Format("%d(0) ", i + 1); //停止状态
      }
      str += tmp;
 }
 GetDlgItem(IDC_STATIC_MOTION_STATE)->SetWindowText(str); //显示到UI

//检测通道1批处理运动是否运行结束
str = "通道1批处理运动状态:";
if (g_cAllStatus.arm2Status.bCheckBatchDone1==1)
 {
      tmp="运动结束 ";
 }
 else
 {
      tmp="运动中… ";
 }
 str += tmp;
 GetDlgItem(IDC_STATIC_BAT_MOTION_STATE)->SetWindowText(str); //显示到UI
```

第 4 章 I/O 控制

本章介绍如何通过指令调用实现对控制器的输出口和输入口控制。P2 系列运动控制器主卡具备 8 路通用输出口，通过主卡上的 EX I/O 端口可以外接 IO 扩展卡 EA3232D，实现另外 32 路通用输出口和 32 路通用输入口控制。如果在某些使用场合，以上 IO 口仍不能满足控制要求，那么通过主卡上的 RS485 端口外接 IO 扩展卡 SMC3232D，可以再次实现 32 路通用输出口和 32 路通用输入口控制。

4.1 指令列表

4.2.1 通用输出口控制	
outport_bit	控制单路输出口输出
outport_byte	控制多路输出口输出
4.2.2 通用输入口有效电平设置	
set_input_level	配置通用输入口的有效电平

4.2 重点说明

4.2.1 通用输出口控制

调用示例

```
//OUT 依次打开输出口
while(1)
{
    static int mIndexSmc3232D = 1;
    static int mIndexEa3232D = 1;
    static int mIndexMainBoard = 1;
     if (mIndexMainBoard > 8)//主卡输出口的状态
     {
          mIndexMainBoard = 1;
          outport_byte(0xFFFFFFFF, 0, 0, 0, 0, 0);//关闭主卡 8 路输出口
     }
     else
     {
          outport_bit (mIndexMainBoard, 1);//依次点亮主卡 8 路输出口
          mIndexMainBoard++;
     }
     if (mIndexEa3232D > 32)//EA3232D 输出口控制
     {
          mIndexEa3232D = 1;
          outport_byte(0, 0, 0xFFFFFFFF, 0, 0, 0);//关闭 EA3232D 的 32 路输出口
     }
     else
     {
          outport_bit(32 + mIndexEa3232D, 1);//依次点亮 EA3232D 的 32 路输出口
          mIndexEa3232D++;
     }
     if (mIndexSmc3232D > 32)//SMC3232D 输出口控制
```

```
    {
        mIndexSmc3232D = 1;
        outport_byte(0, 0,0x0 , 0, 0xFFFFFFFF, 0);//关闭 SMC3232D 的 32 路输出口
    }
    else
    {
        outport_bit(64 + mIndexSmc3232D, 1);//依次点亮 SMC3232D 的 32 路输出口
        mIndexSmc3232D++;
    }
    Sleep(100);//间隔100ms操作后面的端口
}
```

指令说明

outport_bit

```
int outport_bit(int portIndex, int state, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数:
//      portIndex - 输出口端口号，1~32：主卡,33~64 表示 EA3232D 对应 Out17~Out48 号输出口，65~96 表示 SMC3232D
//      对应的 OUT1~OUT32 号输出口
//      state     - 输出口状态，0-关闭，1-打开
//      uCmdCh    - 通道号,取值：0、1、2、3
//      idMC      -  控制器 ID
// 输出参数:
// 返回值：0-成功 -1 失败
// 功能描述：控制单路输出口输出
//-------------------------------------------------
```

outport_byte

```
int outport_byte(unsigned int mainBoardPorts, unsigned int mainBoardState,unsigned int extendBoardPorts,
unsigned int extendBoardState,   unsigned int smcBoardPorts, unsigned int smcBoardState,int uCmdCh = 0, int
idMc = 0);
//-------------------------------------------------
// 输入参数:
//      mainBoardPorts  - 按位表示 1~32 路主卡输出口是否修改，1-修改,0-不修改；P2 系列控制器主卡 8 路输出
//      mainBoardState  - 按位表示主卡对应输出口修改后的状态，0-关闭，1-打开；
//      extendBoardPorts - 按位表示扩展板 EA3232D 的 OUT17~OUT48 路输出口是否修改，1-修改,0-不修改；
//      extendBoardState  - 按位表示 EA3232D 对应输出口修改后的状态，0-关闭，1-打开；
//      smcBoardPorts -  按位表示扩展板 SMC3232D 的 1~32 路输出口是否修改，1-修改,0-不修改；
//      smcBoardState  - 按位表示 SMC3232D 对应输出口修改后的状态，0-关闭，1-打开；
//      uCmdCh     - 通道号,取值：0、1、2、3
//      idMC      - 控制器 ID
// 输出参数:
// 返回值：0-成功 -1 失败
// 功能描述：控制多路输出口状态
//-------------------------------------------------
```

4.2.2 通用输入口有效电平设置

调用示例

```
    static int flg = 0;
    if (0 == flg)//设置 EA3232D 和 SMC3232D 的 32 路通用输入口为高电平有效
    {
        flg = 1;
        set_input_level(0, 0, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF);
    }
    else //设置 EA3232D 和 SMC3232D 的 32 路通用输入口为低电平有效
    {
        flg = 0;
        set_input_level(0, 0, 0xFFFFFFFF, 0, 0xFFFFFFFF, 0);
    }
```

指令说明

set_input_level

```
int set_input_level(unsigned int mainMask, unsigned int mainInputLevel, unsigned int extendMask, unsigned int
extendInputLevel,unsigned int smcMask, unsigned int smcInputLevel, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//      mainMask   -  按位表示主卡输入口是否配置，1-配置,0-不配置；P2 系列控制器该参数无意义
//      mainInputLevel  - 按位表示主卡输入有效电平，0-低电平，1-高电平；P2 系列控制器该参数无意义
//       extendMask - 按位表示扩展板 EA3232D 的 1~32 路输入口是否配置，1-配置,0-不配置；
//      extendInputLevel - 按位表示 EA3232D 对应输入口有效电平，0-低电平，1-高电平；
//       smcMask -  按位表示扩展板 SMC3232D 的 1~32 路输入口是否配置，1-配置,0-不配置；
//      smcInputLevel - 按位表示 SMC3232D 对应输入口有效电平，0-低电平，1-高电平；
//       uCmdCh    - 通道号,取值：0、1、2、3
//      idMC       -  控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：配置通用输入口的有效电平。输入口外接信号物理状态不变，修改其有效电平，会改变输入口状态。
//-------------------------------------------------
```

第 5 章 轴运动控制

根据运动模式不同，P2 系列运动控制器的运动指令分为三类：回零运动、点位运动和插补运动。其中点位运动和插补运动提供绝对位置和相对位置两种模式。

P2 系列运动控制器的回零模式有两种，一个是检测限位信号有效停止，二是检测编码器的 Z 脉冲信号有效停止。实际使用时可以将两种回零模式组合使用，以提高设置的回零效率和回零精度。注意：P2 系列运动控制器没有专用的原点信号。

P2 系列控制器有立即和批处理两种指令发送模式。立即模式下调用的相关设置和运动指令对控制器立即生效，因此立即模式下调用轴的运动指令之前需要通过 get_mc_all_states 函数来判断该轴当前是否处于停止状态，否则会导致运动指令调用失败。

批处理模式下调用的相关设置指令和运动指令会首先进入控制器的批处理缓冲区，然后由控制器按照顺序依次输出。速度前瞻模式是带有速度规划功能的特殊批处理。开启速度前瞻后，各段运动指令按照预先设置好的前瞻速度参数和轨迹矢量关系，由运动控制器自动规划整段轨迹的速度升降，保证整段轨迹运动的高效和稳定。

打包指令是减少 PC 与 P2 系列控制器通讯频率的一种方式。开启打包指令（begin_package_bat_cmd）以后，PC 端软件调用的指令不再发送到运动控制器，而是等到结束指令打包（end_package_bat_cmd）后一起发送。

5.1 指令列表

5.2.1 运动指令立即模式&批处理模式	
switch_cmd_type	切换通道指令输出模式
begin_package_bat_cmd	批处理打包指令开始
end_package_bat_cmd	结束指令打包，将缓冲中的数据发送到控制器中
5.2.2 回零运动	
set_profile	配置轴回零运动和点位运动的 T 型速度曲线
set_profile_s	配置轴回零运动和点位运动的 S 型速度速度
enable_axis_pulse_out	配置轴输出脉冲
fast_hmove	控制单轴回零运动
fast_hmove_multi	控制多轴回零运动
fast_hmove_by_z_encoder	控制单轴 Z 脉冲回零
fast_hmove_multi_by_z_encoder	控制多轴 Z 脉冲回零
5.2.3 点位运动	
fast_pmove	相对位置模式下的单轴点位运动（T 型速度曲线）
fast_pmove2	相对位置模式下的两轴点位运动（T 型速度曲线）
fast_pmove3	相对位置模式下的三轴点位运动（T 型速度曲线）
fast_pmove4	相对位置模式下的四轴点位运动（T 型速度曲线）
fast_pmove5	相对位置模式下的五轴点位运动（T 型速度曲线）
fast_pmove_s	相对位置模式下的单轴点位运动（S 型速度曲线）
fast_pmove2_s	相对位置模式下的两轴点位运动（S 型速度曲线）

fast_pmove3_s	相对位置模式下的三轴点位运动（S 型速度曲线）
fast_pmove4_s	相对位置模式下的四轴点位运动（S 型速度曲线）
fast_pmove5_s	相对位置模式下的五轴点位运动（S 型速度曲线）
fast_abs_pmove	绝对位置模式下的单轴点位运动（T 型速度曲线）
fast_abs_pmove2	绝对位置模式下的两轴点位运动（T 型速度曲线）
fast_abs_pmove3	绝对位置模式下的三轴点位运动（T 型速度曲线）
fast_abs_pmove4	绝对位置模式下的四轴点位运动（T 型速度曲线）
fast_abs_pmove5	绝对位置模式下的五轴点位运动（T 型速度曲线）
fast_abs_pmove_s	绝对位置模式下的单轴点位运动（S 型速度曲线）
fast_abs_pmove2_s	绝对位置模式下的两轴点位运动（S 型速度曲线）
fast_abs_pmove3_s	绝对位置模式下的三轴点位运动（S 型速度曲线）
fast_abs_pmove4_s	绝对位置模式下的四轴点位运动（S 型速度曲线）
fast_abs_pmove5_s	绝对位置模式下的五轴点位运动（S 型速度曲线）
axis_decel_stop	控制轴缓停
axis_sudden_stop	控制轴急停
5.2.4 插补运动	
set_vector_profile	配置插补运动矢量 T 型速度曲线
set_vector_profile_s	配置插补运动矢量 S 型速度曲线
fast_line2	相对位置模式下的两轴直线插补运动（T 型速度曲线）
fast_line3	相对位置模式下的三轴直线插补运动（T 型速度曲线）
fast_line4	相对位置模式下的四轴直线插补运动（T 型速度曲线）
fast_line5	相对位置模式下的五轴直线插补运动（T 型速度曲线）
fast_line6	相对位置模式下的六轴直线插补运动（T 型速度曲线）
fast_line2_s	相对位置模式下的两轴直线插补运动（S 型速度曲线）
fast_line3_s	相对位置模式下的三轴直线插补运动（S 型速度曲线）
fast_line4_s	相对位置模式下的四轴直线插补运动（S 型速度曲线）
fast_line5_s	相对位置模式下的五轴直线插补运动（S 型速度曲线）
fast_abs_line2	绝对位置模式下的两轴直线插补运动（T 型速度曲线）
fast_abs_line3	绝对位置模式下的三轴直线插补运动（T 型速度曲线）
fast_abs_line4	绝对位置模式下的四轴直线插补运动（T 型速度曲线）
fast_abs_line5	绝对位置模式下的五轴直线插补运动（T 型速度曲线）
fast_abs_line2_s	绝对位置模式下的两轴直线插补运动（S 型速度曲线）
fast_abs_line3_s	绝对位置模式下的三轴直线插补运动（S 型速度曲线）
fast_abs_line4_s	绝对位置模式下的四轴直线插补运动（S 型速度曲线）
fast_abs_line5_s	绝对位置模式下的五轴直线插补运动（S 型速度曲线）
5.2.5 速度前瞻运动	
curve_begin	速度前瞻运动开始
curve_end	速度前瞻运动结束
set_arc_segment_dis	配置圆弧和整圆拆分精度
c_set_track_speed	设置前瞻运动矢量速度
c_change_track_speed	前瞻运动中轨迹速度修改
fast_arc_center	三维空间圆弧插补运动
fast_arc_circle	三维空间整圆插补运动

5.2 重点说明

5.2.1 运动指令立即模式&批处理模式

调用示例

