总线脉冲控制卡

编程手册

HOURS

www.hours-shop.com

前 言

尊敬的客户：

欢迎选用本公司运动控制卡

为了回报客户，我们将提供一流的运动控制卡，高效的技术支持，完善的售后服务，帮助你建立自己的控制系统。

本手册主要对运动控制卡的软件应用等内容作全面介绍，提供了使用运动控制卡所需知识及注意事项，请在熟知本产品的安全注意事项后使用。

由于产品的改进，规格、编写版本的变更等原因，会有适当的改动，恕不另行通知。

不承担由于使用手册或产品不当，所造成直接的、间接的、特殊的、附带的或相应产生的损失或责任。

如果您需要技术支持、服务与相关信息，或者您对于使用产品的过程中有任何问题，请与我们联系。我们期待能够依照您的需求，提供我们最好的技术支持和服务。

目录

运动控制函数列表

表 1-1

指令列表	说明
1 控制卡打开函数	
MCF_Set_Switch_State_Net	网络并联模式设置函数
MCF_Open_Net	打开运动控制卡
MCF_Get_Open_Net	读取打开控制卡参数
MCF_Close_Net	关闭运动控制卡
MCF_Set_Link_TimeOut_Net	链接超时紧急停止所有轴 DO 输出 TimeOut_Output
MCF_Get_Link_TimeOut_Net	获取链接中断发生的次数
MCF_Set_Trigger_Output_Bit_Net	链接超时紧急停止触发使能函数
MCF_Get_Link_State_Net	链接监测
2 通用输入输出函数	
MCF_Set_Output_Net	设置对应模块数字 IO 输出状态
MCF_Get_Output_Net	读取对应模块数字 IO 输出状态
MCF_Set_Output_Bit_Net	按位设置对应模块数字 IO 输出状态
MCF_Get_Output_Bit_Net	按位读取对应模块数字 IO 输出状态
MCF_Set_Output_Time_Bit_Net	通用 IO 输出复用：按位输出保持时间函数
MCF_Set_Output_Time_All_Net	通用 IO 输出复用：全部按位输出保持时间函数
MCF_Set_Compare_Output_Bit_Net	通用 IO 输出复用：按位置输出保持时间函数
MCF_Get_Input_Net	读取对应模块数字 IO 输入状态
MCF_Get_Input_Bit_Net	按位读取对应模块数字 IO 输入状态
MCF_Clear_Input_Fall_Bit_Net	通用 IO 按位输入下升沿高速捕获清除函数
MCF_Get_Input_Fall_Bit_Net	通用 IO 按位输入下升沿高速捕获读取函数
MCF_Get_Input_Fall_Count_Bit_Net	通用 IO 按位输入下升沿高速计数读取函数
MCF_Open_Input_Lock_Bit_Net	通用 IO 按位输入数据锁存保持
MCF_Set_Input_Filter_Time_Bit_Net	通用 IO 按位输入滤波函数
3 轴专用输入输出函数	
MCF_Set_Servo_Enable_Net	设置轴使能状态

MCF_Get_Servo_Enable_Net	读取轴使能状态
MCF_Set_Servo_Alarm_Reset_Net	设置伺服报警复位
MCF_Get_Servo_Alarm_Reset_Net	读取伺服报警复位设置
MCF_Get_Servo_Alarm_Net	读取伺服报警电平状态
MCF_Get_Servo_INP_Net	读取编码器 Z 相输入
MCF_Get_Z_Net	读取伺服定位完成输入
MCF_Get_Home_Net	读取原点电平状态
MCF_Get_Positive_Limit_Net	读取正限位电平状态
MCF_Get_Negative_Limit_Net	读取负限位电平状态
4 轴设置函数	
MCF_Set_Pulse_Mode_Net	设置轴脉冲输出模式
MCF_Get_Pulse_Mode_Net	读取轴脉冲输出模式
MCF_Set_Position_Net	设置轴规划位置
MCF_Get_Position_Net	读取轴规划位置
MCF_Set_Encoder_Net	设置轴编码器位置
MCF_Get_Encoder_Net	读取轴编码器位置
MCF_Z_Clear_Encoder_Net	触发 Z 相信号时清除编码器值
MCF_Z_Output_Bit_Net	触发 Z 相信号后固定编码器距离输出指定 IO
MCF_Get_Vel_Net	读取实时命令速度和实际编码器速度
5 轴硬件触发停止运动函数	
MCF_Set_EMG_Bit_Net	设置通用输入作为硬件急停
MCF_Set_EMG_Output_Net	设置硬件急停后所有 DO 的输出
MCF_Set_EMG_Output_Enable_Net	通用 IO 触发恢复 DO 输出状态使能
MCF_Set_Input_Trigger_Net	通用 IO 输入复用：做为触发停止
MCF_Get_Input_Trigger_Net	读取通用 IO 输入复用：做为触发停止设置
MCF_Set_Soft_Limit_Net	设置轴软件限位
MCF_Get_Soft_Limit_Net	读取轴软件限位
MCF_Set_Soft_Limit_Enable_Net	设置软件限位开启/关闭
MCF_Get_Soft_Limit_Enable_Net	读取软件限位开启/关闭
MCF_Set_Alarm_Trigger_Net	设置伺服报警触发模式
MCF_Get_Alarm_Trigger_Net	读取伺服报警触发模式
MCF_Set_Index_Trigger_Net	设置 Index 信号触发模式
MCF_Get_Index_Trigger_Net	读取 Index 信号触发模式

MCF_Set_Home_Trigger_Net	设置原点信号触发模式
MCF_Get_Home_Trigger_Net	读取原点信号触发模式
MCF_Set_ELP_Trigger_Net	设置正限位触发模式
MCF_Get_ELP_Trigger_Net	读取正限位触发模式
MCF_Set_ELN_Trigger_Net	设置负限位触发模式
MCF_Get_ELN_Trigger_Net	读取负限位触发模式
MCF_Get_Home_Rise_Position_Net	捕捉原点上升沿位置值
MCF_Get_Home_Fall_Position_Net	捕捉原点下降沿位置值
MCF_Get_Home_Rise_Encoder_Net	捕捉原点上升沿编码器值
MCF_Get_Home_Fall_Encoder_Net	捕捉原点下降沿编码器值
MCF_Clear_Axis_State_Net	清除报警原因寄存器
MCF_Get_Axis_State_Net	查询轴状态
6 轴回原点函数	
MCF_Search_Home_dMaxA_Time_Net	设置回零参数
MCF_Search_Home_Set_Net	设置回零参数
MCF_Search_Home_Start_Net	设置回零启动
MCF_Search_Home_Stop_Net	设置回零停止
MCF_Search_Home_Get_State_Net	获取回零状态
MCF_Search_Home_Stop_Time_Net	设置回零缓停时间
MCF_Search_Home_Keep_Position_Net	设置回零完成后保持位置值
MCF_Search_Home_Keep_Encoder_Net	设置回零完成后保持编码器值
MCF_Search_Home_Leave_Vel_Net	设置回零在零点位置离开速度
7 点位运动控制函数	
MCF_JOG_Net	开始速度环控制,曲线运动参数设置
MCF_Uniaxial_dDist_Change_Net	单轴运动位置改变函数
MCF_Uniaxial_dMaxV_Change_Net	单轴运动速度改变函数
MCF_Uniaxial_dMaxA_Change_Net	单轴运动加速度改变函数
MCF_Set_Axis_Profile_Net	设置单轴曲线函数
MCF_Get_Axis_Profile_Net	获取单轴曲线设置函数
MCF_Uniaxial_Net	单轴运动函数
MCF_Set_Axis_Stop_Profile_Net	单轴停止曲线函数
MCF_Get_Axis_Stop_Profile_Net	获取单轴停止曲线设置函数
MCF_Axis_Stop_Net	轴停止函数

8 插补运动控制函数	
MCF_Set_Coordinate_Profile_Net	设置坐标系运动曲线参数
MCF_Get_Coordinate_Profile_Net	读取坐标系运动曲线参数
MCF_Arc2_Radius_Net	圆半径插补运动函数
MCF_Arc2_Centre_Net	圆圆心插补运动函数
MCF_Line2_Net	启动两轴直线插补
MCF_Line3_Net	启动三轴直线插补
MCF_Line4_Net	启动四轴直线插补
MCF_Set_Coordinate_Stop_Profile_Net	设置坐标系停止曲线函数
MCF_Get_Coordinate_Stop_Profile_Net	读取坐标系停止曲线函数
MCF_Screw3_Radius_Net	螺旋线圆半径插补运动函数
MCF_Screw3_Centre_Net	螺旋线圆圆心插补运动函数
MCF_Coordinate_Stop_Net	坐标系停止函数
9 缓冲区函数	
MCF_Buffer_Set_Stop_Profile_Net	设置缓冲区停止曲线函数
MCF_Buffer_Stop_Net	缓冲区停止函数
MCF_Buffer_Change_Velocity_Ratio _Net	缓冲区在线改变速度倍率
MCF_Buffer_Start_Net	缓冲区建立开始函数
MCF_Buffer_Set_Velocity_Ratio_Enable_Net	缓冲区速度倍率
MCF_Buffer_Set_Reduce_Ratio_Net	设置缓冲区速度倍率
MCF_Buffer_Set_Profile_Net	设置缓冲区运动参数指令
MCF_Buffer_Uniaxial _Net	缓冲区点动指令
MCF_Buffer_Sync_Follow_Net	缓冲区单轴运动距离同步跟随函数
MCF_Buffer_Line2_Net	缓冲区两轴直线插补指令
MCF_Buffer_Line3_Net	缓冲区三轴直线插补指令
MCF_Buffer_Line4_Net	缓冲区四轴直线插补指令
MCF_Buffer_Arc_Radius_Net	缓冲区半径圆弧插补指令
MCF_Buffer_Arc_Centre_Net	缓冲区圆心圆弧插补指令
MCF_Buffer_Delay _Net	缓冲区延时指令
MCF_Buffer_Set_Output_Bit _Net	缓冲区操作输出信号指令
MCF_Buffer_Wait_Input_Bit _Net	缓冲区等待输入信号指令
MCF_Buffer_End_Net	缓冲区数据写入结束指令

MCF_Buffer_Execute_Net	缓冲区启动指令
MCF_Buffer_Execute_BreakPoint_Net	缓冲区暂停恢复指令
MCF_Buffer_Get_State_Net	缓冲区查询函数
MCF_Buffer_Get_Remainder_Space_Net	缓冲区可填充指令查询
MCF_Buffer_Insert_Start_Net	缓冲区开始插入
MCF_Buffer_Insert_End_Net	缓冲区结束插入
MCF_Buffer_Count_Occupy_Space_Net	计算加入指令所占用的空间百分比
10 示波器 10K 采样频率数据捕捉函数	
MCF_Capture_Open_Net	数据捕捉打开函数
MCF_Capture_Close_Net	数据捕捉关闭函数
MCF_Capture_State_Net	数据捕捉检查数据更新函数
MCF_Capture_Read_Command_Net	读取 10K 速度采样连续 1000 个位置命令数据
MCF_Capture_Read_Encoder_Net	读取 10K 速度采样连续的 1000 个编码器数据
MCF_Capture_Read_AD_Net	读取 10K 速度采样连续的 1000 个模拟量数据
MCF_Capture_Filter_AD_Net	ADC 采样滤波
MCF_Capture_Frequency_Net	数据捕捉频率设置
MCF_Capture_Time_Net	数据捕捉缓存时间设置
11 电子齿轮控制函数	
MCF_Set_Gear_Net	设置电子齿轮参数
MCF_Get_Gear_Net	读取电子齿轮参数
MCF_Set_Gear_Enable_Net	设置电子齿轮启用状态
MCF_Get_Gear_Enable_Net	读取电子齿轮启用状态
MCF_Set_Gear_Auto_Disable_Net	电子齿轮运动距离后自动关闭
12 位置比较输出函数	
MCF_Set_Compare_Config_Net	设置一维位置比较器
MCF_Get_Compare_Config_Net	获取一维位置比较器设置
MCF_Clear_Compare_Points_Net	清除一维位置比较点
MCF_Clear_Compare_Current_Points_Net	清除第一个添加的位置比较点
MCF_Disable_Compare_Any_Points_Net	禁用指定比较点
MCF_Add_Compare_Point_Net	添加一维位置比较点
MCF_Get_Compare_Current_Point_Net	读取当前一维比较点位置
MCF_Get_Compare_Points_Runned_Net	查询已经比较过的一维比较点个数

MCF_Get_Compare_Points_Remained_Net	查询可以加入的一维比较点个数
MCF_Get_Compare_Points_Incomplete_Net	查询所有未完成一维比较点个数和位置
13 PWM 输出函数	
MCF_Set_Pwm_Config_Net	设置 PWM 输出参数
MCF_Get_Pwm_Config_Net	读取 PWM 输出参数
MCF_Set_Pwm_Output_Net	输出 PWM 信号
MCF_Get_Pwm_State_Net	PWM 完成信号
14 手轮函数	
MCF_Hand_Wheel_Open_Net	打开手轮
MCF_Hand_Wheel_Close_Net	关闭手轮
MCF_Hand_Wheel_Config_Encoder_Net	设置硬件手轮编码器通道
MCF_Hand_Wheel_Config_X1_Net	设置硬件手轮速率配置输入点
MCF_Hand_Wheel_Config_X10_Net	设置硬件手轮速率配置输入点
MCF_Hand_Wheel_Config_X100_Net	设置硬件手轮速率配置输入点
MCF_Hand_Wheel_Speed_X1_Net	设置硬件手轮 1 倍速接口对应的速率大小
MCF_Hand_Wheel_Speed_X10_Net	设置硬件手轮 10 倍速接口速率大小
MCF_Hand_Wheel_Speed_X100_Net	设置硬件手轮 100 倍速接口速率大小
MCF_Hand_Wheel_Config_Axis_Net	设置硬件手轮轴号配置输入点
MCF_Hand_Wheel_Config_Filter_Time_Net	设置手轮运动平滑滤波时间
15 模拟量输入输出函数	
MCF_Single_Read_AD_Net	读取单次 ADC 采样
MCF_Single_Write_DA_Net	读取单次 DAC 输出
MCF_Set_AD_Compare_Net	设置 AD 双向比较器停止对应轴
MCF_Set_AD_Capture_Net	设置 AD 捕获模式和数值
MCF_Clear_AD_Capture_Net	清除 AD 捕获模式和数值
MCF_Get_Capture_AD_1_Net	读取到达设置通道 1 的 AD 值，捕获通道 5 的 AD 值和轴 X 位置
MCF_Get_Capture_AD_2_Net	读取到达设置通道 2 的 AD 值，捕获通道 6 的 AD 值和轴 X 位置
MCF_Get_Capture_AD_3_Net	读取到达设置通道 3 的 AD 值，捕获通道 7 的 AD 值和轴 X 位置

MCF_Get_Capture_AD_4_Net	读取到达设置通道 4 的 AD 值，捕获通道 8 的 AD 值和轴 X 位置
MCF_Get_Capture_AD_5_Net	读取到达设置通道 5 的 AD 值，捕获通道 1 的 AD 值和轴 X 位置
MCF_Get_Capture_AD_6_Net	读取到达设置通道 6 的 AD 值，捕获通道 2 的 AD 值和轴 X 位置
MCF_Get_Capture_AD_7_Net	读取到达设置通道 7 的 AD 值，捕获通道 3 的 AD 值和轴 X 位置
MCF_Get_Capture_AD_8_Net	读取到达设置通道 8 的 AD 值，捕获通道 4 的 AD 值和轴 X 位置
MCF_Set_Position_Capture_AD_Net	设置触发位置数值
MCF_Get_Position_Capture_AD_Net	读取触发位置 AD 值
MCF_Clear_Position_Capture_AD_Net	清除触发位置对应存储的 AD 值
MCF_Get_Limit_AD_Net	获取 AD 最大值
MCF_Clear_Limit_AD_Net	清除存储的 AD 值
16 系统函数	
MCF_Get_Version_Net	模块版本号
MCF_Get_Serial_Number_Net	序列号
MCF_Get_Run_Time_Net	模块运行时间
MCF_Flash_Write_Net	写入 Flash 数据
MCF_Flash_Read_Net	读取 Flash 数据
MCF_Set_CallBack_Net	系统定时回调函数

第 1 章 初始化运动控制卡

1.1 级联模式

运动控制卡支持串联模式和并联模式：

函数原型	short **MCF_Set_Switch_State_Net** (unsigned short Mode = 0);
使用说明	网络并联模式设置函数
参数说明	Mode：级联模式； 0：串联 1：并联

设置运动控制卡对应网口：

函数原型	short **MCF_Set_Card_Number_Net** (unsigned short Card_Number = 0);
使用说明	设置运动控制卡对应网口函数
参数说明	Card_Number：控制卡对应网口号； 0：自动选择

读取运动控制卡对应网口：

函数原型	short **MCF_Get_Card_Number_Net** (unsigned short *Card_Number);
使用说明	读取运动控制卡对应网口函数
参数说明	*Card_Number：控制卡对应网口号；

1.2 打开/关闭运动控制卡

在使用运动控制卡之前，必须最先调用指令 MCF_Open_Net 配置连接模块数量和类型，并打开总线控制卡。其中：

Connection_Number 表示实际连接扩展模块的数量；

Station_Number 表示分配每个扩展模块的通讯站号，通常用数组{0,1,2……}表示，站号可以自由分配，但不可重复；

Station_Type 表示实际连接的扩展模块的类型，类型需要与接线匹配

0	NIO0808R/NIO1616R/NIO2416	8/16/24 输入 8/16/1616 输出模块
1	NIO4832/NIO3232/NIO4000	48/32/40 输入 32/32/0 输出模块

2	NMC1200R/NMC1400R/NMC3400/NMC3401/ NMC3201	2/4 轴网络总线运动控制卡
3	NMC5800/NMC5600R/NMC1800/NMC1600R	6/8 轴网络总线运动控制卡
4	NMC5120R/NMC5160/NMC1120R	12/16 轴网络总线运动控制卡
5	LMC3400/LMC3100/LMC3401/LMC3101R/ LMC5000	网络运动控制激光卡
6	EIO0840	8 路编码器 24 输入 16 输出模块
7	NAD0804/NAD0402/NAD0802/NAD0400/ NAD0800/NAD0004/NAD0002	模拟量 4 路输出 8 路输入 数字量 12 输入 12 输出
8	NIO0040	40 路输出模块