```
switch_cmd_type(1); //切换指令模式为批处理模式
begin_package_bat_cmd();//开始指令打包
//…
//指令序列…
//…
end_package_bat_cmd();//指令打包结束，并将打包的指令序列发送到控制器执行
```

注意：指令打包的目的是减少PC与控制器的通讯次数，降低通讯压力

指令说明

switch_cmd_type

```
int switch_cmd_type(unsigned char cmdType, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     cmdType - 指令模式，立即指令或者批处理指令,0-立即指令，1-批处理指令
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：切换指令模式，切换为立即指令或者批处理指令
//-------------------------------------------------
```

begin_package_bat_cmd

```
int begin_package_bat_cmd(int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：批处理打包指令开始，注意，该指令会清空缓冲区，如果缓冲区有指令则会被清空
//-------------------------------------------------
```

end_package_bat_cmd

```
int end_package_bat_cmd(int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：结束指令打包，将缓冲中的数据发送到控制器中
//-------------------------------------------------
```

5.2.2 回零运动

调用示例

```
switch_cmd_type(1);        // 切换指令模式为批处理模式
set_profile(3, 20, 20, 20, 2000);// 设置 3 号轴回零速度参数
fast_hmove(3, 0);  // 启动3号轴负向回零运动，当负向限位信号有效时立即停止
```

指令说明

set_profile

```
int set_profile(unsigned char ch, float startSpeedMM, float HighSpeedMM, float endSpeedMM, float accSpeedMM,
int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//      ch - 轴号，1~10 号轴；
//      startSpeedMM - 起始速度，单位 mm/s
//      HighSpeedMM  - 目标速度，单位 mm/s
//      endSpeedMM   - 终止速度，单位 mm/s
//      accSpeedMM   - 加速度，单位 mm/s
//      uCmdCh    - 通道号,取值：0、1、2、3
//      idMC       -控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置轴回零和点位运动速度(T 型速度曲线)
//-------------------------------------------------
```

set_profile_s

```
int set_profile(unsigned char ch, float startSpeedMM, float HighSpeedMM, float endSpeedMM, float
accSpeedMM,float accaccSpeedMM, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//      ch - 轴号，1~10 号轴；
//      startSpeedMM - 起始速度，单位 mm/s
//      HighSpeedMM  - 目标速度，单位 mm/s
//      endSpeedMM   - 终止速度，单位 mm/s
//      accSpeedMM   - 加速度，单位 mm/s
//      accaccSpeedMM - 加加速度，单位 mm/s
//      uCmdCh    - 通道号,取值：0、1、2、3
//      idMC       -控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置轴回零和点位运动速度（S 型速度曲线）
//-------------------------------------------------
```

enable_axis_pulse_out

```
int enable_axis_pulse_out(unsigned char ch, int enable, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//      ch - 轴号，1~10 号轴；
//     enable 0：输出脉冲；1：不输出脉冲
//      uCmdCh    - 通道号,取值：0、1、2、3
//      idMC       -控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置轴回零和点位运动速度（S 型速度曲线）
//--------------------------------------------------
```

fast_hmove

```
int fast_hmove(unsigned char ch, unsigned char dir, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//     ch - 轴号，1~10 号轴；
//     dir - 回零方向，0-负向，1-正向
//     uCmdCh  - 通道号,取值：0、1、2、3
//     idMC        - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：单轴回零运动，注意该条运动指令可以在立即和批处理模式下使用。
//--------------------------------------------------
```

fast_hmove_multi

```
int fast_hmove_multi(unsigned int axisHome, unsigned int axisDir, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//     axisHome - 轴号，使用 D0~D9 分别表示 1~10 号轴；
//     axisDir - 回零方向，使用 D0~D9 分别表示 1~10 号轴，0-负向，1-正向
//     uCmdCh  - 通道号,取值：0、1、2、3
//     idMC - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：多轴回零运动，注意该条运动指令可以在立即和批处理模式下使用。
//--------------------------------------------------
```

fast_hmove_by_z_encoder

```
int fast_hmove_by_z_encoder(unsigned char ch, unsigned char dir, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//    ch - 轴号，1~10 号轴；
//    dir - 回零方向，0-负向，1-正向
//    uCmdCh  - 通道号,取值：0、1、2、3
```

```
//    idMC - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：单轴 Z 脉冲回零运动，注意该条运动指令可以在立即和批处理模式下使用。
//--------------------------------------------------
```

fast_hmove_multi_by_z_encoder

```
int fast_hmove_multi_by_z_encoder(unsigned int axisHome, unsigned int axisDir, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//     axisHome - 轴号，使用 D0~D9 分别表示 1~10 号轴；
//     axisDir - 回零方向，使用 D0~D9 分别表示 1~10 号轴，0-负向，1-正向
//     uCmdCh  - 通道号,取值：0、1、2、3
//     idMC - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：多轴 Z 脉冲回零，注意该条运动指令可以在立即和批处理模式下使用。
//--------------------------------------------------
```

5.2.3 点位运动

调用示例

```
//--------------------------------------------------
//单轴相对点位运动指令序列
switch_cmd_type(1);        // 切换为批处理模式
begin_package_bat_cmd();    // 批处理打包指令开始
set_profile(2, 20, 100, 20, 1000);//设置 2 号轴的点位运动速度参数
fast_pmove(2, 2000,0);      //启动 2 号轴运动到距离当前点 2000pp 处
end_package_bat_cmd();       //结束指令打包，将上述指令数据发送到控制器中
//--------------------------------------------------


//--------------------------------------------------
//单轴绝对点位运动指令序列
switch_cmd_type(1);        // 切换为批处理模式
begin_package_bat_cmd();    // 批处理打包指令开始
set_profile(2, 20, 100, 20, 1000);//设置 2 号轴的点位运动速度参数
fast_abs_pmove(2, 2000,0);      //启动 2 号轴运动到距离当前点 2000pp 处
end_package_bat_cmd();       //结束指令打包，将上述指令数据发送到控制器中
//--------------------------------------------------
```

指令说明

fast_pmove

```
int fast_pmove(int ch1, int dis1,int isWaitDoneThenNextCmd =1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//      ch1 - 轴号，1~10 号轴；
//      dis1 - 相对运动位移值，单位，脉冲
//      isWaitDoneThenNextCmd -是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//      uCmdCh  - 通道号,取值：0、1、2、3
//      idMC   - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：相对位置模式下的单轴点位运动(T型速度曲线)
//-------------------------------------------------
```

fast_pmove2

```
int fast_pmove2(int ch1, int dis1, int ch2, int dis2, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴运动位移值，单位，脉冲
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴运动位移值，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的两轴点位运动(T 型速度曲线)
//-------------------------------------------------
```

fast_pmove3

```
int fast_pmove3(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值，单位，脉冲
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴运动位移值，单位，脉冲
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3  - 3 号轴运动位移值，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条
```

```
指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的三轴点位运动(T 型速度曲线)
//--------------------------------------------------
```

fast_pmove4

```
int fast_pmove4(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int
isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值，单位，脉冲
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴运动位移值，单位，脉冲
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴运动位移值，单位，脉冲
//     ch4   -  4 号轴轴号，1~10 号轴；
//     dis4  - 4 号轴运动位移值，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条
指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的四轴点位运动(T 型速度曲线)
//--------------------------------------------------
```

fast_pmove5

```
int fast_pmove5(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5, int dis5,
int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值，单位，脉冲
//     ch2    - 2 号轴轴号，1~10 号轴；
//     dis2   - 2 号轴运动位移值，单位，脉冲
//     ch3    - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴运动位移值，单位，脉冲
//     ch4    - 4 号轴轴号，1~10 号轴；
//     dis4   - 4 号轴运动位移值，单位，脉冲
//     ch5    - 5 号轴轴号，1~10 号轴；
//     dis5   - 5 号轴运动位移值，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条
```

```
指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的五轴点位运动(T 型速度曲线)
//--------------------------------------------------
```

fast_pmove_s

```
int fast_pmove_s(int ch1, int dis1,int isWaitDoneThenNextCmd =1, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数
//      ch1 - 轴号，1~10 号轴；
//      dis1 - 相对运动位移值，单位，脉冲
//      isWaitDoneThenNextCmd -是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条
指令
//      uCmdCh  - 通道号,取值：0、1、2、3
//      idMC    - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：相对位置模式下的单轴点位运动(S型速度曲线)
//--------------------------------------------------
```

fast_pmove2_s

```
int fast_pmove2_s(int ch1, int dis1, int ch2, int dis2, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int
idMc = 0);
//--------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴运动位移值，单位，脉冲
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴运动位移值，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条
指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的两轴点位运动(S 型速度曲线)
//--------------------------------------------------
```

fast_pmove3_s

```
int fast_pmove3_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int isWaitDoneThenNextCmd = 1, int
uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
```

```
//     dis1  - 1 号轴运动位移值，单位，脉冲
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴运动位移值，单位，脉冲
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3  - 3 号轴运动位移值，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的三轴点位运动(S 型速度曲线)
//-------------------------------------------------
```

fast_pmove4_s

```
int fast_pmove4_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值，单位，脉冲
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴运动位移值，单位，脉冲
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴运动位移值，单位，脉冲
//     ch4   -  4 号轴轴号，1~10 号轴；
//     dis4  - 4 号轴运动位移值，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的四轴点位运动(S 型速度曲线)
//-------------------------------------------------
```

fast_pmove5_s

```
int fast_pmove5_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5, int dis5, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值，单位，脉冲
//     ch2    - 2 号轴轴号，1~10 号轴；
//     dis2   - 2 号轴运动位移值，单位，脉冲
//     ch3    - 3 号轴轴号，1~10 号轴；
```

```
//     dis3   - 3 号轴运动位移值，单位，脉冲
//     ch4    - 4 号轴轴号，1~10 号轴；
//     dis4   - 4 号轴运动位移值，单位，脉冲
//     ch5    - 5 号轴轴号，1~10 号轴；
//     dis5   - 5 号轴运动位移值，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的五轴点位运动(S 型速度曲线)
//--------------------------------------------------
```

fast_abs_pmove

```
int fast_abs_pmove(int ch1, int dis1, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数
//     ch1 - 轴号，1~10 号轴；
//     dis1 - 绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd -是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//     uCmdCh  - 通道号,取值：0、1、2、3
//     idMC   - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：绝对位置模式下的单轴点位运动，（T型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//--------------------------------------------------
```

fast_abs_pmove2

```
int fast_abs_pmove2(int ch1, int dis1, int ch2, int dis2, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴绝对运动目标位置，单位，脉冲
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的两轴点位运动，（T 型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//--------------------------------------------------
```

fast_abs_pmove3

```
int fast_abs_pmove3(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int isWaitDoneThenNextCmd = 1,
int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴绝对运动目标位置，单位，脉冲
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴绝对运动目标位置，单位，脉冲
//     ch3  - 3 号轴轴号，1~10 号轴；
//     dis3 - 3 号轴绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条
指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的三轴点位运动，（T 型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//-------------------------------------------------
```

fast_abs_pmove4

```
int fast_abs_pmove4(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int
isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴绝对运动目标位置，单位，脉冲
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴绝对运动目标位置，单位，脉冲
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴绝对运动目标位置，单位，脉冲
//     ch4   -  4 号轴轴号，1~10 号轴；
//     dis4  - 4 号轴绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条
指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的四轴点位运动，（T 型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//-------------------------------------------------
```

fast_abs_pmove5

```
int fast_abs_pmove5(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5, int
dis5, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
```

```
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴绝对运动目标位置，单位，脉冲
//     ch2    - 2 号轴轴号，1~10 号轴；
//     dis2   - 2 号轴绝对运动目标位置，单位，脉冲
//     ch3    - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴绝对运动目标位置，单位，脉冲
//     ch4    - 4 号轴轴号，1~10 号轴；
//     dis4   - 4 号轴绝对运动目标位置，单位，脉冲
//     ch5    - 5 号轴轴号，1~10 号轴；
//     dis5   - 5 号轴绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的五轴点位运动，（T 型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//-------------------------------------------------
```

fast_abs_pmove_s

```
int fast_abs_pmove_s(int ch1, int dis1, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//     ch1 - 轴号，1~10 号轴；
//     dis1 - 绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd -是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//     uCmdCh  - 通道号,取值：0、1、2、3
//     idMC   - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：绝对位置模式下的单轴点位运动，（S型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//-------------------------------------------------
```

fast_abs_pmove2_s

```
int fast_abs_pmove2_s(int ch1, int dis1, int ch2, int dis2, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴绝对运动目标位置，单位，脉冲
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
```

```
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的两轴点位运动，（S 型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//-------------------------------------------------
```

fast_abs_pmove3_s

```
int fast_abs_pmove3_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴绝对运动目标位置，单位，脉冲
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴绝对运动目标位置，单位，脉冲
//     ch3  - 3 号轴轴号，1~10 号轴；
//     dis3 - 3 号轴绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的三轴点位运动，（S 型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//-------------------------------------------------
```

fast_abs_pmove4_s

```
int fast_abs_pmove4_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴绝对运动目标位置，单位，脉冲
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴绝对运动目标位置，单位，脉冲
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴绝对运动目标位置，单位，脉冲
//     ch4   -  4 号轴轴号，1~10 号轴；
//     dis4  - 4 号轴绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
```

```
// 功能描述：绝对位置模式下的四轴点位运动，（S 型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//------------------------------------------------
```

fast_abs_pmove5_s

```
int fast_abs_pmove5_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5,
int dis5, int isWaitDoneThenNextCmd = 1, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴绝对运动目标位置，单位，脉冲
//     ch2    - 2 号轴轴号，1~10 号轴；
//     dis2   - 2 号轴绝对运动目标位置，单位，脉冲
//     ch3    - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴绝对运动目标位置，单位，脉冲
//     ch4    - 4 号轴轴号，1~10 号轴；
//     dis4   - 4 号轴绝对运动目标位置，单位，脉冲
//     ch5    - 5 号轴轴号，1~10 号轴；
//     dis5   - 5 号轴绝对运动目标位置，单位，脉冲
//     isWaitDoneThenNextCmd - 是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条
指令
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的五轴点位运动，（S 型速度曲线）注意该条运动指令只在立即和批处理模式下使用。
//------------------------------------------------
```

axis_decel_stop

```
int axis_decel_stop(int ch, int isImmCmd = 1, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数
//     ch - 轴号，1~10 轴
//      isImmCmd  - 是否立即执行，0-跟随该通道的指令模式，1-立即执行
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：缓停某个运动轴
//------------------------------------------------
```

注意：该指令参数isImmCmd可以控制当前停止指令立即执行或者跟随指令序列顺序执行

axis_sudden_stop

```
int axis_sudden_stop(int ch, int isImmCmd = 1, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数
//     ch - 轴号，1~10 轴
//      isImmCmd  - 是否立即执行，0-跟随该通道的指令模式，1-立即执行
```

```
//        uCmdCh     - 通道号,取值：0、1、2、3
//        idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：急停某个运动轴
//--------------------------------------------------
```

注意：该指令参数isImmCmd可以控制当前停止指令立即执行或者跟随指令序列顺序执行

5.2.4 插补运动

调用示例

```
switch_cmd_type(1);  //切换当前通道为批处理模式
begin_package_bat_cmd();  // 批处理打包指令开始
set_vector_profile(10, 200, 10, 2000); // 设置插补运动的矢量速度参数
fast_line3(1, 2000,2, 1000, 3, 500); //启动三轴直线插补
set_vector_profile(40, 600, 30, 2000); // 修改插补运动的矢量速度参数
fast_line3(1, 1000,2, 2000, 3, 200); //按照修改后的速度执行三轴直线插补
end_package_bat_cmd(uCmdCh);  //结束指令打包，将上述指令数据发送到控制器中
```

指令说明

set_vector_profile

```
int set_vector_profile(float startSpeedMM, float HighSpeedMM, float endSpeedMM, float accSpeedMM, int uCmdCh
= 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：
//     startSpeedMM - 矢量起始速度，单位 mm/s
//     HighSpeedMM  - 矢量目标速度，单位 mm/s
//     endSpeedMM   - 矢量终止速度，单位 mm/s
//     accSpeedMM   - 矢量加速度，单位 mm/s^2
//        uCmdCh     - 通道号,取值：0、1、2、3
//        idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置插补运动的矢量速度参数（T型速度曲线）
//--------------------------------------------------
```

set_vector_profile_s

```
int set_vector_profile_s(float startSpeedMM, float HighSpeedMM, float endSpeedMM, float accSpeedMM, float accaccSpeedMM,int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     startSpeedMM - 矢量起始速度，单位 mm/s
//     HighSpeedMM  - 矢量目标速度，单位 mm/s
//     endSpeedMM   - 矢量终止速度，单位 mm/s
//     accSpeedMM   - 矢量加速度，单位 mm/s^2
//     accaccSpeedMM   - 矢量加加速度，单位 mm/s^2
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置插补运动的矢量速度参数（S型速度曲线）
//-------------------------------------------------
```

fast_line2

```
int fast_line2(int ch1, int dis1, int ch2, int dis2, float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     vectorLength  -该微段的矢量长度，单位毫米，若小于 0，系统自动按照直线来计算
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的两轴直线插补运动，(T 型速度曲线)注意该条运动指令可以立即、批处理和速度前瞻三种模式下使用。
//-------------------------------------------------
```

fast_line3

```
int fast_line3(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3  - 3 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
```

```
//     vectorLength  -该微段的矢量长度，单位毫米，若小于 0，系统自动按照直线来计算
//       uCmdCh     - 通道号,取值：0、1、2、3
//       idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的三轴直线插补运动，(T 型速度曲线)注意该条运动指令可以立即、批处理和速度前瞻三种模式下使用。
//-------------------------------------------------
```

fast_line4

```
int fast_line4(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch4   -  4 号轴轴号，1~10 号轴；
//     dis4  - 4 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     vectorLength  -该微段的矢量长度，单位毫米，若小于 0，系统自动按照直线来计算
//       uCmdCh     - 通道号,取值：0、1、2、3
//       idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的四轴直线插补运动，(T 型速度曲线)注意该条运动指令只在立即和批处理模式下使用。
//-------------------------------------------------
```

fast_line5

```
int fast_line5(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5, int dis5, float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2    - 2 号轴轴号，1~10 号轴；
//     dis2   - 2 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3    - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch4    - 4 号轴轴号，1~10 号轴；
//     dis4   - 4 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch5    - 5 号轴轴号，1~10 号轴；
//     dis5   - 5 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     vectorLength  -该微段的矢量长度，单位毫米，若小于 0，系统自动按照直线来计算
```

```
//        uCmdCh     - 通道号,取值：0、1、2、3
//        idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的五轴直线插补运动，(T 型速度曲线)注意该条运动指令只在立即和批处理模式下使用。
//------------------------------------------------
```

fast_line6

```
int fast_line6(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5, int dis5,int ch6, int dis6,  float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2    - 2 号轴轴号，1~10 号轴；
//     dis2   - 2 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3    - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch4    - 4 号轴轴号，1~10 号轴；
//     dis4   - 4 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch5    - 5 号轴轴号，1~10 号轴；
//     dis5   - 5 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch6    - 6 号轴轴号，1~10 号轴；
//     dis6   - 6 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     vectorLength  -该微段的矢量长度，单位毫米，若小于 0，系统自动按照直线来计算
//        uCmdCh     - 通道号,取值：0、1、2、3
//        idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的五轴直线插补运动，(T 型速度曲线)注意该条运动指令只在立即和批处理模式下使用。
//------------------------------------------------
```

fast_line2_s

```
int fast_line2_s(int ch1, int dis1, int ch2, int dis2, float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     vectorLength  -该微段的矢量长度，单位毫米，若小于 0，系统自动按照直线来计算
//        uCmdCh     - 通道号,取值：0、1、2、3
//        idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
```

```
// 功能描述：相对位置模式下的两轴直线插补运动，(S 型速度曲线)注意该条运动指令可以立即、批处理和速度前瞻三种模式下使用。
//------------------------------------------------
```

fast_line3_s

```
int fast_line3_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3  - 3 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     vectorLength  -该微段的矢量长度，单位毫米，若小于 0，系统自动按照直线来计算
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的三轴直线插补运动，(S 型速度曲线)注意该条运动指令可以立即、批处理和速度前瞻三种模式下使用。
//------------------------------------------------
```

fast_line4_s

```
int fast_line4_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch4   -  4 号轴轴号，1~10 号轴；
//     dis4  - 4 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     vectorLength  -该微段的矢量长度，单位毫米，若小于 0，系统自动按照直线来计算
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的四轴直线插补运动，(S 型速度曲线)注意该条运动指令只在立即和批处理模式下使用。
//------------------------------------------------
```

fast_line5_s

```
int fast_line5_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5, int dis5,
```

```
float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2    - 2 号轴轴号，1~10 号轴；
//     dis2   - 2 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3    - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch4    - 4 号轴轴号，1~10 号轴；
//     dis4   - 4 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch5    - 5 号轴轴号，1~10 号轴；
//     dis5   - 5 号轴运动位移值。单位，脉冲，按照该轴脉冲当量计算转化。
//     vectorLength  -该微段的矢量长度，单位毫米，若小于 0，系统自动按照直线来计算
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：相对位置模式下的五轴直线插补运动，(S 型速度曲线)注意该条运动指令只在立即和批处理模式下使用。
//-------------------------------------------------
```

fast_abs_line2

```
int fast_abs_line2(int ch1, int dis1, int ch2, int dis2, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的两轴直线插补运动，(T 型速度曲线)注意该条运动指令只在批处理模式下使用。
//-------------------------------------------------
```

fast_abs_line3

```
int fast_abs_line3(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3  - 3 号轴轴号，1~10 号轴；
//     dis3 - 3 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
```

```
//        uCmdCh     - 通道号,取值：0、1、2、3
//        idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的三轴直线插补运动，(T 型速度曲线)注意该条运动指令只在批处理模式下使用。
//------------------------------------------------
```

fast_abs_line4

```
 int fast_abs_line4(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：
//      ch1   - 1 号轴轴号，1~10 号轴；
//      dis1   - 1 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//      ch2    - 2 号轴轴号，1~10 号轴；
//      dis2   - 2 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//      ch3    - 3 号轴轴号，1~10 号轴；
//      dis3    - 3 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//      ch4    -  4 号轴轴号，1~10 号轴；
//      dis4   - 4 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//        uCmdCh     - 通道号,取值：0、1、2、3
//        idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的四轴直线插补运动，(T 型速度曲线)注意该条运动指令只在批处理模式下使用。
//------------------------------------------------
```

fast_abs_line5