在使用完运动控制后，同时也必须调用指令 MCF_Close_Net 关闭当前运动控制卡，以便释放运动控制卡资源。

函数原型	short **MCF_Open_Net** (unsigned short Connection_Number, unsigned short *Station_Number, unsigned short *Station_Type);
使用说明	打开运动控制卡
参数说明	Connection_Number：设置站点个数 取值：大于 0 *Station_Number：依次设置站号 *Station_Type： 表示设置站点类型 取值： 0: NIO0808R/NIO1616R/NIO2416 1: NIO4832/NIO3232/NIO4000 2: NMC1200R/NMC1400R/NMC3400/NMC3401/NMC3201 3: NMC5800/NMC5600R/NMC1800/NMC1600R 4: NMC5120R/NMC5160/NMC1120R 5: LMC3400/LMC3100/LMC3401/LMC3101R/LMC5000 6: EIO0840 7: NAD0804/NAD0402/NAD0802/NAD0400/NAD0800/ NAD0004/NAD0002 8: NIO0040

函数原型	short **MCF_Get_Open_Net** (unsigned short *Connection_Number, unsigned short *Station_Number, unsigned short *Station_Type);

使用说明	读取打开控制卡参数
参数说明	*Connection_Number：站点个数 *Station_Number：对应的站号顺序 *Station_Type： 对应的站点类型 取值： 0: NIO0808R/NIO1616R/NIO2416 1: NIO4832/NIO3232/NIO4000 2: NMC1200R/NMC1400R/NMC3400/NMC3401/NMC3201 3: NMC5800/NMC5600R/NMC1800/NMC1600R 4: NMC5120R/NMC5160/NMC1120R 5: LMC3400/LMC3100/LMC3401/LMC3101R/LMC5000 6: EIO0840 7: NAD0804/NAD0402/NAD0802/NAD0400/NAD0800/ NAD0004/NAD0002 8: NIO0040

函数原型	short **MCF_Close_Net** ();
使用说明	关闭运动控制卡
参数说明	

示例：

1. 打开运动控制卡

```
short rtn;
unsigned short Station_Number[3] = {0,1,2};       //设置模块站号依次为：0,1,2
unsigned short Station_Type[3]     = {2,2,1};
//设置模块类型依次为：4 轴控制卡，4 轴控制卡，80 点 IO 模块
rtn = MCF_Open_Net (3, &Station_Number[0], &Station_Type[0]);
```

2. 关闭运动控制卡

```
rtn = MCF_Close_Net();
```

1.3 链接超时紧急停止

1.3.1 链接超时紧急停止

函数原型	short **MCF_Set_Link_TimeOut_Net** (unsigned long Time_1MS, unsigned long TimeOut_Output, unsigned short StationNumber = 0);
使用说明	链接超时紧急停止所有轴 DO 输出 TimeOut_Output
参数说明	Time_1MS：设置超时时间； 范围：[0,60000]；

	TimeOut_Output: 设置超时输出； StationNumber ： 站号；

函数原型	short **MCF_Get_Link_TimeOut_Net** (unsigned long *TimeOut_Number, unsigned short StationNumber = 0);
使用说明	获取链接中断发生的次数
参数说明	*TimeOut_Number: 链接中断次数； StationNumber ： 站号；

1.3.2 链接超时紧急停止触发使能

函数原型	short **MCF_Set_Trigger_Output_Bit_Net** (unsigned short Bit_Output_Number ,unsigned short Bit_Output_Enable, unsigned short StationNumber = 0);
使用说明	链接超时紧急停止触发使能函数
参数说明	Bit_Output_Number：选择 DO 口位号　范围：DO00-DO47 Bit_Output_Enable：该 DO 口位使能 取值： 0：禁止控制 1：允许控制 StationNumber: 站号设置，默认不填为 0

示例：

1. 设置站号0输出口D000 在连接超时时可被控制；
short rtn;
rtn = MCF_Set_Trigger_Output_Bit_Net (0，1，0);

1.4 控制卡链接监测

函数原型	short **MCF_Get_Link_State_Net** (unsigned short StationNumber = 0);
使用说明	链接监测函数
参数说明	StationNumber: 站号设置，默认不填为 0

示例：

1. 监视站号 0 运动控制卡链接是否正常
unsigned short rtn;
rtn = MCF_Get_Link_State_Net (0); // 线程/定时器一直监视函数返回值状态。

第 2 章 通用数字 I/O

运动控制卡提供带光电隔离的通用数字量输入和通用数字量输出接口。

用户使用运动控制卡上的数字I/O口用于检测开关信号、传感器信号等输入信号，或者控制继电器、电磁阀等输出设备的信号。

各款产品对应的通用数字IO数量

型号	数字输入 IO	数字输出 IO
NIO0808R	8	8
NIO1616R	16	16
NIO2416	24	16
NIO3232	32	32
NIO4832	48	32
NIO4000	40	
NIO0040		40
NMC1100R/1200R	8	8
NMC1400R	16	16
NMC3400/3401	16	16
NMC1600R/1800	18	16
NMC5400R/5600R/5800	18	16
NMC5120	12	16
NMC5160		16
NAD0804	12	12

表3-1

2.1 通用 IO 全部输出

在使用函数**MCF_Set_Output_Net**对全部输出口进行置位或者调用**MCF_Get_Output_Net**读取全部输出口电平时应该使用十六进制数进行赋值（尽量避免使用十进制数，特别是在不支持无符号变量的开发环境）。在对 DO 电平进行控制与读取时，使用十六进制数赋值远比使用十进制数赋值更加直观、方便。

函数原型	short **MCF_Set_Output_Net** (unsigned long All_Output_Logic, unsigned short StationNumber = 0);
使用说明	设置通用数字 I/O 的输出信号
参数说明	All_Output_Logic：设置输出 DO00-DO47 电平

	StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Output_Net** (unsigned long *All_Output_Logic, unsigned short StationNumber = 0);
使用说明	读取通用数字 I/O 的输出信号
参数说明	*All_Output_Logic：读取输出 DO00-DO47 电平状态 StationNumber：站号设置，默认不填为 0

示例：

1. 设置站号 0，D000-D031 全部输出高电平(触点断开，硬件灯灭)

```
short rtn;
rtn = MCF_Set_Output_Net (0xffff, 0)          //0xffff，表示输出 DO00 - DO31 为高电平；//站号 0;
```

2.2 通用 IO 按位输出

函数原型	short **MCF_Set_Output_Bit_Net** (unsigned short Bit_Output_Number, unsigned short Bit_Output_Logic, unsigned short StationNumber = 0);
使用说明	按位设置通用数字 I/O 的输出信号
参数说明	Bit_Output_Number：选择 DO 口位号 范围：DO00-DO47 Bit_Output_Logic：设置 DO 口电平 取值： 0 //低电平(触点闭合,硬件灯亮) 1 //高电平(触点断开,硬件灯灭) StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Output_Bit_Net** (unsigned short Bit_Output_Number, unsigned short *Bit_Output_Logic, unsigned short StationNumber = 0);
使用说明	按位读取通用数字 I/O 的输出信号
参数说明	Bit_Output_Number：选择 DO 口位号 范围：DO00-DO47 *Bit_Output_Logic：读取 DO 口电平状态 取值： 0 //低电平(触点闭合,硬件灯亮) 1 //高电平(触点断开,硬件灯灭) StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Set_Output_Block_Bit_Net** (unsigned short Bit_Output_Number, unsigned short Bit_Output_Logic, unsigned short StationNumber = 0);
使用说明	通用 IO 按位输出阻塞函数,需要等待上个输出完成才退出函数
参数说明	Bit_Output_Number：选择 DO 口位号　范围：DO00-DO47 Bit_Output_Logic：读取 DO 口电平状态 取值： 0　//低电平(触点闭合,硬件灯亮) 1　//高电平(触点断开,硬件灯灭) StationNumber：站号设置，默认不填为 0

示例：

1. 设置站号 0 D000 输出低电平(触点闭合, 硬件灯亮)

```
short rtn;
rtn = MCF_Set_Output_Bit_Net(0,  0,  0);
```

2.3 通用 IO 输出复用

2.3.1 按位输出保持时间

运动控制卡提供**MCF_Set_Output_Time_Bit_Net** 函数，将对应输出位输出保持一段时间。

函数原型	short **MCF_Set_Output_Time_Bit_Net** (unsigned short Bit_Output_Number, unsigned short Bit_Output_Logic, unsigned short Output_Time_1MS, unsigned short StationNumber = 0);
使用说明	通用 IO 输出复用：按位输出保持时间函数 **注意：新的输出指令会将此指令打断（覆盖）**
参数说明	Bit_Output_Number：选择 DO 口位号　范围：DO00-DO47 Bit_Output_Logic：DO 口电平状态 取值： 0　//低电平(触点闭合,硬件灯亮) 1　//高电平(触点断开,硬件灯灭) Output_Time_1MS：电平保持时间　单位：ms 取值：0

	StationNumber：站号设置，默认不填为 0

示例：

1. 设置D000 输出低电平，保持100ms；

```
short rtn;
rtn = MCF_Set_Output_Time_Bit_Net(0, 0, 100, 0);
```

函数原型	short **MCF_Set_Output_Time_All_Net** (unsigned short *Bit_Output_Logic, unsigned short *Output_Time_1MS, unsigned short StationNumber = 0);
使用说明	通用 IO 输出复用：全部按位输出保持时间函数 **注意：新的输出指令会将此指令打断（覆盖）**
参数说明	*Bit_Output_Logic：通过数组设定 DO00-DO40 的输出状态，数值最小必须为 40 个

Bit_Output_Logic	功能
Bit_Output_Logic[0]	DO00 的输出状态
Bit_Output_Logic[1]	DO01 的输出状态
Bit_Output_Logic[2]	DO02 的输出状态
Bit_Output_Logic[3]	DO03 的输出状态
……	……
Bit_Output_Logic[37]	DO37 的输出状态
Bit_Output_Logic[38]	DO38 的输出状态
Bit_Output_Logic[39]	DO39 的输出状态
Bit_Output_Logic[40]	DO40 的输出状态

取值：

0 //低电平(触点闭合,硬件灯亮)

1 //高电平(触点断开,硬件灯灭)

*Output_Time_1MS：通过数组设定 DO00-DO40 的输出电平保持时间，数值最小必须为 40 个

Output_Time_1MS	功能
Output_Time_1MS[0]	DO00 的输出时间
Output_Time_1MS[1]	DO01 的输出时间
Output_Time_1MS[2]	DO02 的输出时间

	Output_Time_1MS[3] DO03 的输出时间 …… …… Output_Time_1MS[37] DO37 的输出时间 Output_Time_1MS[38] DO38 的输出时间 Output_Time_1MS[39] DO39 的输出时间 Output_Time_1MS[40] DO40 的输出时间 单位：ms 取值：0 - 65535 StationNumber：站号设置，默认不填为 0

2.3.2 按 XY 插补运动合成位置（编码器）输出保持时间

运动控制卡提供**MCF_Set_Compare_Output_Bit_Net** 函数，设定XY插补运动，编码器值到达指定触发位置， 对应的输出位输出**低电平(触点闭合,硬件灯亮)。**

函数原型	short **MCF_Set_Compare_Output_Bit_Net** (unsigned short Compare_Output_Number, unsigned short Compare_Output_1MS, unsigned long Compare_dDist, unsigned short StationNumber = 0);
使用说明	通用 IO 输出复用：按 XY 插补运动合成位置（编码器）输出保持时间函数
参数说明	Compare_Output_Number：选择 DO 口位号　范围：DO00-DO47 Compare_Output_1MS：电平保持时间　单位：ms 取值：0 - 65535 Compare_dDist：触发位置 StationNumber：站号设置，默认不填为 0

示例：

```
1.设置D000 XY插补编码器位置为10000，输出1000ms低电平(触点闭合,硬件灯亮)；
short rtn;
rtn = MCF_Set_Compare_Output_Bit_Net(0,1000, 10000, 0);
```

2.4 通用 IO 全部输入

使用函数**MCF_Get_Input_Net**读取全部输入点状态；

函数原型	short **MCF_Get_Input_Net** (unsigned long *All_Input_Logic1,

	unsigned long *All_Input_Logic2, unsigned short StationNumber = 0);
使用说明	读取通用数字 I/O 的输入信号
参数说明	*All_Input_Logic1：返回输出 IO 口 DI00 - DI31 的电平状态 *All_Input_Logic2：返回输出 IO 口 DI32 - DI47 的电平状态 StationNumber: 站号设置，默认不填为 0

示例：

```
2. 读取站号 0 所有的输入信号
short rtn;
unsigned long Input_Logic1；
unsigned long Input_Logic2;
Rtn = MCF_Get_Input_Net(&Input_Logic1, &Input_Logic2, 0);
// Input_Logic1 读取到 DI00 - DI31 输入信号
// Input_Logic2 读取到 DI32 - DI47 输入信号
// 站号为 0
```

2.5 通用 IO 按位输入

使用**MCF_Get_Input_Bit_Net**按位读取输入点状态；

函数原型	short **MCF_Get_Input_Bit_Net** (unsigned short Bit_Input_Number unsigned short *Bit_Input_Logic, unsigned short StationNumber = 0);
使用说明	按位读取通用数字 I/O 的输入信号
参数说明	Bit_Input_Number：选择 DI 口位号　范围：DI00-DI47 Bit_Input_Logic：读取 DI 口电平状态 取值： 0　//低电平(触点闭合,硬件灯亮) 1　//高电平(触点断开,硬件灯灭) StationNumber：站号设置，默认不填为 0

示例：

```
1. 按位读站号 0，DI00 通用输入信号
short rtn;
unsigned short Input;
rtn = MCF_Get_Input_Bit_Net(0, &Input, 0);
```

注意： 如果模块是 NIO4832 模块，每个 NIO4832 模块的输入信号有 48 个，所以提供 MCF_Get_Input_Net 和 MCF_Get_Input_Bit_Net 函数进行 NIO4832 模块输入信号的读取，MCF_Get_Input_Net第一个参数读取前32个输入信号，第二个参数读取后16个输入信号。

2.6 通用IO按位输入下降沿高速捕获清除

运动控制卡提供 IO 下降沿高速捕获功能，调用 **MCF_Get_Input_Fall_Bit_Net** 读取对应 IO 输入点是否出现下降沿变化，调用 **MCF_Clear_Input_Fall_Bit_Net** 可以清除 **MCF_Get_Input_Fall_Bit_Net** 的触发状态和 **MCF_Get_Input_Fall_Count_Bit_Net** 的计数和触发的 10 个锁存位置；

函数原型	short **MCF_Clear_Input_Fall_Bit_Net** (unsigned short Bit_Input_Number, unsigned short StationNumber = 0);
使用说明	通用 IO 按位输入下降沿高速捕获清除函数
参数说明	Bit_Input_Number：设置位号； 取值： Bit_Input_0 ~ Bit_Input_3； StationNumber：　站点号；

2.7 通用IO按位输入下降沿高速捕获读取

运动控制卡提供高速 IO 下降沿触发标志。 可以检测 IO 下降沿是否触发。

函数原型	short **MCF_Get_Input_Fall_Bit_Net** (unsigned short Bit_Input_Number, unsigned short *Bit_Input_Fall, unsigned short StationNumber = 0);
使用说明	通用 IO 按位输入下降沿高速捕获读取函数
参数说明	Bit_Input_Number：设置位号； 取值： Bit_Input_0 ~ Bit_Input_3； *Bit_Input_Fall：下降沿状态； 取值： 0 //下降沿(触点闭合,硬件灯亮瞬间)检查中 1 //下降沿(触点闭合,硬件灯亮瞬间)触发 StationNumber： 站点号；

示例：

1. 获取下降沿触发信号， 清除下降沿触发

```
1  //外部创建线程
2  线程循环函数（）
3  {
4      short rtn = 0;
5      unsigned short Bit_Input_Fall = 99;
6      //清除DI00下降沿触发状态
7      rtn = MCF_Clear_Input_Fall_Bit_Net(Bit_Input_0, 0);
8      while(true)
9      {
10         Sleep(5);//建议循环内加个1MS以上延时
11         //线程内部循环检测DI00是否有下降沿信号触发
12         rtn = MCF_Get_Input_Fall_Bit_Net(Bit_Input_0,&Bit_Input_Fall, 0);
13         //有触发， 等待100ms后清除信号触发状态
14         if(Bit_Input_Fall == Bit_Input_Fall_Trigger)
15         {
16             Sleep(100);
17             rtn = MCF_Clear_Input_Fall_Bit_Net(Bit_Input_0, 0);
18         }
19     }
20 }
```

运动控制卡提供高速 IO 下降沿触发记录和 X 轴编码器值锁存功能。可以获取高速 IO 下降沿触发次数和最近 10 位触发后 X 轴的编码器数值。

2.8 通用IO按位输入下降沿高速计数读取

函数原型	short **MCF_Get_Input_Fall_Count_Bit_Net** (unsigned short Bit_Input_Number, unsigned long *Input_Count_Fall,

	unsigned long *Lock_Data_Buffer, unsigned short StationNumber = 0);
使用说明	通用 IO 按位输入下降沿高速计数读取函数
参数说明	Bit_Input_Number：设置位号； 取值： Bit_Input_0 ~ Bit_Input_1； *Input_Count_Fall：返回下降沿触发次数； 取值： 0 ~ (2^32-1)； *Lock_Data_Buffer：返回下降沿触发数据数组； **注意：只能返回 10 个数据的数组；** StationNumber： 站点号；

示例：

1. 清除下降沿触发，获取输入 0 的最近 10 组下降沿触发信号数据

```
//外部创建线程
线程循环函数（）
{
    short         i                      = 0;
    short         rtn                    = 0;
    unsigned long Input_Count_Fall       = 0;
    unsigned long Temp_Input_Count_Fall  = 0;    //储存当前读取的个数
    unsigned long Input_Fall_Num         = 0;    //存储增长的个数
    unsigned long Lock_Data_Buffer[10]   = {0};
    unsigned long Temp_Lock_Data[10]     = {0};  //存储当前的数据
    //清除DI00下降沿触发状态
    rtn = MCF_Clear_Input_Fall_Bit_Net(Bit_Input_0, 0);
    while(true)
    {
        Sleep(10);//建议循环内加个1MS以上延时
        //线程内部循环检测DI00是否有下降沿信号触发
        rtn = MCF_Get_Input_Fall_Count_Bit_Net(Bit_Input_0, &Input_Count_Fall, &Lock_Data_Buffer[0], 0);
        //有触发，  等待100ms后清除信号触发状态
        if(Input_Count_Fall != Temp_Input_Count_Fall)
        {
            //存储增长的个数
            Input_Fall_Num = Input_Count_Fall - Temp_Input_Count_Fall;
            if(Input_Fall_Num > 10) break; //如果增加的个数超出了10个，检测数据溢出
            //储存当前读取的个数
            Temp_Input_Count_Fall = Input_Count_Fall;
            //储存更新的数据
            for(i = 0; i < Input_Fall_Num; i++)
                Temp_Lock_Data[i] = Lock_Data_Buffer[i];
        }
    }
}
```

2.9 通用IO按位输入数据锁存保持打开

函数原型	short **MCF_Open_Input_Lock_Bit_Net** (unsigned short Lock_Mode = 0, unsigned short StationNumber = 0);
使用说明	通用 IO 按位输入数据锁存保持(最近 10 个下降沿数据) **注意：打开函数(必须在 MCF_Open_Net 前面提前调用)**
参数说明	Lock_Mode：设置高速 IO 锁存模式；

	取值： 0； StationNumber： 站点号；

功能描述：输入 0 下降沿捕获 X 轴编码器，输入 1 下降沿捕获 Y 轴编码器，对这两个通道数据增加 10 级缓存，不打开只能获取最近一次下降沿编码器数据，打开后每次保存最近 10 次捕获到的下降沿编码器数据，防止软件查询漏过产品检测个数，保证产品计数准确到位。

示例：

1. 打开高速 IO 下降沿锁存功能

```
short rtn = 0;
unsigned short CardNumber = 1;            //卡数量
unsigned short Station_Number[] = {0};    //卡站号
unsigned short Station_Type[] = {2};      //卡类型：2
//开启通用IO按位输入数据锁存保持
rtn = MCF_Open_Input_Lock_Bit_Net (0, 0);
//打开卡
rtn = MCF_Open_Net(CardNumber, &Station_Number[0], &Station_Type[0]);
//设置滤波函数
//位号：DI00;
//时间：10ms;
//站号：0;
rtn = MCF_Set_Input_Filter_Time_Bit_Net(Bit_Input_0, 10, 0);
```

2.10 通用IO按位输入滤波

函数原型	short **MCF_Set_Input_Filter_Time_Bit_Net** (unsigned short Bit_Input_Number = 0, unsigned long Filter_Time_1MS, unsigned short StationNumber = 0);
使用说明	通用 IO 按位输入滤波函数
参数说明	Bit_Input_Number：设置位号； 取值： Bit_Input_0 ~ Bit_Input_1； Filter_Time_1MS：滤波时间； **(默认 0.4MS)** 取值： 1 ~ 100MS； StationNumber： 站点号；

示例：

1. 设置输入 0 滤波

```
short rtn = 0;
unsigned short CardNumber = 1;            //卡数量
unsigned short Station_Number[] = {0};  //卡站号
unsigned short Station_Type[] = {2};     //卡类型：2
//开启通用IO按位输入数据锁存保持
rtn = MCF_Open_Input_Lock_Bit_Net (0, 0);
//打开卡
rtn = MCF_Open_Net(CardNumber, &Station_Number[0], &Station_Type[0]);
//设置滤波函数
//位号：DI00;
//时间：10ms;
//站号：0;
rtn = MCF_Set_Input_Filter_Time_Bit_Net(Bit_Input_0, 10, 0);
```

第 3 章 专用数字 I/O

运动控制卡轴控模块中每个控制轴都提供多个专用 IO 信号，包括正限位、负限位、原点信号、伺服报警、使能信号和报警复位信号，并提供专门的函数读写专用 IO 信号，主要用于读写专用 IO 的电平。

3.1 伺服使能设置

用户可以通过 MCF_Set_Servo_Enable_Net 函数设置轴使能与关闭使能，其中将参数 Enable 设为 0 表示使能通道输出低电平，反之，输出高电平。

用户可通过 MCF_Get_Servo_Enable_Net 函数读取当前轴是否是使能状态。

函数原型	short **MCF_Set_Servo_Enable_Net** (unsigned short Axis, unsigned short Servo_Logic, unsigned short StationNumber = 0);
使用说明	设置伺服使能
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Servo_Logic：伺服使能设置 取值： 0 //低电平(触点闭合) 1 //高电平(触点断开) StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Servo_Enable_Net** (unsigned short Axis, unsigned short *Servo_Logic, unsigned short StationNumber = 0);
使用说明	读取伺服使能状态
参数说明	Axis： 指定轴号，取值：0-Axis_MAX *Servo_Logic：返回伺服使能状态 返回值定义： 0 //低电平(触点闭合) 1 //高电平(触点断开) StationNumber：站号设置，默认不填为 0

示例：

1. 设置伺服使能

```
short rtn;
rtn = MCF_Set_Servo_Enable_Net (Axis_1, 0); //控制轴为轴 1，0 表示使能端口输出低电平
```

3.2 伺服报警复位设置

用户如果需要使用报警复位功能，轴控模块提供专用的伺服报警复位输出口，但此信号无法清除所有驱动报警情况。用户可以通过 MCF_Set_Alarm_Reset_Net 函数设置报警复位的电平，也可以通过 MCF_Get_Alarm_Reset_Net 函数获取当前伺服报警复位输出口的输出电平。

函数原型	short **MCF_Set_Servo_Alarm_Reset_Net** (unsigned short Axis, unsigned short Alarm_Logic, unsigned short StationNumber = 0);
使用说明	设置伺服报警复位
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Alarm_Logic：设置复位电平 取值： 0 //低电平(触点闭合) 1 //高电平(触点断开) StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Servo_Alarm_Reset_Net** (unsigned short Axis, unsigned short *Alarm_Logic, unsigned short StationNumber = 0);
使用说明	读取伺服报警复位状态
参数说明	Axis： 指定轴号，取值：0-Axis_MAX *Alarm_Logic：返回复位电平 返回值定义： 0 //低电平(触点闭合) 1 //高电平(触点断开) StationNumber：站号设置，默认不填为 0

示例：

1. 设置报警复位

```
short rtn;
rtn = MCF_Set_Servo_Alarm_Reset_Net (Axis_1, 0); //控制轴为轴 1，0 表示报警复位端口输出低电平
```

3.3 伺服报警输入获取

用户如果需要获取伺服报警信号，需要将驱动器的报警输出信号线与轴专用的伺服报警接口相连。通过 **MCF_Get_Servo_Alarm_Net** 函数获取伺服报警的实际电平，再将轴配置伺服报警有效，当伺服报警触发时，轴控模块会停止该轴运动，并将该轴使能信号关闭，当清除伺服报警信号后，需要重新上使能。

函数原型	short **MCF_Get_Servo_Alarm_Net** (unsigned short Axis, unsigned short *Servo_Alarm_State, unsigned short StationNumber = 0);
使用说明	读取伺服报警电平
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Servo_Alarm_State：返回伺服报警信号电平 返回值定义： 0 //低电平(触点闭合) 1 //高电平(触点断开) StationNumber：站号设置，默认不填为 0

示例：

1. 读取伺服报警电平

```
short rtn;
rtn = MCF_Get_Servo_Alarm_Net(Axis_1, &Alm);      //控制轴为轴 1，Alm 返回伺服报警端口电平
```

3.4 伺服定位完成输入获取

用户可以通过 **MCF_Get_Servo_INP_Net** 函数获取伺服定位完成信号。

函数原型	short **MCF_Get_Servo_INP_Net** (unsigned short Axis, unsigned short *Servo_INP_State, unsigned short StationNumber = 0);
使用说明	读取伺服定位完成输入
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Servo_INP_State：返回伺服定位完成输入 返回值定义： 0 //低电平(触点闭合) 1 //高电平(触点断开) StationNumber：站号设置，默认不填为 0

示例：

1. 读取伺服定位完成信号

```
short rtn;
rtn = MCF_Get_Servo_INP_Net(0,&Servo_INP_State,0);
//控制轴为轴 1，INP 返回伺服定位完成端口电平
```

3.5 编码器 Z 相输入获取

用户可以通过 MCF_Get_Z_Net 函数获取 Z 相输出信号。

函数原型	short **MCF_Get_Z_Net**

	(unsigned short Axis, unsigned short *Z_State, unsigned short StationNumber = 0);
使用说明	读取编码器 Z 相输入
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Z_State：返回编码器 Z 相输入 0 //低电平(触点闭合) 1 //高电平(触点断开) StationNumber：站号设置，默认不填为 0

示例：

1. 读取 Z 信号

```
short rtn;
rtn = MCF_Get_Z_Net(0,&Z_State,0);          //控制轴为轴 1，Z+ Z-返回端口信号
```

3.6 原点输入获取

用户可以通过 **MCF_Get_Home_Net** 函数获取原点信号的实际电平，原点信号主要用户初始化回零。建议使用常开型开关作为原点信号，即低电平有效。

函数原型	short **MCF_Get_Home_Net** (unsigned short Axis, unsigned short *Home_State, unsigned short StationNumber = 0);
使用说明	读取正限位电平
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Home_State：返回原点信号电平 0 //低电平(触点闭合,硬件灯亮) 1 //高电平(触点断开,硬件灯灭) StationNumber：站号设置，默认不填为 0

示例：

1. 读取原点开关电平

```
short rtn;
rtn = MCF_Get_Home_Net(Axis_1, &Home);       //控制轴为轴 1，Home 返回原点信号端口电平
```

3.7 正限位输入获取函数

用户可以通过 **MCF_Get_Positive_Limit_Net** 函数读取正限位的实际电平，再将轴配置正限位有效，如果轴运动超出限位范围，限位触发，运动停止并禁止继续往限位方向运动，用户可以往反方向离开限位回到正常运动范围内。

函数原型	short **MCF_Get_Positive_Limit_Net** (unsigned short Axis,

	unsigned short *Positive_Limit_State, unsigned short StationNumber = 0);
使用说明	读取正限位电平
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Positive _Limit_State：返回正限位信号电平 返回值定义： 0 //低电平(触点闭合,硬件灯亮) 1 //高电平(触点断开,硬件灯灭) StationNumber：站号设置，默认不填为 0

示例：

1. 读取正限位电平

```
short rtn;
rtn= MCF_Get_Positive_Limit_Net(Axis_1, & Positive_Limit);
//控制轴为轴 1，Positive_Limit 返回正限位端口电平
```

3.8 负限位输入获取函数

用户可以通过 **MCF_Get_Negative_Limit_Net** 函数读取负限位的实际电平，再将轴配置负限位有效，如果轴运动超出限位范围，限位触发，运动停止并禁止继续往限位方向运动，用户可以往反方向离开限位回到正常运动范围内。

函数原型	short **MCF_Get_Negative_Limit_Net** (unsigned short Axis, unsigned short *Negative_Limit_State, unsigned short StationNumber = 0);
使用说明	读取负限位电平
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Negative_Limit_State：返回负限位信号电平 返回值定义： 0 //低电平(触点闭合,硬件灯亮) 1 //高电平(触点断开,硬件灯灭) StationNumber：站号设置，默认不填为 0

示例：

1. 读取负限位电平

```
short rtn;
rtn = MCF_Get_Negative_Limit_Net(Axis_1, & Negative _Limit);
//控制轴为轴 1，Negative _Limit 返回负限位端口电平
```

第4章 设置轴参数

4.1 脉冲通道输出设置

控制卡提供 MCF_Set_Pulse_Mode_Net 函数来设置 Mode 寄存器，从而实现 6 种不同脉冲输出模式。

<table>
<tr><td>函数原型</td><td>short MCF_Set_Pulse_Mode_Net
(unsigned short Axis,
unsigned long Pulse_ModeS,
unsigned short StationNumber = 0);</td></tr>
<tr><td>使用说明</td><td>设置轴脉冲输出模式</td></tr>
<tr><td>参数说明</td><td>Axis：指定轴号，取值：0-Axis_MAX
Pulse_Mode： 设置脉冲输出模式
取值：如下表
<table>
<tr><th rowspan="2">模式</th><th colspan="2">正方向脉冲</th><th colspan="2">负方向脉冲</th></tr>
<tr><th>Pulse 脚输出</th><th>Dir 脚输出</th><th>Pulse 脚输出</th><th>Dir 脚输出</th></tr>
<tr><td>0 脉冲/方向(PULSE/DIR)</td><td>脉冲</td><td>高电平</td><td>脉冲</td><td>低电平</td></tr>
<tr><td>1 脉冲/方向(PULSE/DIR)</td><td>脉冲</td><td>低电平</td><td>脉冲</td><td>高电平</td></tr>
<tr><td>2 双脉冲(CW/CCW)</td><td>脉冲</td><td>低电平</td><td>低电平</td><td>脉冲</td></tr>
<tr><td>3 双脉冲(CW/CCW)</td><td>底电平</td><td>脉冲</td><td>脉冲</td><td>低电平</td></tr>
<tr><td>4 AB 相位</td><td>A 超前 B90 度</td><td>A 超前 B90 度</td><td>A 滞后 B90 度</td><td>A 滞后 B90 度</td></tr>
<tr><td>5 AB 相位</td><td>A 滞后 B90 度</td><td>A 滞后 B90 度</td><td>A 超前 B90 度</td><td>A 超前 B90 度</td></tr>
</table>
StationNumber：站号设置，默认不填为 0</td></tr>
</table>

<table>
<tr><td>函数原型</td><td>short MCF_Get_Pulse_Mode_Net
(unsigned short Axis,
unsigned long *Pulse_Mode,
unsigned short StationNumber = 0);</td></tr>
<tr><td>使用说明</td><td>读取轴脉冲输出模式</td></tr>
<tr><td>参数说明</td><td>Axis：指定轴号，取值：0-Axis_MAX
*Pulse_Mode： 读取设置的脉冲输出模式</td></tr>
</table>

	StationNumber：站号设置，默认不填为 0

示例：

1. 设置脉冲模式

```
short rtn;
rtn = MCF_Set_Pulse_Mode_Net (Axis_1,0); //设置 Axis_1 轴为脉冲/方向输出模式。
```

4.2 位置设置

运动控制卡上每个轴都拥有对应指令位置计数器

用户可以通过MCF_Set_Position_Net指令来设置指令位置，通过MCF_Get_Position_Net来读取指令位置，指令位置计数器是一个32位正负计数器，对控制卡输出的指令脉冲进行计数。当输出一个正向脉冲后，计数器加1；当输出一个负向脉冲后，计数器减1。

函数原型	short **MCF_Set_Position_Net** (unsigned short Axis, long Position, unsigned short StationNumber = 0);
使用说明	设置轴的当前规划位置
参数说明	Axis：　　指定轴号，取值：0-Axis_MAX Position：　设置轴的当前位置，单位：脉冲 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Position_Net** (unsigned short Axis, long *Position, unsigned short StationNumber = 0);
使用说明	读取轴的当前规划位置
参数说明	Axis：　　指定轴号，取值：0-Axis_MAX *Position：　读取轴的当前位置，单位：脉冲 StationNumber：站号设置，默认不填为 0

示例：

1. 设置轴 1 脉冲位置

```
short rtn;
rtn = MCF_Set_Position_Net (Axis_1, 0);
```

2. 读取轴 1 脉冲位置

```
short rtn;
long CmdPos = 0;
rtn = MCF_Get_Position_Net (Axis_1,& CmdPos，0);
```

4.3 编码器设置

运动控制卡上每个轴都拥有对应编码器反馈位置计数器

编码器计数器也是一个32位正负计数器，对ABZ信号进行解码，运动控制根据AB相的相位来分辨正转和反转，具体如图：

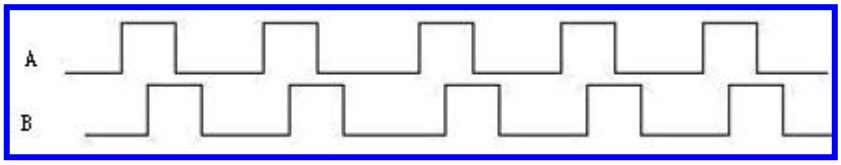


图2-1 正转：A超前B 90度

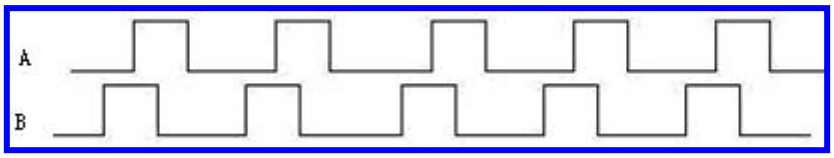


图2-2 反转：B超前A 90度

//在设置完编码器工作模式后，通过MCF_Get_Encoder_Net读取编码器反馈脉冲计数值

4.3.1 设置与读取编码器反馈脉冲计数值

用户可以通过MCF_Set_Encoder_Net指令来设置编码器计数器的值，通过MCF_Get_Encoder_Net来读取编码器计数器的值，编码器计数器也是一个32位正负计数器，对控制卡接受的编码器反馈脉冲进行计数。当接收一个正向脉冲后，计数器加1；当接收一个负向脉冲后，计数器减1。

函数原型	short **MCF_Set_Encoder_Net** (unsigned short Axis, long Encoder, unsigned short StationNumber = 0);
使用说明	设置轴的当前编码器位置
参数说明	Axis：指定轴号，取值：0-Axis_MAX Encoder：设置轴的编码器位置，单位：脉冲 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Encoder_Net** (unsigned short Axis, long *Encoder, unsigned short StationNumber = 0);
使用说明	读取轴的当前编码器位置
参数说明	Axis：指定轴号，取值：0-Axis_MAX *Encoder：读取轴的编码器位置，单位：脉冲 StationNumber：站号设置，默认不填为 0

示例：

1. 设置编码器值

```
short rtn;
rtn = MCF_Set_Encoder_Net (Axis_1, 0);
```

2. 读取编码器值

```
short rtn;
long EncPos = 0;
rtn = MCF_Get_Encoder_Net (Axis_1, & EncPos);
```

4.3.2 通过Z相清除AB编码器值

用户可以通过**MCF_Z_Clear_Encoder_Net**指令来设置当编码器通过Z相后编码器反馈脉冲计数器数值清零。

函数原型	short **MCF_Z_Clear_Encoder_Net** (unsigned short Axis, unsigned short Enable, unsigned short StationNumber = 0);
使用说明	通过 Z 相清除 AB 编码器值
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Enable：使能 StationNumber：站号设置，默认不填为 0

4.3.3 通过Z相后固定距离输出IO

用户可以通过 **MCF_Z_Output_Bit_Net** 指令来设置当编码器通过 Z 相后开始 AB 编码器到达指定位置触发指定输出。

函数原型	short **MCF_Z_Output_Bit_Net** (unsigned short Axis, unsigned short Number, unsigned long dDist, unsigned short Time_1MS, unsigned short StationNumber = 0);
使用说明	通过 Z 相后固定距离输出 IO
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Number：选择 DO 口位号 范围：DO00-DO47(目前固定为 DO0) dDist：设置触发位置 Time_1MS：设置触发维持时间(目前固定为不关闭) StationNumber：站号设置，默认不填为 0