```
int fast_abs_line5(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5, int dis5, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：
//      ch1   - 1 号轴轴号，1~10 号轴；
//      dis1   - 1 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//      ch2     - 2 号轴轴号，1~10 号轴；
//      dis2    - 2 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//      ch3     - 3 号轴轴号，1~10 号轴；
//      dis3    - 3 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//      ch4     - 4 号轴轴号，1~10 号轴；
//      dis4    - 4 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//      ch5     - 5 号轴轴号，1~10 号轴；
//      dis5    - 5 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//        uCmdCh     - 通道号,取值：0、1、2、3
//        idMC       - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
```

```
// 功能描述：绝对位置模式下的五轴直线插补运动，(T 型速度曲线)注意该条运动指令只在批处理模式下使用。
//-------------------------------------------------
```

fast_abs_line2_s

```
int fast_abs_line2_s(int ch1, int dis1, int ch2, int dis2, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的两轴直线插补运动，(S 型速度曲线)注意该条运动指令只在批处理模式下使用。
//-------------------------------------------------
```

fast_abs_line3_s

```
int fast_abs_line3_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1 - 1 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2  - 2 号轴轴号，1~10 号轴；
//     dis2 - 2 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3  - 3 号轴轴号，1~10 号轴；
//     dis3 - 3 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的三轴直线插补运动，(S 型速度曲线)注意该条运动指令只在批处理模式下使用。
//-------------------------------------------------
```

fast_abs_line4_s

```
 int fast_abs_line4_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int uCmdCh
= 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2   - 2 号轴轴号，1~10 号轴；
//     dis2  - 2 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3   - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch4   -  4 号轴轴号，1~10 号轴；
```

```
//     dis4  - 4 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的四轴直线插补运动，(S 型速度曲线)注意该条运动指令只在批处理模式下使用。
//------------------------------------------------
```

fast_abs_line5_s

```
int fast_abs_line5_s(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5, int dis5, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：
//     ch1  - 1 号轴轴号，1~10 号轴；
//     dis1  - 1 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch2    - 2 号轴轴号，1~10 号轴；
//     dis2   - 2 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch3    - 3 号轴轴号，1~10 号轴；
//     dis3   - 3 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch4    - 4 号轴轴号，1~10 号轴；
//     dis4   - 4 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     ch5    - 5 号轴轴号，1~10 号轴；
//     dis5   - 5 号轴绝对运动目标位置。单位，脉冲，按照该轴脉冲当量计算转化。
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：绝对位置模式下的五轴直线插补运动，(S 型速度曲线)注意该条运动指令只在批处理模式下使用。
//------------------------------------------------
```

5.2.5 速度前瞻运动

调用示例

```
//------------------------------------------------
//3 轴速度前瞻轨迹运动指令序列
switch_cmd_type(1);       // 切换通道 0 指令模式为批处理模式
begin_package_bat_cmd();  // 批处理打包指令开始
set_arc_segment_dis(0.37f);  //设置整圆和圆弧的差分矢量长度为 0.37mm
c_set_track_speed(20, 500, 20, 10000);       //设置整段轨迹运动的速度参数
curve_begin();                  //速度前瞻开始
fast_line3(1, 1000, 2, 1000, 3, 500); //直线插补
fast_arc_circle(1, 2, 3, 1500, 1500, 400, 1000, 2000, 500); //圆弧插补
curve_end();//速度前瞻结束
end_package_bat_cmd();  // 批处理打包指令开始
//------------------------------------------------
```

```
//-------------------------------------------------
//速度前瞻轨迹运动中变速指令序列
switch_cmd_type(1);       // 切换通道 0 指令模式为批处理模式
begin_package_bat_cmd();  // 批处理打包指令开始
set_arc_segment_dis(0.37f);  //设置整圆和圆弧的差分矢量长度为 0.37mm
c_set_track_speed(20, 500, 20, 10000);       //设置整段轨迹运动的速度参数
curve_begin();                  //速度前瞻开始
fast_line3(1, 1000, 2, 1000, 3, 500); //直线插补
c_change_track_speed(1000); //调整轨迹前瞻速度为 1000
fast_arc_circle(1, 2, 3, 1500, 1500, 400, 1000, 2000, 500); //圆弧插补
curve_end();//速度前瞻结束
end_package_bat_cmd();  // 批处理打包指令开始
//-------------------------------------------------
```

指令说明

curve_begin

```
int curve_begin(int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数:
//      uCmdCh - 通道号,取值：0、1、2、3
//      idMC  - 控制器 ID
// 输出参数:
// 返回值：0-成功 -1失败
// 功能描述：速度前瞻开始，注意：调用该条指令时指令模式必须处于批处理模式。
//-------------------------------------------------
```

curve_end

```
int curve_end(int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数:
//      uCmdCh - 通道号,取值：0、1、2、3
//      idMC  - 控制器 ID
// 输出参数:
// 返回值：0-成功 -1 失败
// 功能描述：速度前瞻结束，注意：该条指令必须与curve_begin成对使用。
//-------------------------------------------------
```

set_arc_segment_dis

```
int set_arc_segment_dis(float arcSegmentDis, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数:
//      arcSegmentDis - 拆分长度，单位毫米
//        uCmdCh     - 通道号,取值：0、1、2、3
//        idMC       - 控制器 ID
// 输出参数:
// 返回值：0-成功 -1 失败
// 功能描述：设置圆弧和整圆的拆分长度，推荐数值为 0.37mm。
```

```
//-------------------------------------------------
```

c_set_track_speed

```
int c_set_track_speed(float startSpeedMM, float HighSpeedMM, float endSpeedMM, float accSpeedMM, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     startSpeedMM - 前瞻运动起始速度，单位 mm/s
//     HighSpeedMM  -前瞻运动目标速度，单位 mm/s
//     endSpeedMM   -前瞻运动终止速度，单位 mm/s
//     accSpeedMM   -前瞻运动加速度，单位 mm/s^2
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置前瞻速度参数
//-------------------------------------------------
```

c_change_track_speed

```
int c_change_track_speed(float speedNew, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     speedNewMM  - 修改后的新速度,单位 mm/s
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：速度前瞻中轨迹速度修改
//-------------------------------------------------
```

fast_arc_center

```
int fast_arc_center(int ch1, int ch2, int ch3,  int disArcNode1, int disArcNode2, int disArcNode3,int disArcEnd1, int disArcEnd2, int disArcEnd3, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//     ch1 - 1 号轴轴号，1~10 号轴；
//     ch2 - 2 号轴轴号，1~10 号轴；
//     ch3 - 3 号轴轴号，1~10 号轴；
//     disArcNode1 - 1 号轴圆弧中间点相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcNode2 - 2 号轴圆弧中间点相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcNode3 - 3 号轴圆弧中间点相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcEnd1  - 1 号轴圆弧结束点相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcEnd2  - 2 号轴圆弧结束点相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcEnd3  -  3 号轴圆弧结束点相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
```

```
// 返回值：0-成功，其它-失败
// 功能描述：3 轴空间圆弧插补指令，圆弧起点为当前点，圆弧中间点和圆弧结束点由相对于起点的位置确定。注意该条运动指令只能在速度前瞻模式下使用。
//-------------------------------------------------
```

fast_arc_circle

```
int fast_arc_circle(int ch1, int ch2, int ch3, int disArcNode1, int disArcNode2, int disArcNode3, int disArcEnd1, int disArcEnd2, int disArcEnd3, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//     ch1 - 1 号轴轴号，1~10 号轴；
//     ch2 - 2 号轴轴号，1~10 号轴；
//     ch3 - 3 号轴轴号，1~10 号轴；
//     disArcNode1 - 1 号轴整圆中间点相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcNode2 - 2 号轴整圆中间点相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcNode3 - 3 号轴整圆中间点相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcEnd1  - 1 号轴整圆中间点 2 相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcEnd2  - 2 号轴整圆中间点 2 相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//     disArcEnd3  - 3 号轴整圆中间点 2 相对于起点的脉冲位置。单位，脉冲，按照该轴脉冲当量计算转化。
//       uCmdCh    - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：3 轴空间整圆插补指令，整圆起点为当前点，两个整圆中间点由相对于起点的位置确定。注意该条运动指令只能在速度前瞻模式下使用。
//-------------------------------------------------
```

第 6 章 位置硬件捕捉

P2 系列控制器提供位置硬件高速捕捉功能，支持 10 组位置锁存器，每组锁存器的触发信号为每根轴的

正向限位信号。注意使能开启轴的位置捕捉功能后，该轴的正向限位信号自动变为锁存器的触发信号输入端口，不再具备正向限位功能。因此使用完成后需要关闭该轴的位置捕捉功能，以便恢复正限位信号的保护功能。

P2 系列控制器的位置硬件捕捉为单次锁存，锁存信号触发后需要及时读取本次锁存值。当锁存信号再次被触发，上次锁存的位置值将被覆盖。锁存器保存的位置值可以设置为内部脉冲或者是编码器反馈值。锁存数值通过状态查询函数 get_mc_all_states 获取，保存在数据结构 McAllStates 的变量 AxisCaptureLock 中。

6.1 指令列表

位置硬件捕捉	
enable_axis_capture	使能或者禁止硬件位置捕捉功能
set_axis_capture_source	设置轴的位置捕捉源
clear_axis_capture_pos	清除轴的位置锁存值

6.2 重点说明

调用示例

```
//-----------------------------------------------------
// 配置使能硬件位置捕捉功能并启动轴运动指令序列
switch_cmd_type(1);       // 切换通道 0 指令模式为批处理模式
begin_package_bat_cmd();  //指令打包
set_axis_capture_source(1, 1); // 设置 1 号轴硬件位置捕捉源为编码器位置
enable_axis_capture(1, 1); // 使能 1 号轴硬件位置捕捉功能
set_axis_private_input_level(1, 1, 0, 0, 0);// 配置 1 号轴正限位信号高电平有效。注意：根据实际情况确定
clear_axis_capture_pos(1); //清除 1 轴的历史捕捉数据
set_profile(1, 20, 20, 20, 2000);  // 设置点位运动速度参数
// 启动 1 号轴正向运动 20000。在运动过程中正限位信号有效时控制器将自动锁存当前的 1 号轴编码器位置
fast_pmove(1, 20000);
end_package_bat_cmd();  //指令打包结束
//-----------------------------------------------------

//获取捕捉位置值并显示
void CLibDemoDlg::OnBnClickedGetCapturedInfo()
{
    McAllStates allStatus;
    get_mc_all_states(&allStatus);//读取所有状态
    CString tempPos,posCapture;;
     posCapture = "1轴捕捉位置(mm):";
    if (aAllStatus.AxisCaptureLock[0] == 0) //检测1轴是否完成了位置捕捉
    {
         AfxMessageBox(_T("未检测到1轴的位置捕捉信息!"));
    }
    else
    {
         tempPos.Format("%d", aAllStatus.AxisCaptureLock[0]);
         posCapture += tempPos;
         GetDlgItem(IDC_STATIC_AXIS_CAPTURES)->SetWindowText(posCapture); //显示到UI
        clear_axis_capture_pos(1); //清除 1 轴的历史捕捉数据
```

```
    }
}
```

指令说明

enable_axis_capture

```
int enable_axis_capture(int ch, int enable,int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch - 轴号，1~10 号轴；
//     enable  - 使能或者禁止，0-禁止，1-使能
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：使能或者禁止硬件位置捕捉功能。注意：使用完硬件位置捕捉功能后需要禁止以便恢复正限位。
//-------------------------------------------------
```

set_axis_capture_source