4.4 速度获取

用户可以通过 **MCF_Get_Vel_Net** 指令来获取当前速度

函数原型	short **MCF_Get_Vel_Net** (unsigned short Axis, double *Command_Vel,

	double *Encode_Vel, unsigned short StationNumber = 0);
使用说明	读取实时命令速度和实际编码器速度
参数说明	Axis：指定轴号，取值：0-Axis_MAX *Command_Vel：读取实时命令速度，单位：脉冲/s *Encode_Vel：读取实时编码器速度，单位：脉冲/s StationNumber：站号设置，默认不填为 0

示例：

1. 读取当前速度

```
short rtn;
double CmdVel    = 0.0;
double EncVel    = 0.0;
rtn = MCF_Get_Vel_Net (Axis_1, & CmdVel, & EncVel,0);
//读取 Axis_1 轴的指令速度和编码器反馈速度分别存放在 CmdVel 和 EncVel 中
```

第 5 章 轴硬件运动触发停止

5.1 设置通用IO作为急停与停止

5.1.1 通用 IO 输入复用：做为紧急停止

运动控制卡可以通过函数 MCF_Set_EMG_Bit_Net 设置指定的 IO 作为触发轴停止功能；

函数原型	short **MCF_Set_EMG_Bit_Net** (unsigned short EMG_Input_Number, unsigned short EMG_Mode, unsigned short StationNumber = 0);
使用说明	通用 IO 输入复用：做为紧急停止函数
参数说明	EMG_Input_Number： 选择指定输入位 EMG_Mode： 设置触发模式 取值： 0：EMG_Trigger_Close 不使用紧急停止功能 1：EMG_Trigger_Low_IMD 低电平(触点闭合,硬件灯亮)触发紧急停止 2：EMG_Trigger_Low_DEC 低电平(触点闭合,硬件灯亮)触发减速停止 3：EMG_Trigger_High_IMD 高电平(触点断开,硬件灯灭)触发紧急停止 4：EMG_Trigger_High_DEC 高电平(触点断开,硬件灯灭)触发减速停止 StationNumber：站号设置，默认不填为 0

运动控制卡提供了 IO 作为触发恢复 DO 输出设置函数，IO 触发后，客户可以根据目标需求， 将输出点设置好对应的状态；

函数原型	short **MCF_Set_EMG_Output_Net** (unsigned short EMG_Input_Number, unsigned short EMG_Mode, unsigned long EMG_Output, unsigned short StationNumber = 0);
使用说明	通用 IO 触发恢复 DO 输出状态函数
参数说明	EMG_Input_Number：设置 DI 位号 取值：DI00 - DI47 EMG_Mode：设置触发模式(**对应电平触发， 不停止轴**) 取值： 0：Trigger_Close 关闭触发 1：Trigger_Low_IMD 低电平(触点闭合,硬件灯亮)触发

	2：Trigger_Low_DEC　低电平(触点闭合,硬件灯亮)触发 3：Trigger_High_IMD　高电平(触点断开,硬件灯灭)触发 4：Trigger_High_DEC　高电平(触点断开,硬件灯灭)触发 EMG_Output：设置 DO 状态 取值：　(32 位，从低位到高位依次对应 DO00 到 DO47) 0：低电平 1：高电平 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Set_EMG_Output_Enable_Net** (unsigned short Bit_Output_Number, unsigned short Bit_Output_Enable, unsigned short StationNumber = 0);
使用说明	通用 IO 触发恢复 DO 输出状态使能
参数说明	Bit_Output_Number：选择 DO 口位号　范围：DO00-DO47 Bit_Output_Enable：该 DO 口位使能 取值： 0：禁止控制 1：允许控制 StationNumber：站号设置，默认不填为 0

示例：

1. 设置 DI00 为硬件低电平紧急触发轴停止函数
   short rtn = 0;
   rtn = MCF_Set_EMG_Bit_Net(0, EMG_Trigger_Low_IMD, 0);
2. DI00 作为 IO 触发输出全部置为高电平。

```
short rtn = 0;
//设置DI00作为IO低电平触发急停输入点，
//触发急停后，  DO00-DO15置为高电平状态；
rtn = MCF_Set_EMG_Output_Net(Bit_Input_0, EMG_Trigger_Low_IMD,0xffff, 0);
```

5.1.2 通用 IO 输入复用：做为触发停止

运动控制卡可以通过函数 MCF_Set_Input_Trigger_Net 设置指定的输入点作为触发轴停止功能；

函数原型	short **MCF_Set_Input_Trigger_Net** (unsigned short Channel, unsigned short　Axis, unsigned short　Bit_Input_Number,

	unsigned long Trigger_Mode, unsigned short StationNumber = 0);
使用说明	通用 IO 输入复用：做为触发轴停止
参数说明	Channel： 通道号； 取值：0 ~ 7； Axis： 指定停止的轴； Bit_Input_Number： 指定触发的输入点（DI）； Trigger_Mode：设置触发模式 取值： 0：Trigger_Close 关闭触发 1：Trigger_Low_IMD 低电平(触点闭合,硬件灯亮)触发紧急停止 2：Trigger_Low_DEC 低电平(触点闭合,硬件灯亮)触发减速停止 3：Trigger_High_IMD 高电平(触点断开,硬件灯灭)触发紧急停止 4：Trigger_High_DEC 高电平(触点断开,硬件灯灭)触发减速停止 StationNumber: 站号设置，默认不填为 0

函数原型	short **MCF_Get_Input_Trigger_Net** (unsigned short Channel, unsigned short *Axis, unsigned short *Bit_Input_Number, unsigned long *Trigger_Mode, unsigned short StationNumber = 0);
使用说明	读取通用 IO 输入复用：做为触发停止设置
参数说明	Channel： 通道号； 取值：0 ~ 7； *Axis： 指定停止的轴； *Bit_Input_Number： 指定触发的输入点（DI）； *Trigger_Mode：设置触发模式 取值： 0：Trigger_Close 关闭触发 1：Trigger_Low_IMD 低电平(触点闭合,硬件灯亮)触发紧急停止 2：Trigger_Low_DEC 低电平(触点闭合,硬件灯亮)触发减速停止 3：Trigger_High_IMD 高电平(触点断开,硬件灯灭)触发紧急停止 4：Trigger_High_DEC 高电平(触点断开,硬件灯灭)触发减速停止 StationNumber: 站号设置，默认不填为 0

示例：

1. 设置通道 0， DI00 低电平触发轴 1 紧急停止

```
short rtn;
rtn = MCF_Set_Input_Trigger_Net(0, 0, 0, Trigger_Low_IMD, 0);
```

```
// 通道：0；
// 指定停止轴：轴 1；
// 指定输入点：DI00；
// 触发停止模式：  低电平触发紧急停止；
```

5.2 软件限位触发运动停止

运动控制卡可以通过函数 MCF_Set_Soft_Limit_Net 设置对应的轴的正负软件限位，当脉冲数超出设定正负脉冲边界，将无法再往对应方向运动，只能往相反方向运动；

函数原型	short **MCF_Set_Soft_Limit_Net** (unsigned short Axis, long Positive_Position, long Negative_Position, unsigned short StationNumber = 0);
使用说明	设置轴软件限位位置
参数说明	Axis：指定轴号，取值：0-Axis_MAX Positive_Position： 设置软件正限位位置，单位：脉冲 Negative_Position：设置软件负限位位置，单位：脉冲 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Soft_Limit_Net** (unsigned short Axis, long *Positive_Position, long *Negative_Position, unsigned short StationNumber = 0);
使用说明	读取轴软件限位位置
参数说明	Axis：指定轴号，取值：0-Axis_MAX *Positive_Position： 读取软件正限位位置，单位：脉冲 *Negative_Position：读取软件负限位位置，单位：脉冲 StationNumber：站号设置，默认不填为 0

示例：

1. 设置正负软件限位

```
short rtn;
rtn = MCF_Set_Soft_Limit_Net (Axis_1, 200000, -1000000,0);
  //设置 Axis_1 轴软件正限位为 200000 脉冲,负限位为-1000000 脉冲。
```

5.3 软件限位触发运动停止开关

运动控制卡可以通过函数 MCF_Set_Soft_Limit_Enable_Net 设置对应的轴的正负软件限位是否启用，当脉冲数超出设定正负脉冲边界，将无法再往对应方向运动，只能往相反方向运动；

函数原型	short **MCF_Set_Soft_Limit_Enable_Net** (unsigned short Axis, unsigned long Soft_Limit_Enable, unsigned short StationNumber = 0);
使用说明	设置软件限位触发运动开启/关闭函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Soft_Limit_Enable：设置限位开启/关闭 取值： Soft_Limit_Close 0 //软件限位关闭 Soft_Limit_Open 1 //软件限位打开 StationNumber: 站号设置，默认不填为 0

函数原型	short **MCF_Get_Soft_Limit_Enable_Net** (unsigned short Axis, unsigned long *Soft_Limit_Enable, unsigned short StationNumber = 0);
使用说明	读取软件限位触发运动开启/关闭函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Soft_Limit_Enable：读取限位开启/关闭 返回值定义： Soft_Limit_Close 0 //软件限位关闭 Soft_Limit_Open 1 //软件限位打开 StationNumber: 站号设置，默认不填为 0

示例：

1. 启动正负软件限位

```
short rtn;
rtn = MCF_Set_Soft_Limit_Enable_Net (Axis_1, Soft_Limit_Open, 0);
  //开启 Axis_1 轴软件限位使能。
```

5.4 设置伺服报警触发

客户通过调用 **MCF_Set_Alarm_Trigger_Net** 进行原点开关信号触发；

函数原型	Short **MCF_Set_Alarm_Trigger_Net** (unsigned short Axis,

	unsigned long Trigger_Mode, unsigned short StationNumber = 0);
使用说明	设置伺服报警触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Trigger_Mode：设置触发模式 取值： 0：Trigger_Close 关闭触发 1：Trigger_Low_IMD 低电平(触点闭合)触发紧急停止 2：Trigger_Low_DEC 低电平(触点闭合)触发减速停止 3：Trigger_High_IMD 高电平(触点断开)触发紧急停止 4：Trigger_High_DEC 高电平(触点断开)触发减速停止 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Alarm_Trigger_Net** (unsigned short Axis, unsigned long *Trigger_Mode, unsigned short StationNumber = 0);
使用说明	读取伺服报警触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Trigger_Mode：读取触发模式 StationNumber：站号设置，默认不填为 0

示例：

1. 设置轴 0 伺服报警低电平触发紧急停止

```
short rtn;
rtn = MCF_Set_Alarm_Trigger_Net                    (0,Trigger_Low_IMD,0);
```

5.5 Index触发运动停止

客户通过调用 **MCF_Set_Index_Trigger_Net** 进行原点开关信号触发；

函数原型	Short **MCF_Set_Index_Trigger_Net** (unsigned short Axis, unsigned long Trigger_Mode, unsigned short StationNumber = 0);
使用说明	设置 Index 触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Trigger_Mode：设置触发模式 取值：

	0：Trigger_Close 关闭触发 1：Trigger_Low_IMD 低电平(触点闭合)触发紧急停止 2：Trigger_Low_DEC 低电平(触点闭合)触发减速停止 3：Trigger_High_IMD 高电平(触点断开)触发紧急停止 4：Trigger_High_DEC 高电平(触点断开)触发减速停止 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Index_Trigger_Net** (unsigned short Axis, unsigned long *Trigger_Mode, unsigned short StationNumber = 0);
使用说明	读取 Index 触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Trigger_Mode：读取触发模式 StationNumber：站号设置，默认不填为 0

示例：

1. 设置轴 0 伺服报警低电平触发紧急停止

```
short rtn;
rtn = MCF_Set_Index_Trigger_Net                    (0,Trigger_Low_IMD,0);
```

5.6 原点触发运动停止

客户通过调用 **MCF_Set_Home_Trigger_Net** 进行原点开关信号触发；

函数原型	Short **MCF_Set_Home_Trigger_Net** (unsigned short Axis, unsigned long Trigger_Mode, unsigned short StationNumber = 0);
使用说明	设置原点触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Trigger_Mode：设置触发模式 取值： 0：Trigger_Close 关闭触发 1：Trigger_Low_IMD 低电平(触点闭合,硬件灯亮)触发紧急停止 2：Trigger_Low_DEC 低电平(触点闭合,硬件灯亮)触发减速停止 3：Trigger_High_IMD 高电平(触点断开,硬件灯灭)触发紧急停止 4：Trigger_High_DEC 高电平(触点断开,硬件灯灭)触发减速停止 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Home_Trigger_Net** (unsigned short Axis, unsigned long *Trigger_Mode, unsigned short StationNumber = 0);
使用说明	读取原点触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Trigger_Mode：读取触发模式 StationNumber：站号设置，默认不填为 0

示例：

1. 设置轴 0 原点低电平触发紧急停止

```
short rtn;
rtn = MCF_Set_Home_Trigger_Net                    (0,Trigger_Low_IMD,0);
```

5.7 正限位触发运动停止

客户通过调用 **MCF_Set_ELP_Trigger_Net** 进行正限位信号触发；

函数原型	Short **MCF_Set_ELP_Trigger_Net** (unsigned short Axis, unsigned long Trigger_Mode, unsigned short StationNumber = 0);
使用说明	设置正限位触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Trigger_Mode：设置触发模式 取值： 0：Trigger_Close 关闭触发 1：Trigger_Low_IMD 低电平(触点闭合,硬件灯亮)触发紧急停止 2：Trigger_Low_DEC 低电平(触点闭合,硬件灯亮)触发减速停止 3：Trigger_High_IMD 高电平(触点断开,硬件灯灭)触发紧急停止 4：Trigger_High_DEC 高电平(触点断开,硬件灯灭)触发减速停止 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_ELP_Trigger_Net** (unsigned short Axis, unsigned long *Trigger_Mode, unsigned short StationNumber = 0);

使用说明	读取正限位触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Trigger_Mode：读取触发模式 StationNumber：站号设置，默认不填为 0

示例：

1. 设置轴 0 正限位低电平触发紧急停止

```
short rtn;
rtn = MCF_Set_ELP_Trigger_Net                    (0,Trigger_Low_IMD,0);
```

5.8 负限位触发运动停止

客户通过调用 **MCF_Set_ELN_Trigger_Net** 进行负限位信号触发；

函数原型	Short **MCF_Set_ELN_Trigger_Net** (unsigned short Axis, unsigned long Trigger_Mode, unsigned short StationNumber = 0);
使用说明	设置负限位触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX Trigger_Mode：设置触发模式 取值： 0：Trigger_Close 关闭触发 1：Trigger_Low_IMD 低电平(触点闭合,硬件灯亮)触发紧急停止 2：Trigger_Low_DEC 低电平(触点闭合,硬件灯亮)触发减速停止 3：Trigger_High_IMD 高电平(触点断开,硬件灯灭)触发紧急停止 4：Trigger_High_DEC 高电平(触点断开,硬件灯灭)触发减速停止 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_ELN_Trigger_Net** (unsigned short Axis, unsigned long *Trigger_Mode, unsigned short StationNumber = 0);
使用说明	读取负限位触发运动停止函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Trigger_Mode：读取触发模式 StationNumber：站号设置，默认不填为 0

示例：

1. 设置轴 0 负限位低电平触发紧急停止

```
short rtn;
rtn = MCF_Set_ELN_Trigger_Net                    (0,Trigger_Low_IMD,0);
```

5.9 原点触发位置记录

控制卡在每个轴的原点上都设定有原点上下电平变化时， 记录变化位置信息， 记录信息只记录最近的一组数据；

函数原型	short **MCF_Get_Home_Rise_Position_Net** (unsigned short Axis, long *Position, unsigned short StationNumber = 0);
使用说明	捕捉原点限位上升沿位置
参数说明	Axis：设置轴号 *Position: 存储捕捉位置 StationNumber： 站点号

函数原型	short **MCF_Get_Home_Fall_Position_Net** (unsigned short Axis, long *Position, unsigned short StationNumber = 0) ;
使用说明	捕捉原点限位下降沿位置
参数说明	Axis：设置轴号 *Position: 存储捕捉位置 StationNumber： 站点号

函数原型	short **MCF_Get_Home_Rise_Encoder_Net** (unsigned short Axis, long *Encoder, unsigned short StationNumber = 0);
使用说明	捕捉原点限位上升沿编码器位置
参数说明	Axis：设置轴号 *Encoder: 存储捕捉位置 StationNumber： 站点号

函数原型	short **MCF_Get_Home_Fall_Encoder_Net** (unsigned short Axis, long *Encoder, unsigned short StationNumber = 0);

使用说明	捕捉原点限位下降沿编码器位置
参数说明	Axis：设置轴号 *Encoder：存储捕捉位置 StationNumber： 站点号

5.10 轴状态清除

当报警原因解除时，运动控制卡内部将自动清除异常标志，但报警原因寄存器需要通过专用的报警原因清除函数MCF_Clear_Axis_State_Net来清除寄存器，否则MCF_Get_Axis_State_Net会一直保存上一次报警原因，除非轴进行新的动作或者新的报警，报警原因寄存器才会刷新。

函数原型	short **MCF_Clear_Axis_State_Net** (unsigned short Axis, unsigned short StationNumber = 0);
使用说明	轴状态清除函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX StationNumber：站号设置，默认不填为 0

5.11 轴状态触发停止运动查询

在运动过程中，如果由于限位，伺服报警，外部急停等原因导致运动停止或者无法运动，可以通过MCF_Get_Axis_State_Net函数获取轴停止原因，详细原因如下表所示。

如果同时有多个原因导致轴停止，函数将返回优先级最高的原因。

函数原型	short **MCF_Get_Axis_State_Net** (unsigned short Axis, unsigned short *Reason, unsigned short StationNumber = 0);
使用说明	轴状态触发停止运动查询函数
参数说明	Axis： 指定轴号，取值：0-Axis_MAX * Reason： 返回值定义： 0 正常命令执行成功 1 正在执行 2 EMG 立即紧急停止 3 EMG 减速紧急停止

	4 ALM 立即停止 5 ALM 减速停止 6 伺服使能立即停止 7 伺服使能减速停止 8 指令编码器误差立即停止 9 指令编码器误差减速停止 10 Index 立即停止 11 Index 减速停止 12 原点立即停止 13 原点减速停止 14 正硬限位立即停止 15 正硬限位减速停止 16 负硬限位立即停止 17 负硬限位减速停止 18 正软限位立即停止 19 正软限位减速停止 20 负软限位立即停止 21 负软限位减速停止 22 命令立即停止 23 命令减速停止 24 其它原因立即停止 25 其它原因减速停止 26 未知原因立即停止 27 未知原因减速停止 28 外部 IO 减速停止 StationNumber：站号设置，默认不填为 0