```
int set_axis_capture_source(int ch, int sourceType, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch - 轴号，1~10 号轴；
//     sourceType - 位置比较源，0-内部脉冲，1-编码器
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置轴的位置捕捉源。即锁存信号产生时锁存器锁存的位置是内部脉冲值还是编码器反馈值。
//-------------------------------------------------
```

clear_axis_capture_pos

```
int clear_axis_capture_pos(int ch, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     ch - 轴号，1~10 号轴；
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：清除轴的位置锁存值
//-------------------------------------------------
```

第 7 章　单轴位置比较输出

在轴运动过程中（包括点位运动和插补运动，多轴运动时只与其中某一个轴进行位置比较输出），控

制器自动将编码器位置或内部脉冲计数器位置与设定位置进行比较，当编码器位置或内部脉冲计数器位置到达设定位置时，在该轴所配置的输出端口输出脉冲或反转当前的电平信号。请注意，该功能仅仅可以运行在通道 0 与通道 1。

7.1 指令列表

位置比较输出	
config_position_compare_output	配置位置比较输出功能参数
set_compare_link_output	配置位置比较输出功能输出口
write_compare_position	设置非等间距的比较输出位置
set_compare_counter	设置总计的位置比较输出数量
enable_postion_compare	使能或者禁止位置比较输出功能

7.2 重点说明

调用示例

```
switch_cmd_type(1); // 切换通道 0 指令模式为批处理模式
begin_package_bat_cmd();  //指令打包

/// 设置 2 号轴，比较源为内部脉冲，输出宽度为 100000 微秒的脉冲，初始电平为低电平，比较位置之间非等间距
config_position_compare_output(2, 0, 0, 0, 100000, 0, 1);

set_compare_link_output(1, 1, 0); // 配置位置比较输出端口为主卡输出口 1
set_profile(2, 20, 20, 20, 2000, 0);//设置 2 号轴的运动速度参数

///准备比较输出数据
///每运动 20 个脉冲，输出一次信号，连续输出 90 次
int nDIs[90];
for (int i = 0;i < 90; i++)
{
    nDIs[i] = 20* (i + 1);
}

write_compare_position(0, nDIs, 30,0);//设置序号 0^30 的比较位置值
write_compare_position(30, nDIs, 30,0);//设置序号 31^60 的比较位置值
write_compare_position(60, nDIs, 30,0);//设置序号 61^90 的比较位置值
set_compare_counter(90, 0); //启动 90 次位置比较
enable_postion_compare(1, 0);//使能位置比较输出功能
fast_pmove(2, 100000 * m_fPlusePerMM[2],1, 0);//启动2号轴开始运动
end_package_bat_cmd();  //指令打包结束
```

指令说明

config_position_compare_output

```
int config_position_compare_output(int ch, int source, int outMode, int initLevel, int pulseTimeUs, int modeDis, int uCmdCh = 0, int idMc = 0);
```

```
//-------------------------------------------------
// 输入参数：
//     ch - 轴号，1~10 号轴；
//     source - 比较源，0-内部脉冲，1-编码器反馈
//     outMode - 输出模式，0-表示输出脉冲，脉冲宽度由 pulseTimeUs 决定，1-表示电平变化，即反转当前电平
//     initLevel - 初始电平，0-低电平，1-高电平
//     pulseTimeUs - 输出为脉冲模式时，脉冲的宽度，单位微秒
//     modeDis - 距离模式，0-非等间距输出，1-等间距输出。注意：等间距模式暂时未提供。
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：位置比较输出功能配置
//-------------------------------------------------
```

set_compare_link_output

```
int set_compare_link_output(int outputIndex, int enable, int uCmdCh =0, int idMc =0);
//-------------------------------------------------
// 输入参数：
//     outputIndex - 输出口端口号，主卡 1~8 号输出口；
//     enable - 使能或者禁止，0-禁止，1-使能
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：位置比较输出功能输出口关联配置
//-------------------------------------------------
```

write_compare_position

```
int write_compare_position(int startIndex, int* listCompareDis, int counter, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     startIndex - 数组中第 1 个比较位置在总的比较序列中序号，从 0 开始；
//      listCompareDis - 比较输出位置指针；
//      counter - listCompareDis 中有效数据的数量，最大值为 30;
//      uCmdCh - 通道号,取值：0、1、2、3
//     idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置非等间距的比较输出位置。注意：该函数可以重复调用，单次最大设置30个比较位置值。总计位置比较输出点数量最大为4096个
//-------------------------------------------------
```

set_compare_counter

```
int set_compare_counter(int counterCompare, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//     counterCompare -设置实际执行的位置比较输出的数量，最大为 4096
//      uCmdCh - 通道号,取值：0、1、2、3
```

```
//      idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置实际执行的位置比较输出数量。注意：该数量值不能大于 write_compare_position 设置的比较位置总
             数量。该值最大为 4096。
//------------------------------------------------
```

enable_postion_compare

```
int enable_postion_compare(int enable, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：
//      enable -使能或者禁止，0-禁止，1-使能
//      uCmdCh - 通道号,取值：0、1、2、3
//      idMC  - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：位置比较输出使能功能
//------------------------------------------------
```

第 8 章 常见应用

8.1 指令列表

8.2.1 多卡连接	
command_set_mc_ip_addr	设置控制器 IP 地址以及控制器 ID
8.2.2 喷射阀控制	
config_jet_ports	配置喷射阀输出端口
set_jet_track_params	配置喷射阀控制参数

config_track_jet	配置喷射阀轨迹位置比较输出参数
set_compare_datas	配置比较输出点绝对坐标写入硬件 fifo
reset_compare_datas	清除比较输出点硬件 fifo
start_track_jet	启动喷射阀矢量位置比较输出
stop_jet	停止喷射阀输出
change_jet_track_dis	轨迹运动中修改喷射阀打点间距
start_points_jet	启动喷射阀按固定频率输出
wait_jet_done	等待 start_points_jet 指令执行完毕后，再执行后续指令
8.2.3 四轴/五轴轨迹运动配合喷射阀矢量位置比较输出	
fast_line4	轨迹运动 4 轴插补
fast_line5	轨迹运动 5 轴插补
8.2.4 遇限位缓停	
set_axis_stop_mode_el	设置轴遇限位停止模式
8.2.5 轨迹运动中的辅助轴控制	
change_axis_speed	轴运动变速
8.2.6 轨迹运动暂停/恢复/急停/缓停	
motion_pause	加工暂停
motion_resume	加工恢复
motion_sudden_stop	加工停止
motion_decel_stop	加工缓停
set_pause_outports_ctrl	配置输出口暂停时状态
8.2.7 条件暂停	
wait_inport_condition	等待输入口条件有效后再执行后续指令
set_mc_error_code	设置控制器的错误码
8.2.8 延时功能	
delay_time	延时
8.2.9 轴硬限位&轴报警使能与禁止	
enable_axis_el	使能轴硬限位功能
enable_axis_alarm	使能轴报警功能
8.2.10 输出口开启后自动延时关闭	
lag_outport_bit	控制单路输出口状态，滞后控制，滞后过程中，后续指令继续输出
lag_outport_byte	控制多路输出口状态，滞后控制，滞后过程中，后续指令继续输出
8.2.11 立即指令	
imm_set_profile	立即配置轴回零运动和点位运动的 T 型速度曲线
imm_set_profile_s	立即配置轴回零运动和点位运动的 S 型速度曲线
imm_fast_pmove	立即相对位置模式下的单轴点位运动（T 型速度曲线）
imm_fast_pmove_s	立即相对位置模式下的单轴点位运动（S 型速度曲线）
imm_outport_byte	立即控制多路输出口输出
imm_outport_bit	立即控制单路输出口输出
imm_config_jet_ports	立即配置喷射阀轨迹位置比较输出参数
imm_start_points_jet	立即配置比较输出点绝对坐标写入硬件 fifo
imm_stop_jet	立即清除比较输出点硬件 fifo

8.2 重点说明

8.2.1 多卡连接

P2 系列控制器出厂的 IP 地址为：192.168.192.34，控制器 ID 为：0。当需要连接 2 张 P2 系列时，需要修改第 2 张控制器的 IP 地址和控制器 ID，IP 地址需要更 PC 在同一地址段，控制器 ID 不能重复。P2 系列控制器最多可以同时连接 4 张。

调用示例

```
第一步：电脑连接上需要扩展的P2系列卡，调用以下代码修改IP/ID。注意：仅连接1张控制卡
if (0 == init_api())//以默认IP和ID初始化运动控制器
{
    if (0 == command_set_mc_ip_addr("192.168.192.35", 1))//修改IP地址以及控制器ID
    {
        AfxMessageBox(_T("控制器IP/ID修改成功!"));
    }
}
第二步：控制器断电，并将0号控制器和1号控制都连接到PC，调用以下代码实现控制器初始化
if (0 == init_api("192.168.192.34", 0))//初始化0号运动控制器
{
    AfxMessageBox(_T("0号控制器初始化成功!"));
}
if (0 == init_api("192.168.192.35", 1))//初始化1号运动控制器
{
    AfxMessageBox(_T("1号控制器初始化成功!"));
}
第三步：通过指令函数的idMc参数控制对应的控制器。如以下代码所示读不同控制器的状态
McAllStates allStatus1, allStatus2;
get_mc_all_states(&allStatus1, 0);//读取0号运动控制器所有状态
get_mc_all_states(&allStatus2, 1);//读取1号运动控制器所有状态
```

指令说明

command_set_mc_ip_addr

```
//-------------------------------------------------
// 输入参数：ipAddr   -ip地址;
//         : idMC      -控制器ID
// 输出参数:
// 返回值：0-成功 -1失败
// 功能描述：设置控制器的ip地址以及控制器ID
//-------------------------------------------------
注意：P2系列运动控制器默认IP为192.168.192.34，与之连接的电脑网卡IP需要在同一网段
```

8.2.2 喷射阀控制

P2 系列控制器对喷射阀控制分 3 种方案。其一喷射阀频率由胶阀控制器控制，P2 系列仅负责开启和关

闭胶阀控制器。其二是喷射阀频率由 P2 系列控制，进而实现矢量位置比较输出或者定频输出。

调用示例：喷射阀频率由胶阀控制器控制

```
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
outport_bit(1, 1); //打开1号输出口，控制胶阀控制器启动
fast_line2(1, 20000, 2, 0); //轨迹运动
fast_line2(1, 0, 2, 20000);
outport_bit(1, 0); //关闭1号输出口，控制胶阀控制器停止
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
```

调用示例：喷射阀频率内部脉冲位置比较输出

```
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
config_jet_ports(0x01); //将1号输出口配置喷射阀输出口
set_jet_track_params(10000, 10000, 5); //设置喷射阀控制参数为：开阀时间10ms、关阀时间10ms、打点间距5mm
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
start_track_jet();//启动矢量位置比较输出
fast_line2(1, 20000, 2, 0); //轨迹运动
fast_line2(1, 0, 2, 20000);
stop_jet();//关闭喷射阀
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
```

调用示例：喷射阀频率编码器位置比较输出

第一步：配置轨迹输出模式，建议在系统配置界面完成

调用举例：

```
int mode = 1;//轨迹运动喷射阀输出模式，与界面关联
int source = 1; //轨迹运动喷射阀输出比较源，与界面关联
int offset = 1000; //轨迹运动喷射阀输出位置比较误差，与界面关联
int ch = 2;   //通道号
switch_cmd_type(CMD_TYPE_IMM, ch);
config_track_jet(_mode, source, offset, ch);
```

第二步：轨迹运动控制：加工一条直线，起点(1000,1000,1000,1000,1000)，终点（1500,1500,1500,1500,1500），每间隔 100 脉冲输出 1 次，开阀时间 10 毫秒。

① 启动加工时，开启线程写入位置比较坐标点到硬件 FIFO，注意：FIFIO 空間最大 2000 DWORD，该 FIFO 为循环存储模式，通过判断剩余空间可无限写入数据