在运动过程中，如果用户需要判断轴当前的运动状态，也可以通过MCF_Get_Axis_State_Net函数查询轴当前状态。

示例：

1. 读取轴 0 状态

```
short rtn;
unsigned short bFin = 0;
rtn = MCF_Get_Axis_State_Net (Axis_1, & bFin);
if(bFin == 1)          { /*当前轴在运动*/ }
else if(bFin == 0)     {/*当前轴命令执行完成，正常停止*/}
else                   {/*当前轴有异常触发事件，轴因事件停止*/}
```

第 6 章 回原点运动

6.1 回原点模式

在进行精确的运动控制之前，需要设定运动坐标系的原点。运动平台上都设有原点传感器（也称为原点开关）。寻找原点开关的位置并将该位置设为平台的坐标原点的过程即为回原点运动。

运动控制卡针对各类情况提供了数十种回零方案供客户选择，客户可以根据设备的限位安装情况，以及需求进行对应查找相应的模式， 方案如表 6-1、6-2。

回原点模式选择参考表

回原点模式选择参考表						
物理实际位置			模式选择	搜索方向	停止位置	偏移方向
机械负极限	机械中间	机械正极限				
安装负限开关			17	负方向	负限位正边	正方向
		安装正限开关	18	正方向	正极限负边	负方向
		安装原点开关	19	正方向	原点负边外侧	负方向
			20	正方向	原点负边内侧	正方向
安装原点开关			21	负方向	原点正边外侧	正方向
			22	负方向	原点正边内侧	负方向
	安装原点开关	安装正限开关	23	正方向	原点负边外侧	负方向
			24	正方向	原点负边内侧	正方向
			25	正方向	原点正边内侧	负方向
			26	正方向	原点正边外侧	正方向
安装负限开关	安装原点开关		27	负方向	原点正边外侧	正方向
			28	负方向	原点正边内侧	负方向
			29	负方向	原点负边内侧	正方向
			30	负方向	原点负边外侧	负方向

表 6-1

回原点模式选择参考表						
物理实际位置			模式选择	搜索方向	Index 位置	偏移方向
机械负极限	机械中间	机械正极限				
安装负限开关			1	负方向	负限位正边	正方向
		安装正限开关	2	正方向	正极限负边	负方向
		安装原点开关	3	正方向	原点负边外侧	负方向
			4	正方向	原点负边内侧	正方向
安装原点开关			5	负方向	原点正边外侧	正方向
			6	负方向	原点正边内侧	负方向
	安装原点开关	安装正限开关	7	正方向	原点负边外侧	负方向
			8	正方向	原点负边内侧	正方向
			9	正方向	原点正边内侧	负方向
			10	正方向	原点正边外侧	正方向
安装负限开关	安装原点开关		11	负方向	原点正边外侧	正方向
			12	负方向	原点正边内侧	负方向
			13	负方向	原点负边内侧	正方向
			14	负方向	原点负边外侧	负方向
			33		负方向 Index	负方向
			34		正方向 Index	正方向

表 6-2

回原点函数介绍

函数原型	short **MCF_Search_Home_Set_Net** (unsigned short Axis, unsigned short Search_Home_Mode, unsigned short Limit_Logic, unsigned short Home_Logic, unsigned short Index_Logic, double H_dMaxV, double L_dMaxV, long Offset_Position, unsigned short Trigger_Source: unsigned short StationNumber = 0);
使用说明	设置回零参数
参数说明	Axis：设置轴号 Search_Home_Mode: 回零方式（参考回零方式）取值：1 ~ 35 Limit_Logic： 正负限位触发电平 取值: 0：低电平(触点闭合,硬件灯亮)

	1：高电平(触点断开,硬件灯灭) Home_Logic： 原点限位触发电平 取值: 0：低电平(触点闭合,硬件灯亮) 1：高电平(触点断开,硬件灯灭) Index_Logic： Index 触发电平 取值: 0：低电平(触点闭合) 1：高电平(触点断开) H_dMaxV： 高速段速度 L_dMaxV： 低速段速度 Offset_Position： 偏移位置：按回零模式图示箭头指示方向偏移 Trigger_Source :捕捉位置模式 取值: 0：指令位置 1：编码器位置 StationNumber： 站点号

函数原型	short **MCF_Search_Home_dMaxA_Time_Net** (unsigned short Axis, unsigned short H_dMaxA_Time = 10, unsigned short L_dMaxA_Time = 10, unsigned short StationNumber = 0);
使用说明	设置回零参数
参数说明	Axis：设置轴号 H_dMaxA_Time: 高速段加速时间； 单位：MS L_dMaxA_Time: 低速段加速时间； 单位：MS StationNumber： 站点号

函数原型	short **MCF_Search_Home_Start_Net** (unsigned short Axis, unsigned short StationNumber);
使用说明	开始回零
参数说明	Axis：设置轴号 StationNumber： 站点号

函数原型	short **MCF_Search_Home_Stop_Net** (unsigned short Axis, unsigned short StationNumber);
使用说明	停止回零
参数说明	Axis：设置轴号 StationNumber： 站点号

函数原型	short **MCF_Search_Home_Stop_Time_Net** (unsigned short Axis, unsigned short Stop_Time, unsigned short StationNumber);
使用说明	设置碰撞原点缓停时间 （ 该函数必须在设置回零参数 MCF_Search_Home_Set_Net 前设置 ）
参数说明	Axis：设置轴号 Stop_Time: 触发缓停时间； 范围 0 -1000ms； （默认：急停） StationNumber： 站点号 注意： 设置碰撞原点缓停时间，需要配合回零设置的高速段速度，和低速度速度的搭配。设置大了会出现过冲的现象，设置小了停止的时候会有抖动， 需要调节时间，尽量设置找到原点时候在开关中间。

函数原型	short **MCF_Search_Home_Keep_Position_Net** (unsigned short Axis, unsigned short StationNumber);
使用说明	设置回零完成后保持位置值
参数说明	Axis：设置轴号 StationNumber： 站点号

函数原型	short **MCF_Search_Home_Keep_Encoder_Net** (unsigned short Axis, unsigned short StationNumber);
使用说明	设置回零完成后保持编码器值
参数说明	Axis：设置轴号 StationNumber： 站点号

函数原型	short **MCF_Search_Home_Leave_Vel_Net** (unsigned short Axis, double M_dMaxV, unsigned short StationNumber);
使用说明	设置回零在零点位置离开速度
参数说明	Axis：设置轴号 M_dMaxV: 触碰原点后，离开原点的速度； StationNumber： 站点号 注意： 设置在原点限位上， 离开原点限位的速度加快。主要用在二次回零的速度偏慢（保证精度）的情况。

回零模式 1

回零方式1：以负方向限位的正方向端为参考点，以正方向第一个 Z 信号为零点，往正方向偏移位置为客户设定原点。

动作流程：

1、朝负方向寻找负限位

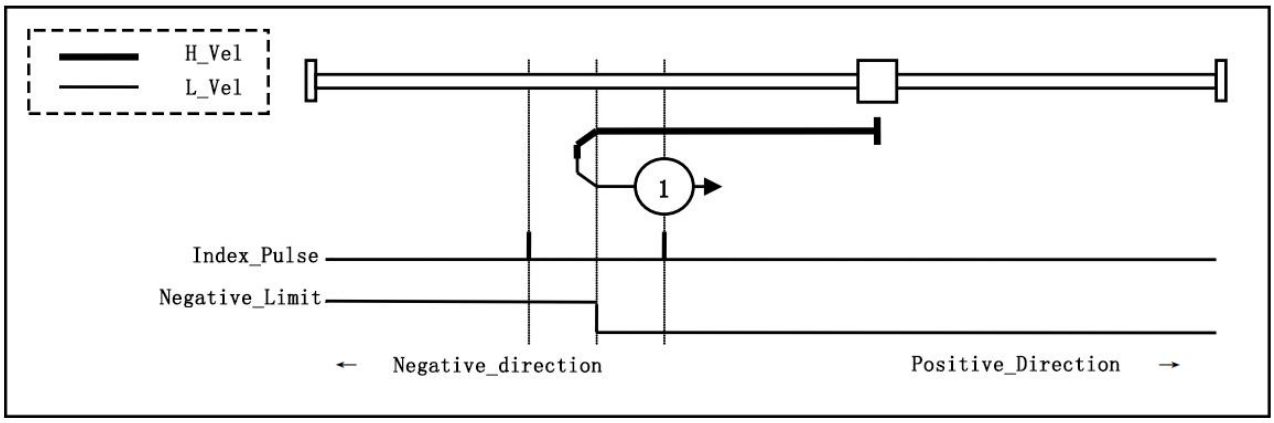
2、触发负限位，调整至负限位正方向边缘外侧

3、开始朝正方向寻找Z信号

4、触发Z信号，开始朝正方向偏移指定距离

5、偏移结束，回零完成

负限位正方向边缘外侧：负限位信号触发时朝正方向移动直至负限位信号消失瞬间。



回零模式 2

回零方式2：以正方向限位的负方向端为参考点，以负方向第一个 Z 信号为零点，往负方向偏移位置为客户设定原点。

动作流程：

1、朝正方向寻找正限位

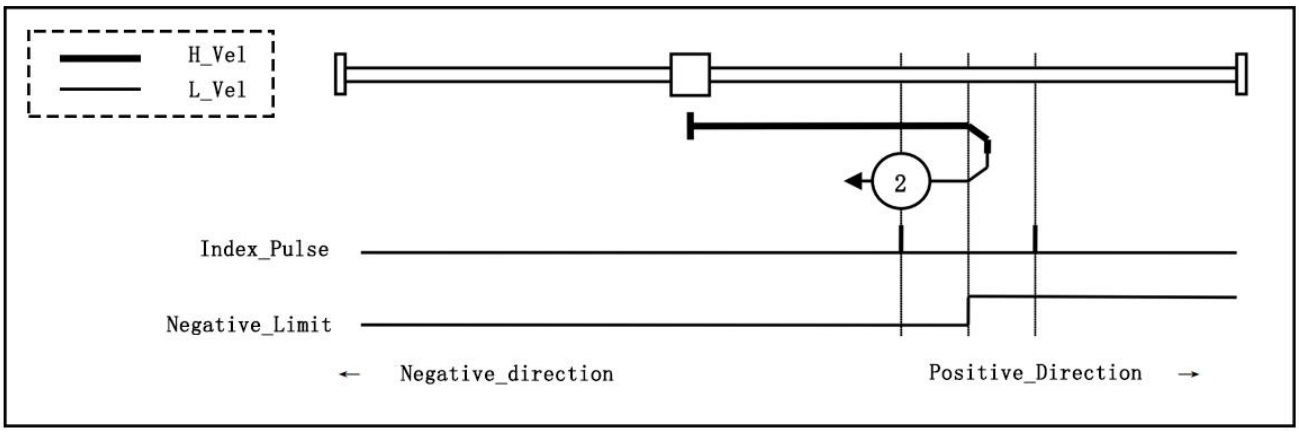
2、触发正限位，调整至正限位负方向边缘外侧

3、开始朝负方向寻找Z信号

4、触发Z信号，开始朝负方向偏移指定距离

5、偏移结束，回零完成

正限位负方向边缘外侧：正限位信号触发时朝负方向移动直至正限位信号消失瞬间。



回零模式 3

回零方式3：以原点限位的负方向端为参考点，以负方向第一个 Z 信号为零点，往负方向偏移位置为客户设定原点。

动作流程：

1、朝正方向寻找原点限位

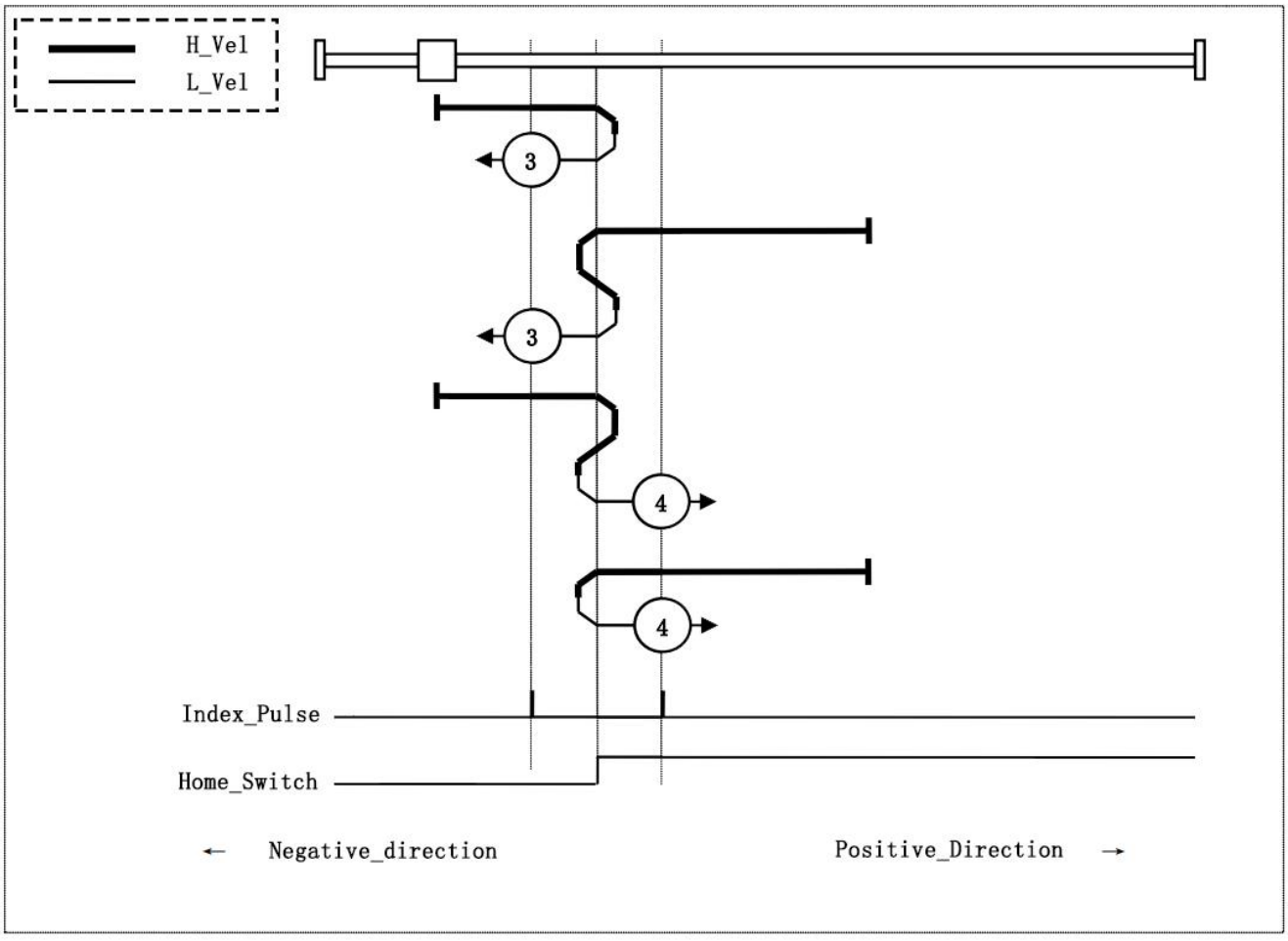
2、触发原点限位，调整至原点限位负方向边缘外侧

3、朝负方向寻找Z信号

4、触发Z信号，开始朝负方向偏移指定距离

5、偏移结束，回零完成

原点限位负方向边缘外侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间。



回零模式 4

回零方式4：以原点限位的负方向端为参考点，以正方向第一个 Z 信号为零点，往正方向偏移位置为客户设定原点。

动作流程：

1、朝正方向寻找原点限位

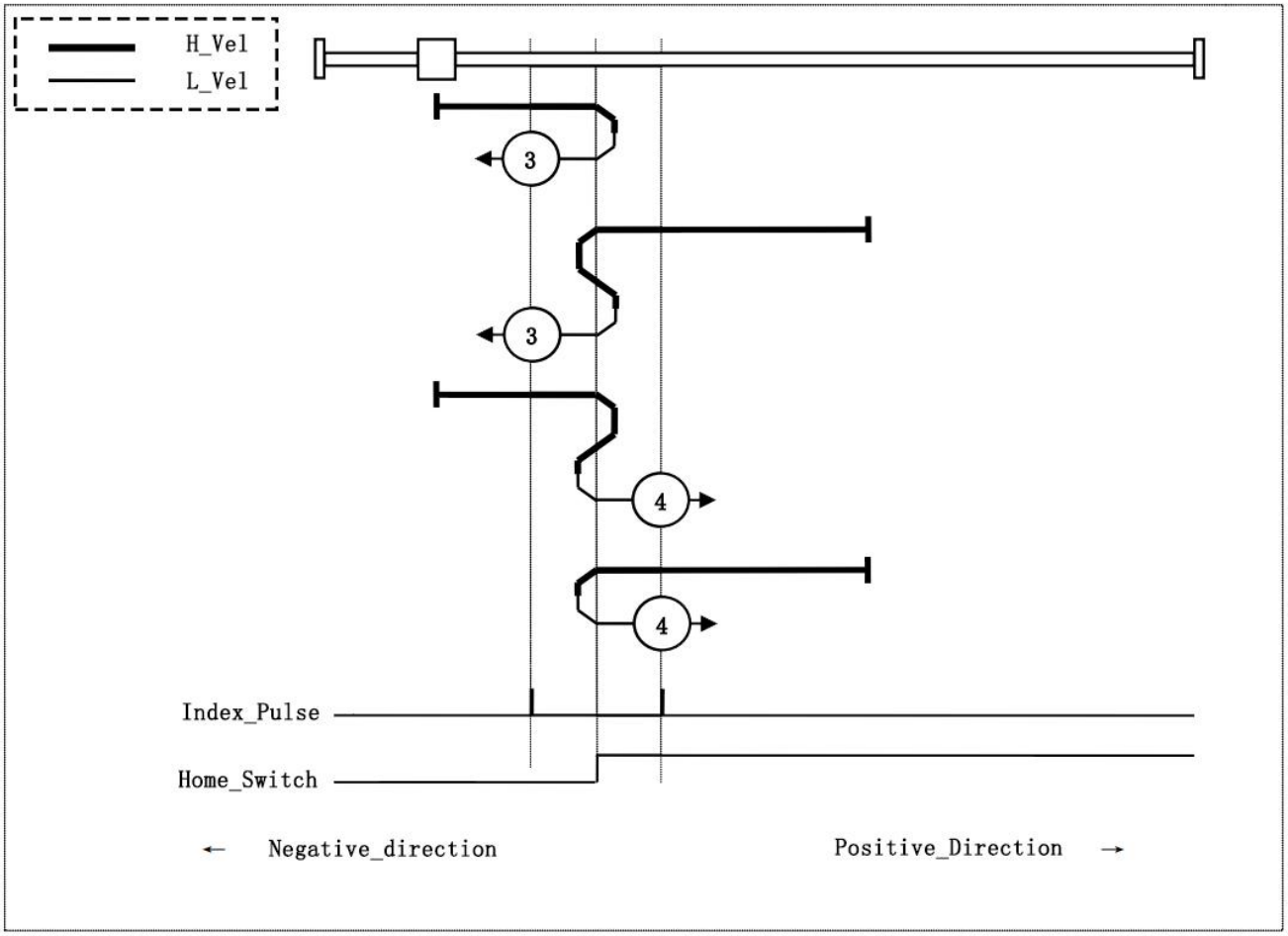
2、触发原点限位，调整至原点限位负方向边缘内侧

3、朝正方向寻找Z信号

4、触发Z信号，开始朝正方向偏移指定距离

5、偏移结束，回零完成

原点限位负方向边缘内侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 5

回零方式5：以原点限位的正方向端为参考点，以正方向第一个 Z 信号为零点，往正方向偏移位置为客户设定原点。

动作流程：

1、朝负方向寻找原点限位

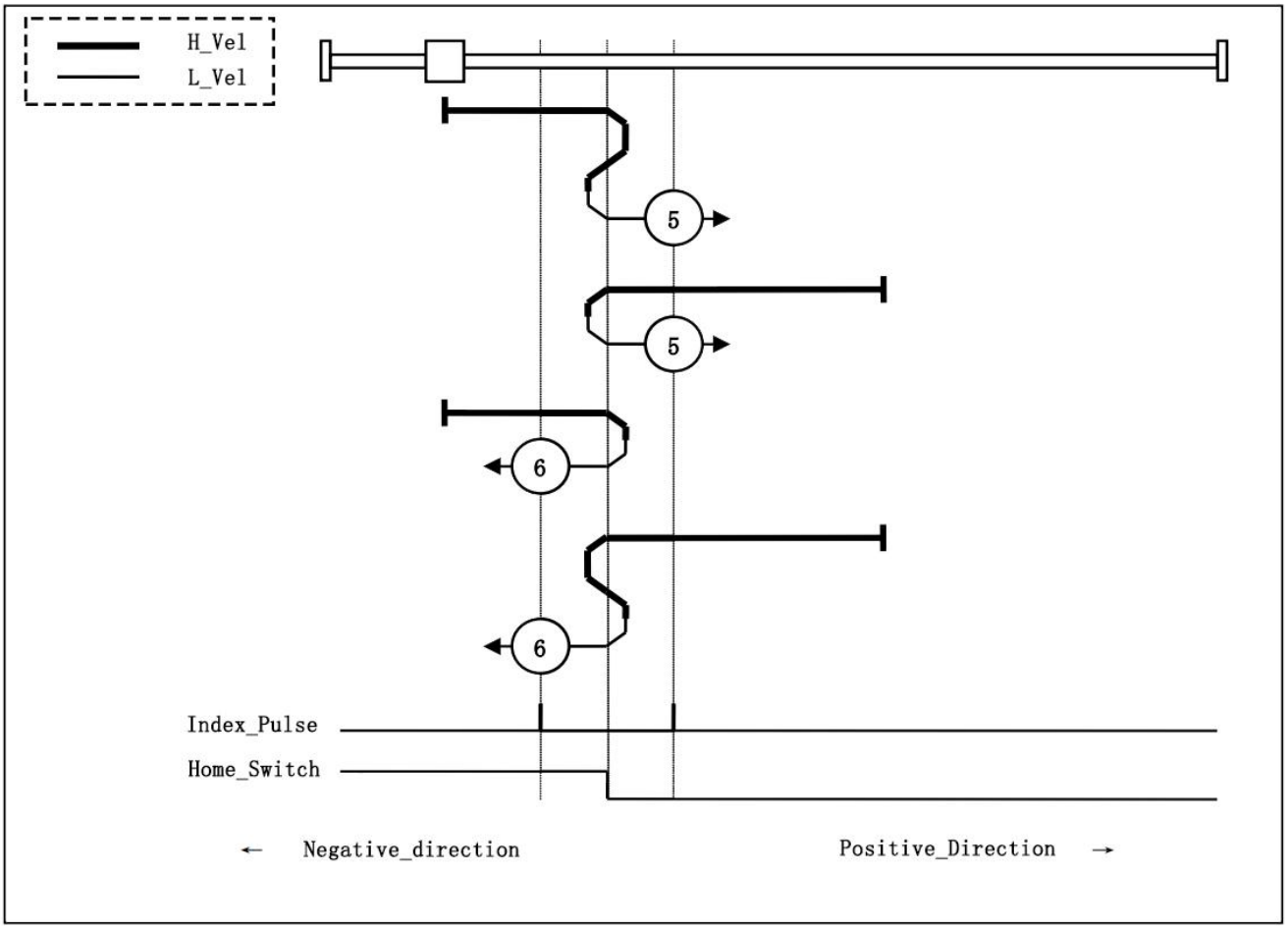
2、触发原点限位，调整至原点限位正方向边缘外侧

3、朝正方向寻找Z信号

4、触发Z信号，开始朝正方向偏移指定距离

5、偏移结束，回零完成

原点限位正边缘外侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间。



回零模式 6

回零方式6：以原点限位的负方向端为参考点，以负方向第一个 Z 信号为零点，往负方向偏移位置为客户设定原点。

动作流程：

1、朝负方向寻找原点限位

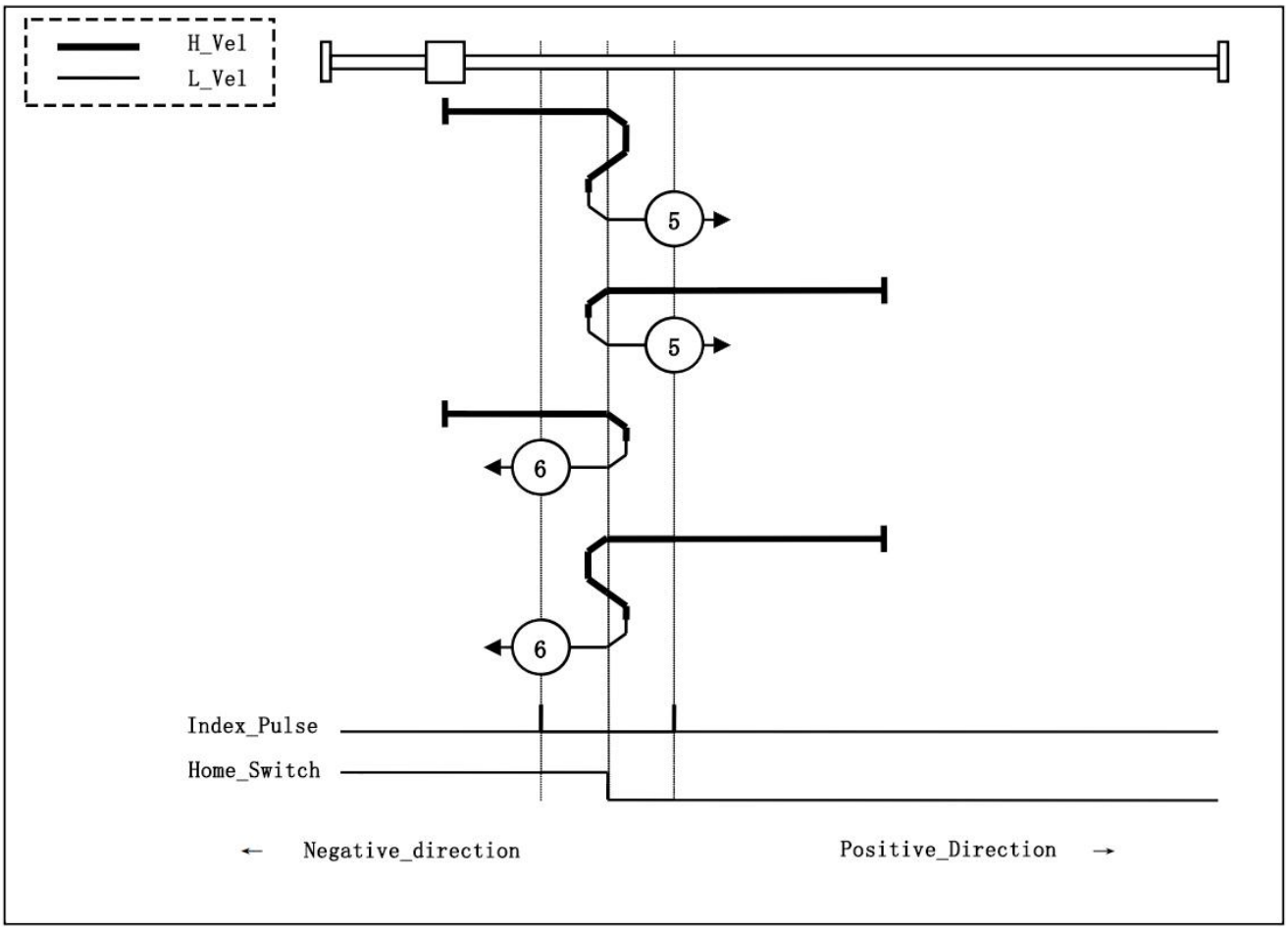
2、触发原点限位，调整至原点限位正方向边缘内侧

3、朝负方向寻找Z信号

4、触发Z信号，开始朝负方向偏移指定距离

5、偏移结束，回零完成

原点限位正方向边缘内侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 7

回零方式7：以原点限位的负方向端为参考点，以负方向第一个 Z 信号为零点，往负方向偏移位置为客户设定原点。

动作流程：

1、朝正方向寻找原点限位，如果搜索过程中触发正限位，则反方向寻找原点限位，若反方向寻找时触发负限位，则回零失败