```
int *datas = new int[51 * 6];
int i = 0;
double microDis = 100;
int ch = 1;
```

```
    for (i = 0; i <= 50; i++)
    {
        datas[i * 6] = 0;//轨迹起点以及中间点标记位置0
        if(i == 50)
            datas[i * 6] = 1;//轨迹终点标记位置1
        datas[i * 6 + 1] = 1000 + microDis * i;
        datas[i * 6 + 2] = 1000 + microDis * i;
        datas[i * 6 + 3] = 1000 + microDis * i;
        datas[i * 6 + 4] = 1000 + microDis * i;
        datas[i * 6 + 5] = 1000 + microDis * i;
    }
    get_mc_all_states(&g_cAllStatus);
    if(2000-g_cAllStatus.FifoDatasLeftover[ch] > 51*6*2)//检测FIFO空間并写入数据
    {
        set_compare_datas(30, datas, ch);//将51个点位置比较点写入FIFO
        set_compare_datas(21, &datas[30*6], ch);

        set_compare_datas(30, datas, ch);//将51个点位置比较点写入FIFO
        set_compare_datas(21, &datas[30*6], ch);
    }
    ②  主线程发送运动指令
    switch_cmd_type(CMD_TYPE_BAT, ch);
    set_channel_work_axis_group(0x1F, 0x1F, 1, ch);
    set_vector_profile(0, 200, 0, 2000, ch);
    fast_abs_line5(1, 1000, 2, 1000, 3, 1000,4, 1000, 5, 1000, ch);
    c_set_track_speed(0, 10, 0, 5000, ch);
    config_jet_ports(0x03, ch);
    set_jet_track_params(10000, 2000, 1, ch);
    start_track_jet(ch);
    curve_begin(ch);
    fast_line5(1, 500, 2, 500, 3, 500, 4, 500, 5, 500, -1, ch);
    curve_end(ch);
```

指令说明

config_jet_ports

```
int config_jet_ports(int portsBits, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：portsBits  -输出口BIT0~BIT7对应输出口1~8，1-表示关联，0-表示不关联
//         ：uCmdCh     -通道号,取值:0、1、2、3
//         ：idMC        -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：配置喷射阀输出口
```

```
//-------------------------------------------------
```

注意：P2系列最多可以控制主卡上8路输出口按照完全相同的开/关频率和打点间距进行喷射阀控制

set_jet_track_params

```
int set_jet_track_params(int openTimeUs, int closeTimeUs, float disDots, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：openTimeUs-开阀时间，单位us
//          : closeTimeUs   -关阀时间，单位us
//          : disDots       -打点间距，单位毫米
//          : uCmdCh        -通道号，取值:0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置喷射阀的轨迹位置比较输出的参数
//-------------------------------------------------
```

config_track_jet

```
int config_track_jet(int mode, int source, int offset, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 函数名称：config_track_jet
// 输入参数：mode  -轨迹位置比较输出模式 0-虚拟轴模式，1-矢量位置模式
//          : source -矢量位置模式下，位置比较源 0-编码器，1-内部脉冲
//          : offset -比较点位置误差
//          : uCmdCh     -通道号,0或者1
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：配置喷射阀轨迹位置比较输出参数。
//-------------------------------------------------
```

set_compare_datas

```
int set_compare_datas(int dotCount, int* dotPos, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：dotCount   -比较输出点数量，单次最多写入30个比较点，每个点占6个int空间
//          : dotPos   -比较输出点绝对坐标，每6个int对应1个点，每个点对应数据如下：
//                                [0]:标记位，0-轨迹起点和中间点，1-轨迹结束点
//                                [1]~[5]:XYZUV绝对坐标
//          : uCmdCh     -通道号,0或者1
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：将比较输出点绝对坐标写入硬件fifo。
//-------------------------------------------------
```

reset_compare_datas

```
int reset_compare_datas(int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
```

```
//          : uCmdCh      -通道号,0或者1
//          : idMC        -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：清除比较输出点硬件fifo
//------------------------------------------------
```

start_track_jet

```
int start_track_jet(int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：uCmdCh   -通道号, 取值:0、1、2、3
//          : idMC        -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：开始按照位置比较输出喷射阀控制
//------------------------------------------------
```

stop_jet

```
int stop_jet(int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：uCmdCh      -通道号, 取值:0、1、2、3
//          : idMC          -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：停止喷射阀输出
//------------------------------------------------
```

调用示例：轨迹运动中修改喷射位置比较距离

```
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
config_jet_ports(0x01); //将1号输出口配置喷射阀输出口
set_jet_track_params(10000, 10000, 5); //设置喷射阀控制参数为：开阀时间10ms、关阀时间10ms、打点间距5mm
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
start_track_jet();//启动矢量位置比较输出
fast_line2(1, 20000, 2, 0); //轨迹运动
change_jet_track_dis(10); //将打点间距修改为10mm
fast_line2(1, 0, 2, 20000); //轨迹运动
stop_jet();//关闭喷射阀
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
```

指令说明

change_jet_track_dis

```
int change_jet_track_dis(float disDots, int uCmdCh = 0, int idMc = 0);
//------------------------------------------------
// 输入参数：disDots    -打点间距，单位毫米
//          : uCmdCh      -通道号, 取值:0、1、2、3
```

```
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：轨迹中改变打点间距的距离
//--------------------------------------------------
```

注意：在轨迹运动中修改位置比较输出距离后，喷射阀的开阀时间不会发生改变。

调用示例：轨迹运动中喷射阀定频输出

```
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
config_jet_ports(0x01); //将1号输出口配置喷射阀输出口
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
start_points_jet(10000, 10000, 0);//启动喷射阀按固定频率输出，开阀时间10ms、关阀时间10ms
fast_line2(1, 20000, 2, 0); //轨迹运动
fast_line2(1, 0, 2, 20000); //轨迹运动
stop_jet();//关闭喷射阀
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
```

指令说明

start_points_jet

```
int start_points_jet(int openTimeUs, int closeTimeUs, int counterDots, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：openTimeUs-开阀时间，单位微秒
//          : closeTimeUs     -关阀时间，单位微秒
//          : counterDots     -打点数量，<=0表示一直输出
//          : uCmdCh      -通道号，取值:0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：控制喷射阀按照指定频率输出
//--------------------------------------------------
```

注意：打点数量参数设置为0时，控制器会按照开/关阀时间持续输出，直到调用stop_jet后停止输出

调用示例：喷射阀打点指定次数后再执行后续动作

```
    switch_cmd_type(CMD_TYPE_BAT); //切换为批处理模式
    config_jet_ports(0x01); //将1号输出口配置喷射阀输出口
     c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
    start_points_jet(10000, 10000, 100);//启动喷射阀按照开10ms关10ms的频率执行100次
    wait_jet_done();//等待100次喷射阀输出完成后，再执行后续指令
     curve_begin();//轨迹速度前瞻开始
     fast_line2(1, 20000, 2, 0); //轨迹运动
     fast_line2(1, 0, 2, 20000); //轨迹运动
     curve_end();//轨迹速度前瞻结束
```

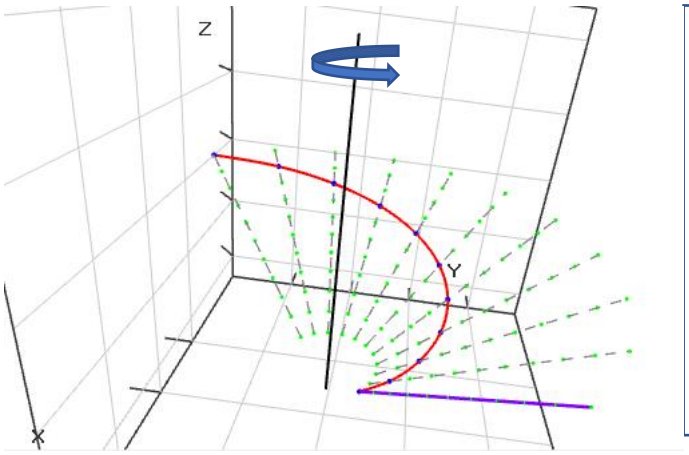
指令说明
wait_jet_done

```
int wait_jet_done(int uCmdCh = 0, int idMc = 0);
//------------------------------------------------------
// 输入参数：uCmdCh     -通道号，取值:0、1、2、3
//          : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：等待喷射阀执行完成后，再执行后续指令
//------------------------------------------------------
```

注意：该指令配合start_points_jet使用，当start_points_jet设置的打点数量参数不为0时，可通wait_jet_done指令截断后续指令的输出，直到start_points_jet指令执行完成

8.2.3 四轴/五轴轨迹运动配合喷射阀矢量位置比较输出

高自由度点胶设备（3 自由度以上），机构带动喷射阀运动的矢量位移与零件上需要点胶的路径矢量长度是不一致的（如下图所示）。本章介绍如何通过 P2 系列控制器运动指令实现高维度空间轨迹的矢量位置比较输出。



左图表述了 4 自由度机构加工一条直线的插补运动轨迹。
蓝色直线：零件点胶路径
红色曲线：机构插补路径

蓝线长度<红线长度

实现如上图加工场景需要执行 3 个步骤。首先将点胶轨迹在零件坐标系进行拆分，其次通过机械结构的轴数学模型将拆分后的零件坐标系位置信息转换到机械坐标系，最后将转换后的机械坐标系信息转化为 P2 系列运动指令序列下发到运动控制器执行。

前 2 个步骤为轨迹规划过程，不同结构的设备轨迹规划方法不同，本文不做具体描述。本文仅针对轨迹规划后的运动控制进行说明。如上图所示的 4 轴轨迹运动实现矢量位置比较输出指令调用如下：

调用示例

```
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
config_jet_ports(0x01); //将1号输出口配置喷射阀输出口
set_jet_track_params(10000, 10000, 5); //设置喷射阀控制参数为：开阀时间10ms、关阀时间10ms、打点间距5mm
```

```
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
start_track_jet();//启动矢量位置比较输出
fast_line4(1, 100, 2, 100, 3, 20, 4, 2, 0.37); //按照轨迹规划生成轨迹运动指令序列
fast_line4(1, 100, 2, 100, 3, 5, 4, 2, 0.37);
...
fast_line4(1, 100, 2, 100, 3, 10, 4, 2, 0.37);
stop_jet();//关闭喷射阀
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
```

注意：fast_line4指令的vectorLength参数设置为0.37，表示当前插补运动对应的胶头相对零件的实际运动距离，该值为零件坐标系下轨迹拆分后的矢量长度。在实际胶路长度与插补运动矢量长度不一致的场景下，如果不设置正确的实际胶路长度，矢量位置比较输出功能则无法正确执行

指令说明

fast_line4

```
int fast_line4(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//          : ch1      -1号轴轴号
//          : dis1      -1号轴运动位置，单位：脉冲
//          : ch2      -2号轴轴号
//          : dis2      -2号轴运动位置，单位：脉冲
//          : ch3      -3号轴轴号
//          : dis3      -3号轴运动位置，单位：脉冲
//          : ch4      -4号轴轴号
//          : dis4      -4号轴运动位置，单位：脉冲
//          : vectorLength      -该微段的矢量长度，单位：毫米。取值小于0，系统按照各轴分量的平方和开方来计算
//          : uCmdCh      -通道号，取值:0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：4轴插补指令，相对位置指令
//-------------------------------------------------
```

注意：指令的vectorLength参数表示当前插补运动对应的胶头相对零件的实际运动距离。该值为零件坐标系下轨迹拆分后的矢量长度，如果保持默认值-1，则系统按照各轴分量的平方和开方来计算

fast_line5