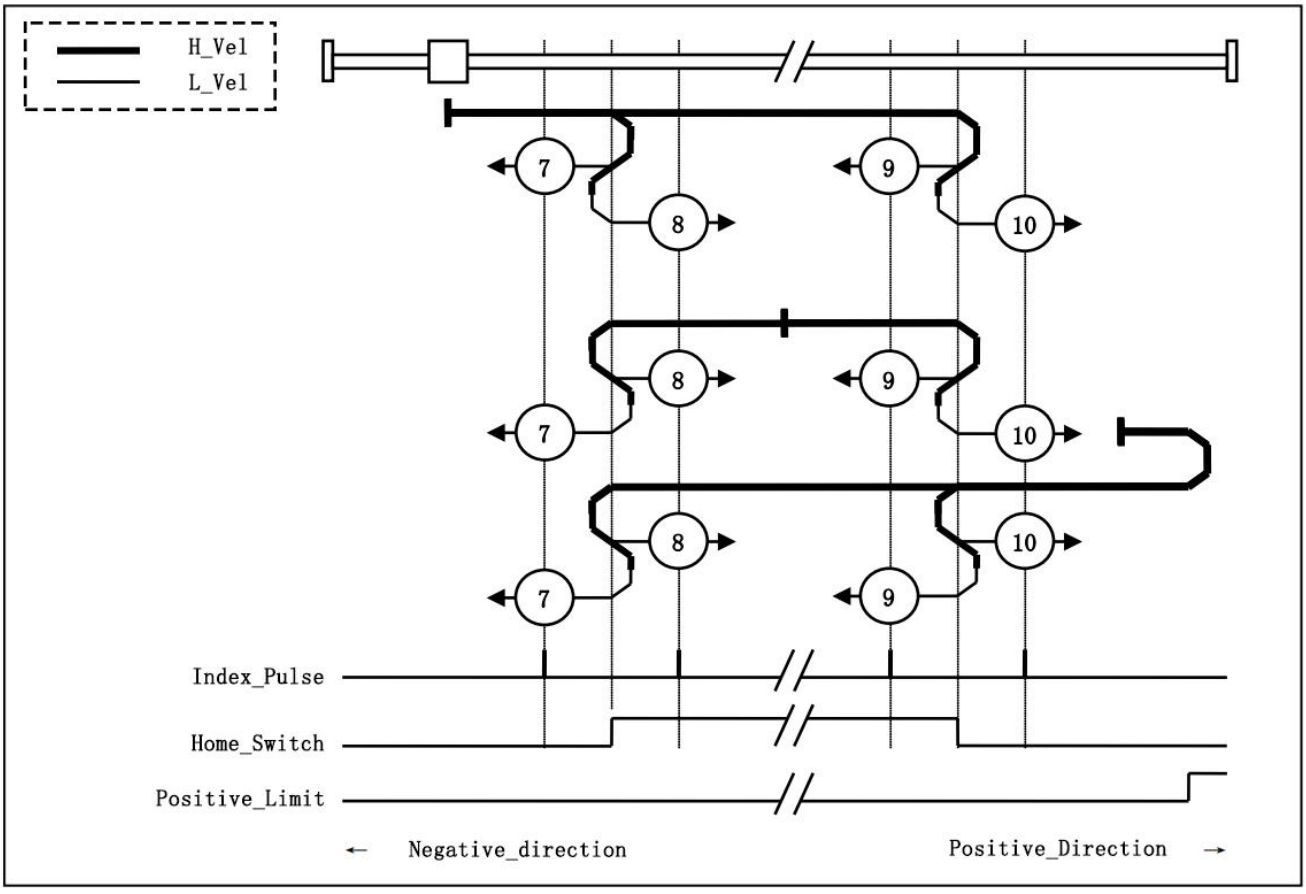
2、触发原点限位，调整至原点限位负方向边缘外侧

3、朝负方向寻找Z信号

4、触发Z信号，开始朝负方向偏移指定距离

5、偏移结束，回零完成

原点限位负方向边缘外侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间。



回零模式 8

回零方式8：以原点限位的负方向端为参考点，以正方向第一个 Z 信号为零点，往正方向偏移位置为客户设定原点。

动作流程：

1、朝正方向寻找原点限位，如果搜索过程中触发正限位，则反方向寻找原点限位，若反方向寻找时触发负限位，则回零失败

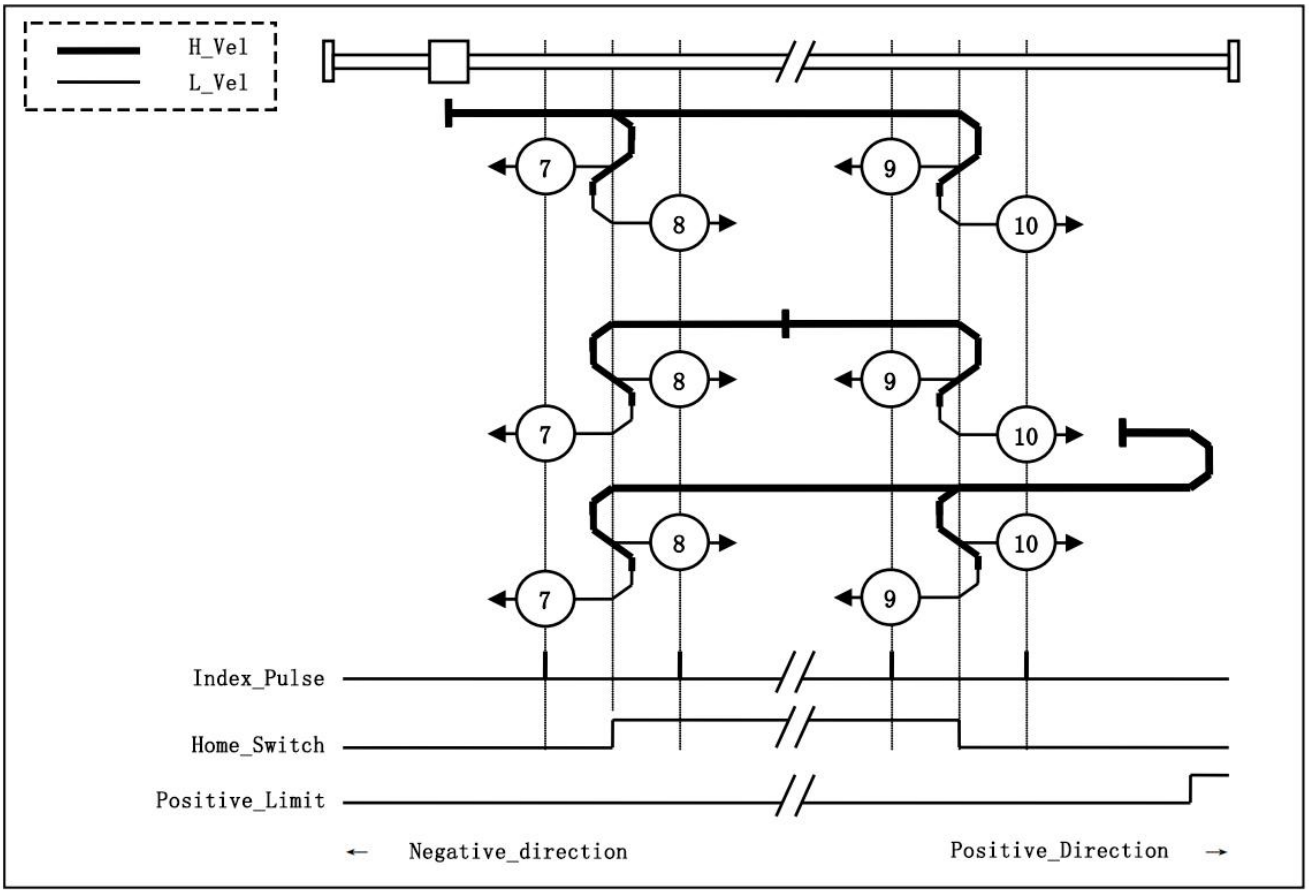
2、触发原点限位，调整至原点限位负方向边缘内侧

3、朝正方向寻找Z信号

4、触发Z信号，开始朝正方向偏移指定距离

5、偏移结束，完成回零

原点限位负方向边缘内侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 9

回零方式9：以原点限位的正方向端为参考点，以负方向第一个 Z 信号为零点，往负方向偏移位置为客户设定原点。

动作流程：

1、朝正方向寻找原点限位，如果搜索过程中触发正限位，则反方向寻找原点限位，若反方向寻找时触发负限位，则回零失败

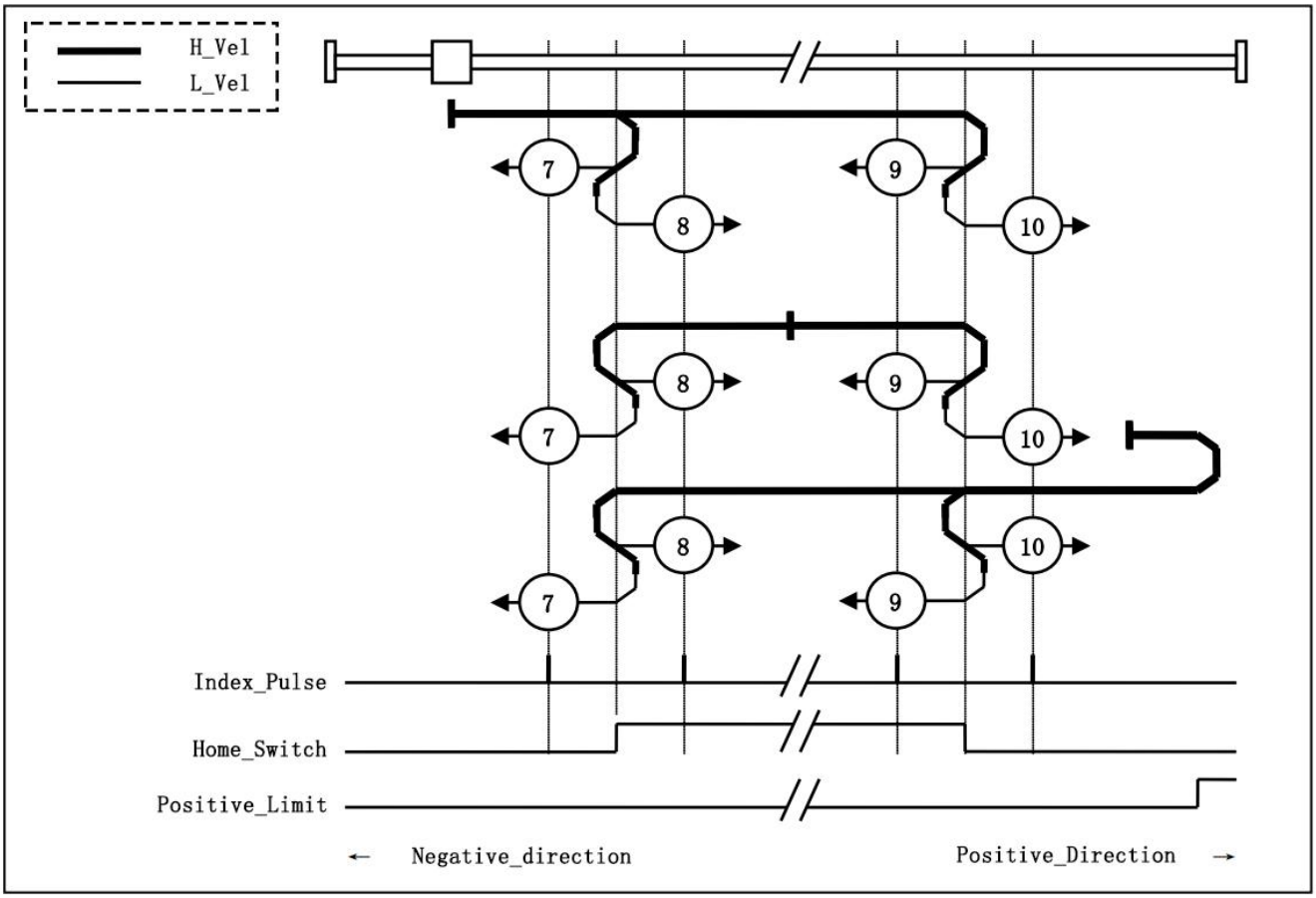
2、触发原点限位，调整至原点限位正方向边缘内侧

3、朝负方向寻找Z信号

4、触发Z信号，开始朝负方向偏移指定距离

5、偏移结束，回零完成

原点限位正方向边缘内侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 10

回零方式10：以原点限位的正方向端为参考点，以正方向第一个 Z 信号为零点，往正方向偏移位置为客户设定原点。

动作流程：

1、朝正方向寻找原点限位，如果搜索过程中触发正限位，则反方向寻找原点限位，若反方向寻找时触发负限位，则回零失败

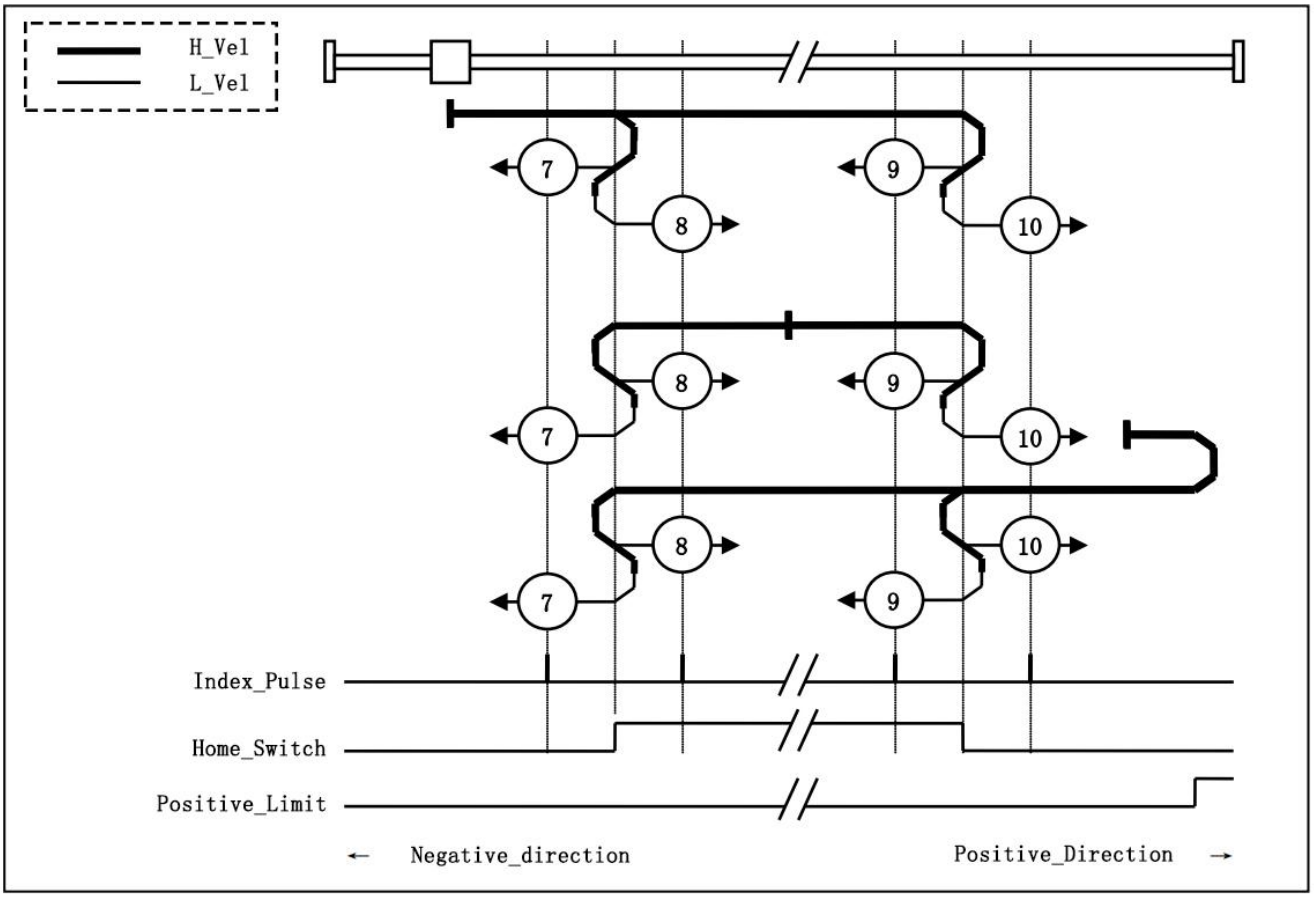
2、触发原点限位，调整至原点限位正方向边缘外侧

3、朝正方向寻找Z信号

4、触发Z信号，开始朝正方向偏移指定距离

5、偏移结束，完成回零

原点限位正方向边缘外侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间。



回零模式 11

回零方式11：以原点限位的正方向端为参考点，以正方向第一个 Z 信号为零点，往正方向偏移位置为客户设定原点。

动作流程：

1、朝负方向寻找原点限位，如果搜索过程中触发负限位，则反方向寻找原点限位，若反方向寻找时触发正限位，则回零失败

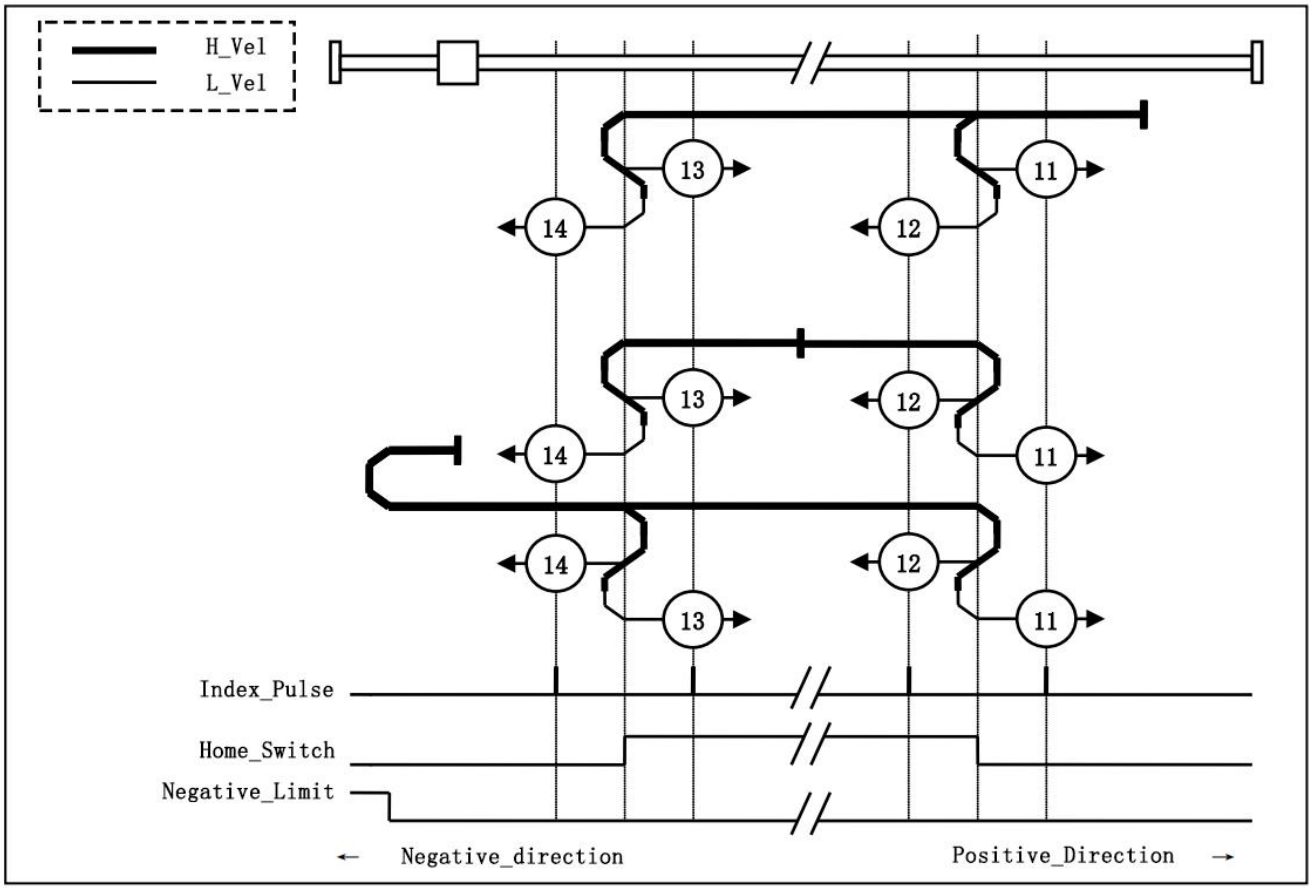
2、触发原点限位，调整至原点限位正方向边缘外侧

3、朝正方向寻找Z信号

4、触发Z信号，开始朝正方向偏移指定距离

5、偏移结束，回零完成

原点限位正方向边缘外侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间。



回零模式 12

回零方式12：以原点限位的正方向端为参考点，以负方向第一个 Z 信号为零点，往负方向偏移位置为客户设定原点。

动作流程：

1、朝负方向寻找原点限位，如果搜索过程中触发负限位，则反方向寻找原点限位，若反方向寻找时触发正限位，则回零失败

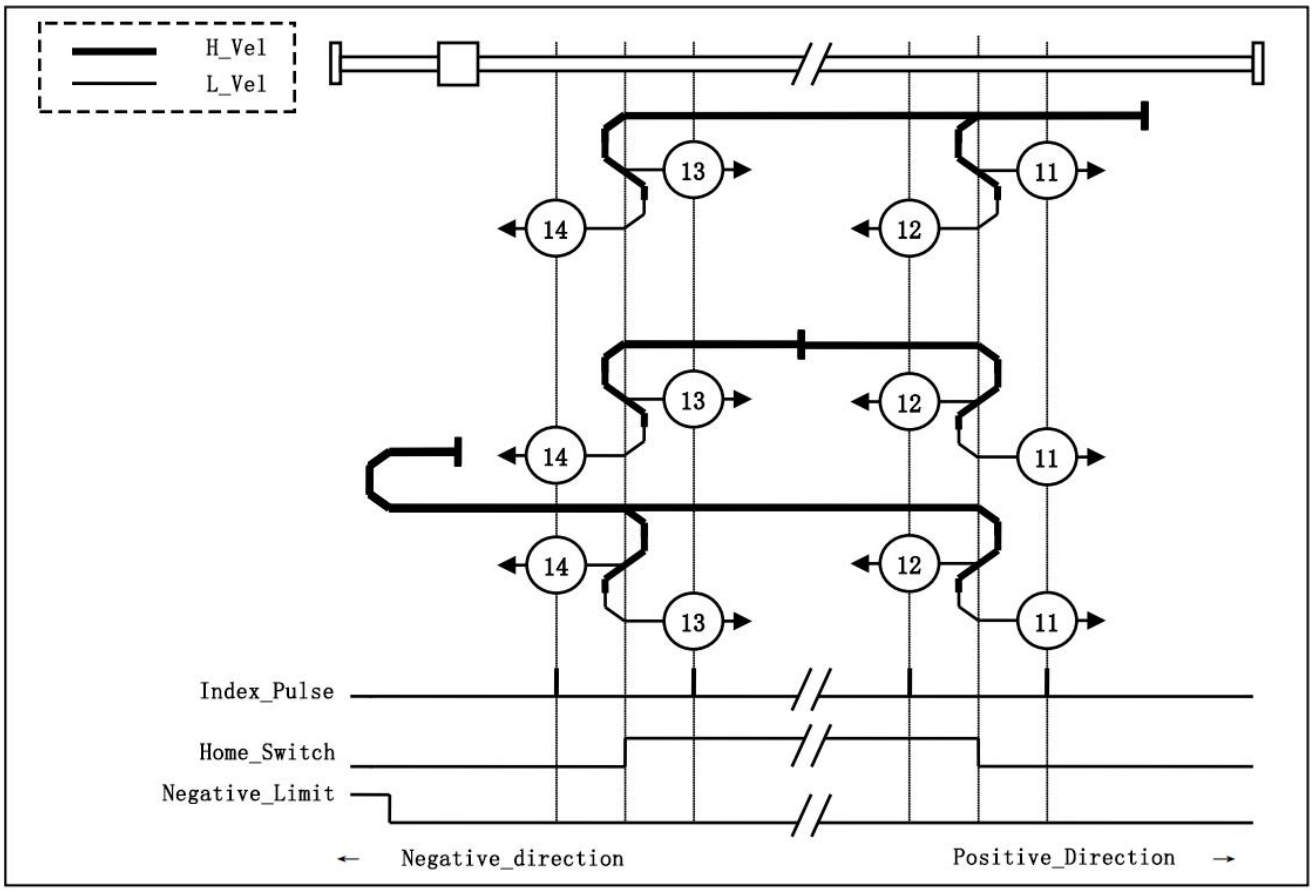
2、触发原点限位，调整至原点限位正方向边缘内侧