```
int fast_line5(int ch1, int dis1, int ch2, int dis2, int ch3, int dis3, int ch4, int dis4, int ch5, int dis5,float vectorLength = -1.0f, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//          : ch1      -1号轴轴号
//          : dis1      -1号轴运动位置，单位：脉冲
//          : ch2      -2号轴轴号
```

```
//          : dis2      -2号轴运动位置，单位：脉冲
//          : ch3      -3号轴轴号
//          : dis3      -3号轴运动位置，单位：脉冲
//          : ch4      -4号轴轴号
//          : dis4      -4号轴运动位置，单位：脉冲
//          : ch5      -5号轴轴号
//          : dis5      -5号轴运动位置，单位：脉冲
//          : vectorLength      -该微段的矢量长度，单位：毫米。取值小于0，系统按照各轴分量的平方和开方来计算
//          : uCmdCh      -通道号，取值:0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：4轴插补指令，相对位置指令
//-----------------------------------------------------
```

注意：指令的vectorLength参数表示当前插补运动对应的胶头相对零件的实际运动距离。该值为零件坐标系下轨迹拆分后的矢量长度，如果保持默认值-1，则系统按照各轴分量的平方和开方来计算

8.2.4 遇限位缓停

常规情况下，轴运动中硬限位有效后会执行急停，放置过度冲击对机械设备造成损伤。由于 P2 系列控制器未设计原点信号，回零运动依靠限位信号完成。在加工幅面比较大的情况下，设备上电回零需要快速找原点，此时遇硬限位时急停的功能会造成设备冲击。对此，P2 系列设计了遇限位缓停功能，在设备回零时开启该功能，回零完成后关闭此功能即可。

调用示例

```
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
set_axis_stop_mode_el(1, 1 );//1轴遇限位停止模式设置为缓停
set_profile(1, 20, 200, 20, 5000); //设置1轴回零速度
fast_hmove(1, 0);//1轴负向回零
set_axis_stop_mode_el(1, 0);//1轴遇限位停止模式设置为急停
end_package_bat_cmd();//指令打包结束
```

指令说明

set_axis_stop_mode_el

```
int set_axis_stop_mode_el(unsigned char ch, int modeStop, int uCmdCh = 0, int idMc = 0);
//-----------------------------------------------------
// 输入参数：ch-轴号，1~10号轴；
//          : modeStop    -停止模式，0-急停，1-缓停
//          : uCmdCh      -通道号，取值:0、1、2、3
//          : idMC         -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置轴遇到限位的停止模式
//-----------------------------------------------------
```

8.2.5 轨迹运动中的辅助轴控制

轨迹运动中辅助轴控制常用于挤胶轴场景（轨迹运动中出胶依靠辅助轴运动带动螺杆将胶水挤出）。

调用示例：辅助轴伴随轨迹运动同时运动

```
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
set_profile(3, 1, 5, 1, 1000); //设置3轴运动速度—此处3轴作为辅助轴
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
fast_pmove(3,9999999,0);//辅助轴开始运动，该指令第3个参数设置为0，保证后续指令不用等待当前指令结束
fast_line2(1, 20000, 2, 0); //2轴插补轨迹运动
fast_line2(1, 0, 2, 20000);
axis_sudden_stop(3,0);//辅助轴停止，该指令第2个参数设置为0，保证该停止指令可以进入批处理序列，而不是立即执行
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
```

注意：该用法是fast_pmove指令和axis_sudden_stop指令的进阶用法，指令参数含义参见对应指令说明

调用示例：辅助轴在轨迹运动中变速

```
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
set_profile(3, 1, 5, 1, 1000); //设置3轴运动速度—此处3轴作为辅助轴
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
fast_pmove(3,9999999,0);//辅助轴开始运动，该指令第3个参数设置为0，保证后续指令不用等待当前指令结束
fast_line2(1, 20000, 2, 0); //2轴插补轨迹运动
change_axis_speed(3, 10); //将辅助轴3轴运动速度调整为10mm/s
fast_line2(1, 0, 2, 20000);
axis_sudden_stop(3,0);//辅助轴停止，该指令第2个参数设置为0，保证该停止指令可以进入批处理序列，而不是立即执行
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
```

指令说明

change_axis_speed

```
int change_axis_speed(int ch, float speedNewMM, int uCmdCh = 0, int idMc = 0);
//-----------------------------------------------
// 输入参数
//          : ch            -轴号
//          : speedNewMM    -新速度,单位mm/s
//          : uCmdCh        -通道号，取值：0、1、2、3
//          : idMC          -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
```

```
// 功能描述：轴运动变速
//-------------------------------------------------
```

8.2.6 轨迹运动暂停/恢复/急停/缓停

本章介绍轨迹运动中如何实现暂停、暂停恢复、停止以及缓停，以及在暂停状态下如何处理功能输出口。

调用示例：仅轴运动

```
//-------------------------------------------------
//发送仅包含轴运动的指令序列
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
fast_line2(1, 20000, 2, 0); //2轴插补轨迹运动
fast_line2(1, 0, 2, 20000);
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
//-------------------------------------------------

void CLibDemoDlg::OnBnClickedStop()//急停
{
    motion_sudden_stop();
}

void CLibDemoDlg::OnBnClickedDecelStop()//缓停
{
    motion_decel_stop();
}

void CLibDemoDlg::OnBnClickedPause()//暂停
{
    motion_pause();
}

void CLibDemoDlg::OnBnClickedResume()//恢复
{
    motion_resume();
}
```

指令说明

motion_pause

```
int motion_pause(int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
```

```
// 输入参数
//          : uCmdCh     -通道号，取值0、1、2、3
//          : idMC        -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：加工暂停
//--------------------------------------------------
```

motion_resume

```
int motion_resume(int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数
//          : uCmdCh     -通道号，取值0、1、2、3
//          : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：加工恢复
//--------------------------------------------------
```

motion_sudden_stop

```
int motion_sudden_stop(int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数
//          : uCmdCh     -通道号，取值0、1、2、3
//          : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：指定通道运动急停
//--------------------------------------------------
```

motion_decel_stop

```
int motion_decel_stop(int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数
//          : uCmdCh     -通道号，取值0、1、2、3
//          : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：指定通道运动缓停
//--------------------------------------------------
```

调用示例：轴运动+通用输出口

```
//--------------------------------------------------
//发送轴运动+通用输出的指令序列
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
```

```
outport_bit(1, 1); //打开1号输出口，控制胶阀控制器启动
fast_line2(1, 20000, 2, 0); //轨迹运动
fast_line2(1, 0, 2, 20000);
outport_bit(1, 0); //关闭1号输出口，控制胶阀控制器停止
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
//-------------------------------------------------

void CLibDemoDlg::OnBnClickedStop()//急停
{
    motion_sudden_stop();
    outport_bit(1, 0); //关闭1号输出口
}

void CLibDemoDlg::OnBnClickedDecelStop()//缓停
{
    motion_decel_stop();
    outport_bit(1, 0); //关闭1号输出口
}

void CLibDemoDlg::OnBnClickedPause()//暂停
{
    //切换为立即指令模式，并配置暂停时对应输出口的状态。注：该段代码可以在应用软件的初始化阶段完成
    switch_cmd_type(CMD_TYPE_IMM);
    set_pause_outports_ctrl(0x01, 0x00, 0x00, 0x00, 0x00, 0x00);

    motion_pause();
}

void CLibDemoDlg::OnBnClickedResume()//恢复
{
    motion_resume();
}
```

指令说明

set_pause_outports_ctrl

```
int set_pause_outports_ctrl(unsigned int mainBoardMask, unsigned int mainBoardState, unsigned int
extendBoardMask,unsigned int extendBoardState, unsigned int smcBoardMask, unsigned int smcBoardState, int
uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：mainBoardMask  -BITD0~BITD31位表示主卡输出口1~32路输出口选择，1表示选择修改；
//          : mainBoardState  -选择的对应端口的状态,0-暂停时关闭，1-暂停时打开
//          : extendBoardMask - BITD0~BITD31位表示扩展EA3232D输出口1~32路输出口选择，1表示选择修改；
//          : extendBoardState  -选择的对应端口的状态,0-暂停时关闭，1-暂停时打开
```

```
//          : smcBoardMask - BITD0~BITD31位表示SMC3232D输出口1~32路输出口选择，1表示选择修改；
//          : smcBoardState  -选择的对应端口的状态,0-暂停时关闭，1-暂停时打开
//          : uCmdCh      -通道号
//          : idMC        -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置暂停时输出口的状态，恢复时自动回复为原来的状态
//------------------------------------------------
```

调用示例：轴运动+喷射阀控制

```
//------------------------------------------------
//发送轴运动+喷射阀控制的指令序列
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
config_jet_ports(0x01); //将1号输出口配置喷射阀输出口
set_jet_track_params(10000, 10000, 5); //设置喷射阀控制参数为：开阀时间10ms、关阀时间10ms、打点间距5mm
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
start_track_jet();//启动矢量位置比较输出
fast_line2(1, 20000, 2, 0); //轨迹运动
fast_line2(1, 0, 2, 20000);
stop_jet();//关闭喷射阀
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
//------------------------------------------------

void CLibDemoDlg::OnBnClickedStop()//急停
{
    motion_sudden_stop();
    stop_jet();//关闭喷射阀
}

void CLibDemoDlg::OnBnClickedDecelStop()//缓停
{
    motion_decel_stop();
    stop_jet();//关闭喷射阀
}

void CLibDemoDlg::OnBnClickedPause()//暂停
{
    motion_pause();
}

void CLibDemoDlg::OnBnClickedResume()//恢复
{
```

```
    motion_resume();
}
```

8.2.7 条件暂停

条件暂停功能是指：批处理指令序列中，P2 系列解析到条件暂停时自动暂停后续指令输出，仅当设置的输入口状态满足条件后，方可执行后续指令。

调用示例

```
//------------------------------------------------------
//发送包含条件暂停的指令序列
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
outport_bit(1, 1); //打开1号输出口，控制胶阀控制器启动
fast_line2(1, 20000, 2, 0); //轨迹运动
fast_line2(1, 0, 2, 20000);
outport_bit(1, 0); //关闭1号输出口，控制胶阀控制器停止
curve_end();//轨迹速度前瞻结束
wait_inport_condition(1,1,1000,0xfc); //条件暂停。等待输入口1有效，等待超时时间：1000ms，超时后控制器抛出错误码：0xfc。输入口1有效后执行后续运动
curve_begin();//轨迹速度前瞻开始
outport_bit(1, 1); //打开1号输出口，控制胶阀控制器启动
fast_line2(1, 20000, 2, 0); //轨迹运动
fast_line2(1, 0, 2, 20000);
outport_bit(1, 0); //关闭1号输出口，控制胶阀控制器停止
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
//------------------------------------------------------

//通过定时器或线程监控错误码
void CLibDemoDlg::OnTimer(UINT_PTR nIDEvent)
{
     McAllStates allStatus;
    get_mc_all_states(&allStatus);//读取所有状态
    if(allStatus.ErrorCodeArm1 == 0xfc) //监控到条件暂停等待超自定义的错误码
    {
        //do samething…
        AfxMessageBox(_T("err:条件暂停等待超时!"));
        set_mc_error_code(1,0x00); //清除历史错误码
    }
}
```

指令说明
wait_inport_condition

int wait_inport_condition(int inportsIndex, int waitStatus, int timeOutMs = 0, int timeOutErrorCode = 0, int uCmdCh = 0, int idMc = 0);
//---
// 输入参数
//　　: inportsIndex　　-等待的输入口端口号，1~32表示主卡，32~64表示EA扩展卡，65~96表示smc扩展卡
//　　: waitStatus　-等待的输入口状态，1-有效，0-无效
//　　: timeOutMs　　-等待超时时间，0-表示一直等待,单位毫秒
//　　: timeOutErrorCode　-超时后的错误码
//　　: uCmdCh　　-通道号，取值：0、1、2、3
//　　: idMC　　-控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：等待输入口条件有效后再执行后续指令
//---
注意：该指令不能在前瞻运动指令之间调用

set_mc_error_code

int set_mc_error_code(int errorType, unsigned int errorcode, int idMc = 0);
//---
// 输入参数：errorType -错误码类型，0-Arm0错误码,1-Arm1错误码；
//　　: errorcode -待设置错误码
//　　: idMC　　-控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：设置控制器的错误码
//---

8.2.8 延时功能

延时功能是指：批处理指令序列中，P2 系列解析到延时指令时自动暂停后续指令输出并延时，仅当延时计时满足条件后，方可执行后续指令。

调用示例

switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
outport_bit(1, 1); //打开1号输出口，控制胶阀控制器启动
fast_line2(1, 20000, 2, 0); //轨迹运动
fast_line2(1, 0, 2, 20000);
outport_bit(1, 0); //关闭1号输出口，控制胶阀控制器停止
curve_end();//轨迹速度前瞻结束
delay_time(1000); //延时1000ms。延时计时完成后执行后续运动

```
curve_begin();//轨迹速度前瞻开始
outport_bit(1, 1); //打开1号输出口，控制胶阀控制器启动
fast_line2(1, 20000, 2, 0); //轨迹运动
fast_line2(1, 0, 2, 20000);
outport_bit(1, 0); //关闭1号输出口，控制胶阀控制器停止
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
```

指令说明

delay_time

```
int delay_time(int delayMs, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//          : delayMs    -延时时间，单位毫秒
//          : uCmdCh    -通道号，取值：0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：延时指令
//-------------------------------------------------
```

注意：该指令不能在前瞻运动指令之间调用

8.2.9 轴硬限位&轴报警使能与禁止

调用示例

```
switch_cmd_type(1); //切换为批处理模式
begin_package_bat_cmd();//指令打包
enable_axis_el(1,1,0); //关闭1轴的硬件正限位功能
enable_axis_el(1,0,0); //关闭1轴的硬件负限位功能
enable_axis_alarm(1,0); //关闭1轴的轴报警功能
end_package_bat_cmd();//指令打包结束
```

指令说明

enable_axis_el

```
int enable_axis_el(int ch, int elPorN, int enable, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//          : ch            -轴号
//          : elPorN        -正限位或负限位，0-负限位，1-正限位
//          : enable        -使能或者禁止，0-禁止，1-使能
//           : uCmdCh      -通道号，取值：0、1、2、3
//          : idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：使能轴的限位功能
//-------------------------------------------------
```

enable_axis_alarm

```
int enable_axis_alarm(int ch,int enable, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//          : ch            -轴号
//          : enable        -使能或者禁止，0-禁止，1-使能
//           : uCmdCh      -通道号，取值：0、1、2、3
//          : idMC          -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：使能轴的报警功能
//-------------------------------------------------
```

8.2.10 输出口开启后自动延时关闭

在轨迹运动中，如果需要打开某路输出口，并在指定延时后自动关闭，且该输出口动作不能影响轨迹的运动的连续性时可以采用此功能。

调用示例

```
switch_cmd_type(CMD_TYPE_BAT); //切换为批处理模式
begin_package_bat_cmd();//指令打包
c_set_track_speed(20, 200, 20, 5000); //设置轨迹速度
curve_begin();//轨迹速度前瞻开始
outport_bit(1, 1); //打开1号输出口，控制胶阀控制器启动
fast_line2(1, 20000, 2, 0); //轨迹运动
outport_bit(3, 1); //打开3号输出口
lag_outport_bit(3, 0, 3000); //3000ms后3号输出口关闭。后续指令继续输出
fast_line2(1, 0, 2, 20000);
outport_bit(1, 0); //关闭1号输出口，控制胶阀控制器停止
curve_end();//轨迹速度前瞻结束
end_package_bat_cmd();//指令打包结束
```

注意：lag_outport_bit指令不会影响其前后运动指令的速度前瞻规划。如需要同时操作多路输出口可调用lag_outport_byte指令

指令说明

lag_outport_bit

```
int lag_outport_bit(int portIndex, int state,int lagTimeMs, int uCmdCh = 0, int idMc = 0);//----------
//-------------------------------------------------
// 输入参数：portIndex  -输出口端口号，1~32路：主卡1~32路,33~64表示EA3232D对应输出口，65~96表示SMC3232D输出口
//           : state      -输出口状态，0-关闭，1-打开
//           : lagTimeMs -滞后时间，单位ms，在滞后过程中，后续指令继续输出
//           : uCmdCh     -通道号，取值：0、1、2、3
//          : idMC         -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
```

```
// 功能描述：控制单路输出口状态，滞后控制，滞后过程中，后续指令继续出去
//--------------------------------------------------
```

lag_outport_byte

```
int lag_outport_byte(unsigned int mainBoardPorts, unsigned int mainBoardState, unsigned int extendBoardPorts,
unsigned int extendBoardState, unsigned int smcBoardPorts, unsigned int smcBoardState, int lagTimeMs, int
uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：mainBoardPorts  -BITD0~BITD31位表示主卡输出口1~32路输出口选择，1表示选择修改；
//          ：mainBoardState  -选择的对应端口的状态；
//          ：extendBoardPorts - BITD0~BITD31位表示扩展EA3232D输出口1~32路输出口选择，1表示选择修改；
//          ：extendBoardState  -选择的对应端口的状态；
//          ：smcBoardPorts - BITD0~BITD31位表示SMC3232D输出口1~32路输出口选择，1表示选择修改；
//          ：smcBoardState  -选择的对应端口的状态；
//          ：lagTimeMs - 滞后时间，单位毫秒
//          ：state      -输出口状态，0-关闭，1-打开
//          ：uCmdCh     -通道号，取值：0、1、2、3
//          ：idMC       -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：控制多路输出口状态，滞后控制，滞后过程中，后续指令继续出去
//--------------------------------------------------
```

8.2.11 强制立即指令

本章节的所有的函数均为强制立即指令，可在立即模式或者批处理模式立刻进行使用，无需切换模式。在使用 imm_fast_pmove 和 imm_fast_pmove_s 请确保当前轴为静止状态。

指令说明

imm_set_profile

```
int imm_set_profile(unsigned char ch, float startSpeedMM, float HighSpeedMM, float endSpeedMM, float
accSpeedMM, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//      ch - 轴号，1~10 号轴；
//      startSpeedMM - 起始速度，单位 mm/s
//      HighSpeedMM  - 目标速度，单位 mm/s
//      endSpeedMM   - 终止速度，单位 mm/s
//      accSpeedMM   - 加速度，单位 mm/s
//      uCmdCh    - 通道号,取值：0、1、2、3
//      idMC       -控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置轴回零和点位运动速度(T 型速度曲线)
//-------------------------------------------------
```

imm_set_profile_s

```
int imm_set_profile(unsigned char ch, float startSpeedMM, float HighSpeedMM, float endSpeedMM, float
accSpeedMM,float accaccSpeedMM, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//      ch - 轴号，1~10 号轴；
//      startSpeedMM - 起始速度，单位 mm/s
//      HighSpeedMM  - 目标速度，单位 mm/s
//      endSpeedMM   - 终止速度，单位 mm/s
//      accSpeedMM   - 加速度，单位 mm/s
//      accaccSpeedMM - 加加速度，单位 mm/s
//      uCmdCh    - 通道号,取值：0、1、2、3
//      idMC       -控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：设置轴回零和点位运动速度（S 型速度曲线）
//-------------------------------------------------
```

imm_fast_pmove

```
int imm_fast_pmove(int ch1, int dis1,int isWaitDoneThenNextCmd =1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//      ch1 - 轴号，1~10 号轴；
//      dis1 - 相对运动位移值，单位，脉冲
//      isWaitDoneThenNextCmd -是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//      uCmdCh  - 通道号,取值：0、1、2、3
//      idMC   - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：相对位置模式下的单轴点位运动(T型速度曲线)
//-------------------------------------------------
```

imm_fast_pmove_s

```
int imm_fast_pmove_s(int ch1, int dis1,int isWaitDoneThenNextCmd =1, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数
//      ch1 - 轴号，1~10 号轴；
//      dis1 - 相对运动位移值，单位，脉冲
//      isWaitDoneThenNextCmd -是否等当前指令运动完成后再加载下一条指令，1-等待，0-不等待，直接加载下一条指令
//      uCmdCh  - 通道号,取值：0、1、2、3
//      idMC   - 控制器 ID
// 输出参数：
// 返回值：0-成功，其它-失败
// 功能描述：相对位置模式下的单轴点位运动(S型速度曲线)
//-------------------------------------------------
```

imm_outport_bit

```
int imm_outport_bit(int portIndex, int state, int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//      portIndex - 输出口端口号，1~32：主卡,33~64 表示 EA3232D 对应 Out17~Out48 号输出口，65~96 表示 SMC3232D
//      对应的 OUT1~OUT32 号输出口
//      state     - 输出口状态，0-关闭，1-打开
//       uCmdCh    - 通道号,取值：0、1、2、3
//      idMC       -  控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：控制单路输出口输出
//-------------------------------------------------
```

imm_outport_byte

```
int imm_outport_byte(unsigned int mainBoardPorts, unsigned int mainBoardState,unsigned int extendBoardPorts, unsigned int extendBoardState,   unsigned int smcBoardPorts, unsigned int smcBoardState,int uCmdCh = 0, int idMc = 0);
//-------------------------------------------------
// 输入参数：
//      mainBoardPorts  - 按位表示 1~32 路主卡输出口是否修改，1-修改,0-不修改；P2 系列控制器主卡 8 路输出
//      mainBoardState  - 按位表示主卡对应输出口修改后的状态，0-关闭，1-打开；
```

```
//       extendBoardPorts - 按位表示扩展板 EA3232D 的 OUT17~OUT48 路输出口是否修改，1-修改,0-不修改；
//       extendBoardState  - 按位表示 EA3232D 对应输出口修改后的状态，0-关闭，1-打开；
//       smcBoardPorts -  按位表示扩展板 SMC3232D 的 1~32 路输出口是否修改，1-修改,0-不修改；
//      smcBoardState  - 按位表示 SMC3232D 对应输出口修改后的状态，0-关闭，1-打开；
//       uCmdCh     - 通道号,取值：0、1、2、3
//       idMC      - 控制器 ID
// 输出参数：
// 返回值：0-成功 -1 失败
// 功能描述：控制多路输出口状态
//--------------------------------------------------
```

imm_config_jet_ports

```
int imm_config_jet_ports(int portsBits, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：portsBits  -输出口BIT0~BIT7对应输出口1~8，1-表示关联，0-表示不关联
//         : uCmdCh     -通道号,取值:0、1、2、3
//         : idMC        -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：配置喷射阀输出口
//--------------------------------------------------
```

注意：P2系列最多可以控制主卡上8路输出口按照完全相同的开/关频率和打点间距进行喷射阀控制

imm_start_points_jet

```
int imm_start_points_jet(int openTimeUs, int closeTimeUs, int counterDots, int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：openTimeUs-开阀时间，单位微秒
//         : closeTimeUs    -关阀时间，单位微秒
//         : counterDots    -打点数量，<=0表示一直输出
//         : uCmdCh     -通道号，取值:0、1、2、3
//         : idMC      -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：控制喷射阀按照指定频率输出
//--------------------------------------------------
```

注意：打点数量参数设置为0时，控制器会按照开/关阀时间持续输出，直到调用stop_jet后停止输出

imm_stop_jet

```
int imm_stop_jet(int uCmdCh = 0, int idMc = 0);
//--------------------------------------------------
// 输入参数：uCmdCh     -通道号，取值:0、1、2、3
//         : idMC        -控制器ID
// 输出参数：
// 返回值：0-成功 -1失败
// 功能描述：停止喷射阀输出
//--------------------------------------------------
```