3、朝负方向寻找Z信号

4、触发Z信号，开始朝负方向偏移指定距离

5、偏移结束，回零完成

原点限位正方向边缘内侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 13

回零方式13：以原点限位的负方向端为参考点，以正方向第一个 Z 信号为零点，往正方向偏移位置为客户设定原点。

动作流程：

1、朝负方向寻找原点限位，如果搜索过程中触发负限位，则反方向寻找原点限位，若反方向寻找时触发正限位，则回零失败

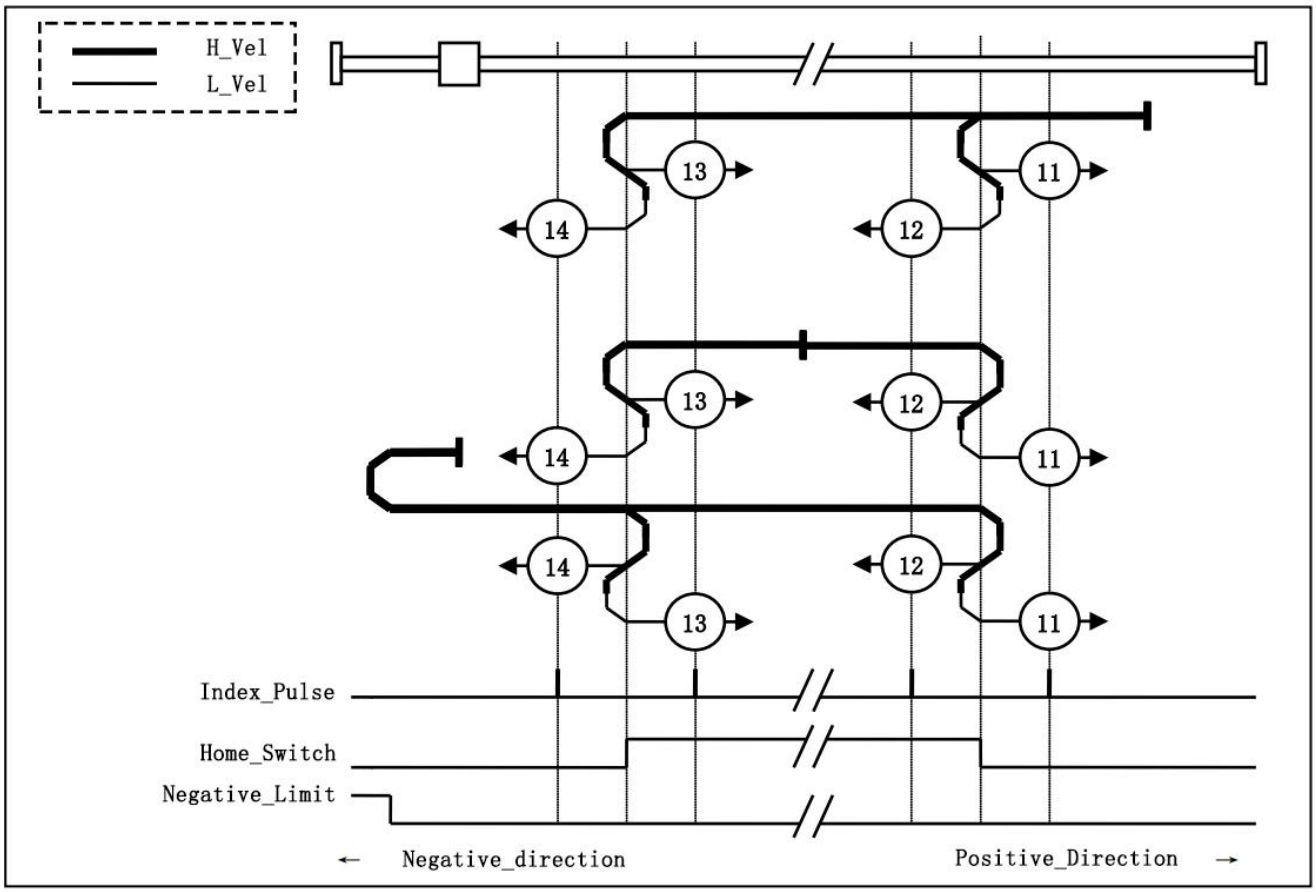
2、触发原点限位，调整至原点限位负方向边缘内侧

3、朝正方向寻找Z信号

4、触发Z信号，开始朝正方向偏移指定距离

5、偏移结束，回零完成

原点限位负方向边缘内侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 14

回零方式14：以原点限位的负方向端为参考点，以负方向第一个 Z 信号为零点，往负方向偏移位置为客户设定原点。

动作流程：

1、朝负方向寻找原点限位，如果搜索过程中触发负限位，则反方向寻找原点限位，若反方向寻找时触发正限位，则回零失败

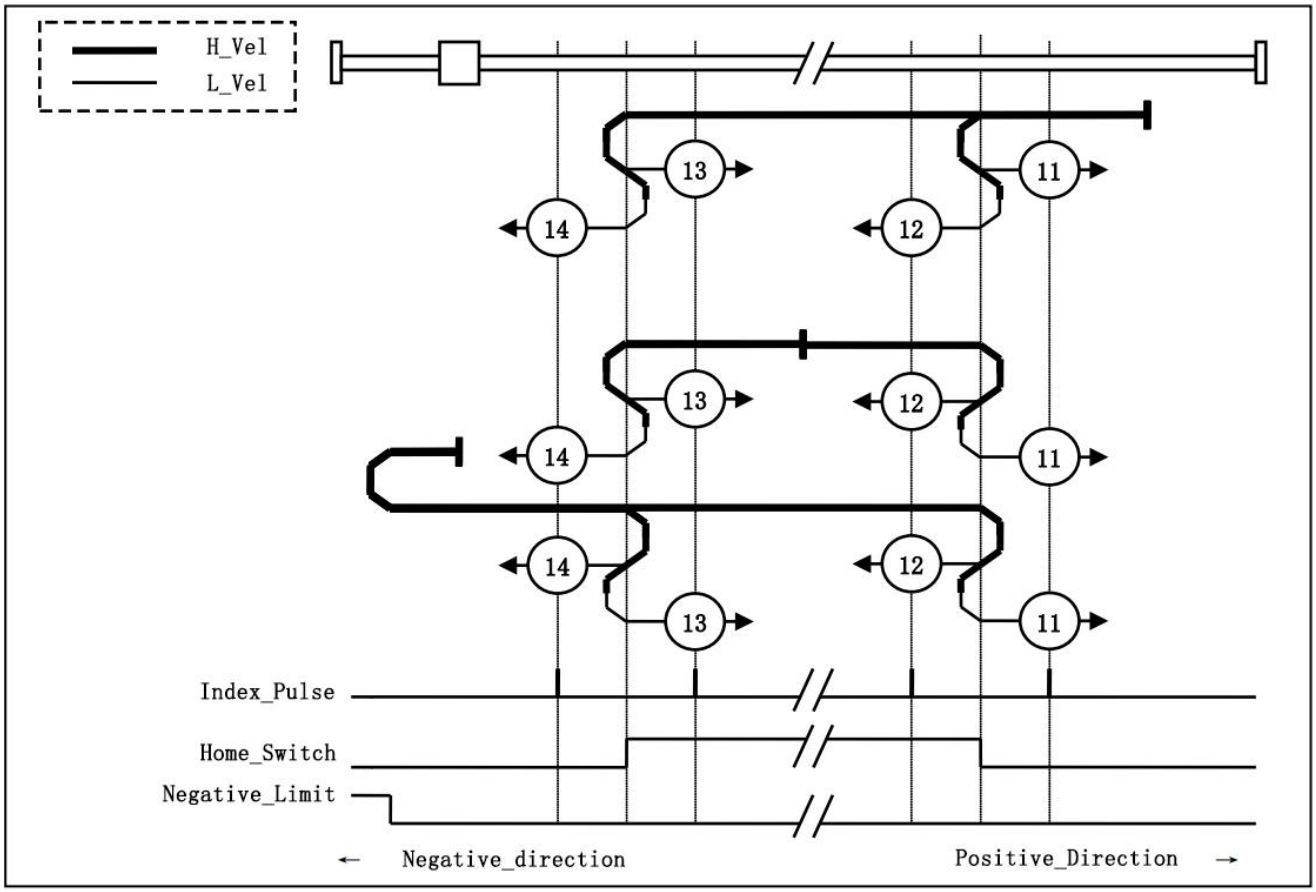
2、触发原点限位，调整至原点限位负方向边缘外侧

3、朝负方向寻找Z信号

4、触发Z信号，开始朝负方向偏移指定距离

5、偏移结束，回零完成

原点限位负方向边缘外侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间。



回零模式 17

回零方式17：以负方向限位的正方向端为参考点，往正方向偏移位置为客户设定原点。

动作流程：

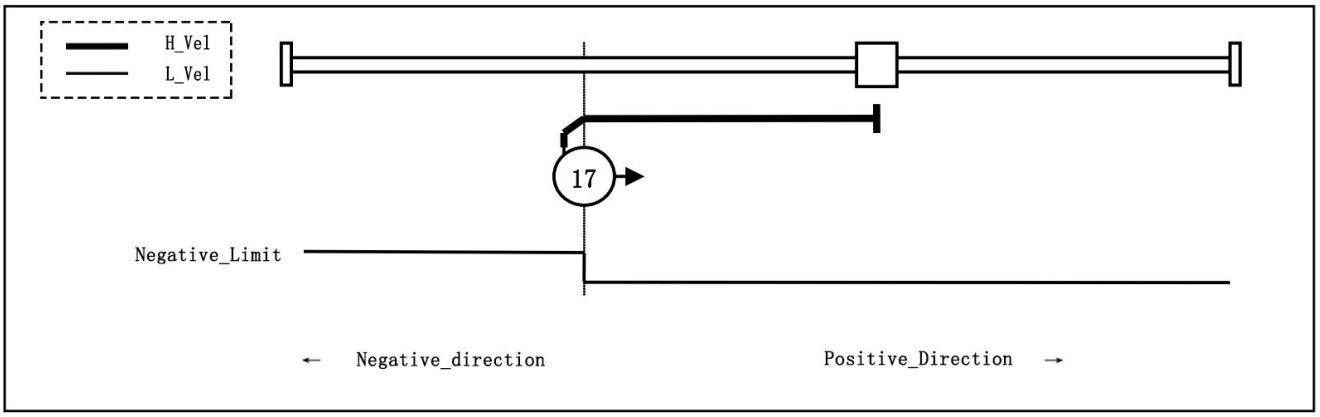
1、朝负方向寻找负限位

2、触发负限位，调整至负限位正方向边缘外侧

3、朝正方向偏移指定距离

4、偏移结束，回零完成

负限位正方向边缘外侧：负限位信号触发时朝正方向移动直至负限位信号消失瞬间。



回零模式 18

回零方式18：以正方向限位的负方向端为参考点，往负方向偏移位置为客户设定原点。

动作流程：

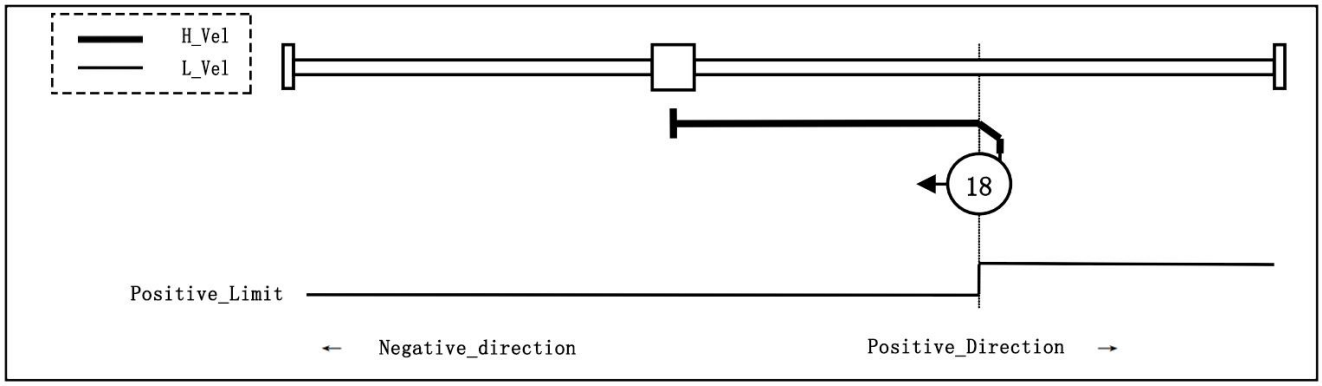
1、朝正方向寻找正限位

2、触发正限位，调整至正限位负方向边缘外侧

3、朝负方向偏移指定距离

4、偏移结束，回零完成

正限位负方向边缘外侧：正限位信号触发时朝负方向移动直至正限位信号消失瞬间。



回零模式 19

回零方式19：以原点限位的负方向端为参考点，往负方向偏移位置为客户设定原点。

动作流程：

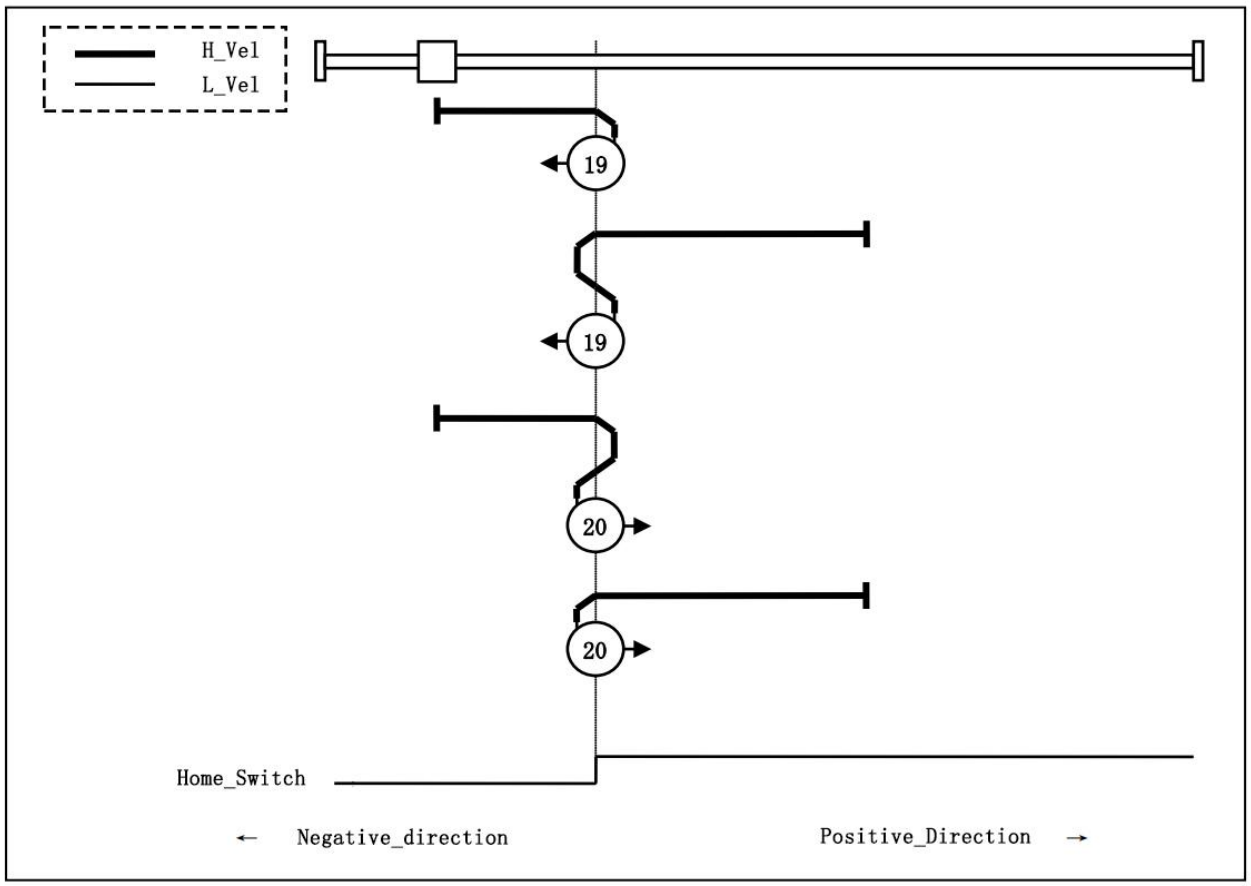
1、朝正方向寻找原点限位

2、触发原点限位，调整至原点限位负方向边缘外侧

3、朝负方向偏移指定距离

4、偏移结束，回零完成

原点限位负方向边缘外侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间。



回零模式 20

回零方式20：以原点限位的负方向端为参考点，往正方向偏移位置为客户设定原点。

动作流程：

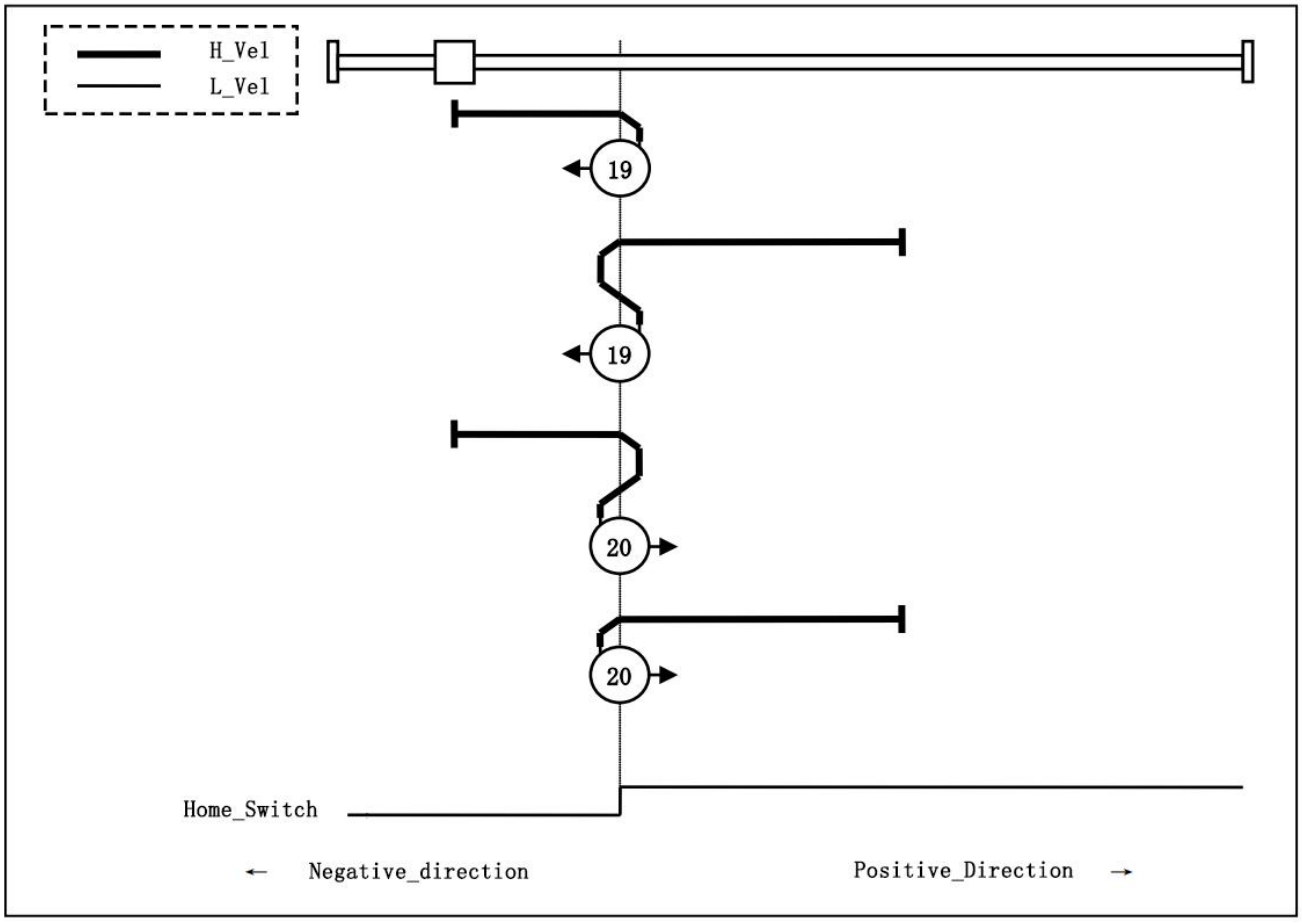
1、朝正方向寻找原点限位

2、触发原点限位，调整至原点限位负方向边缘内侧

3、朝正方向偏移指定距离

4、偏移结束，回零完成

原点限位负方向边缘内侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 21

回零方式21：以原点限位的正方向端为参考点，往正方向偏移位置为客户设定原点。

动作流程：

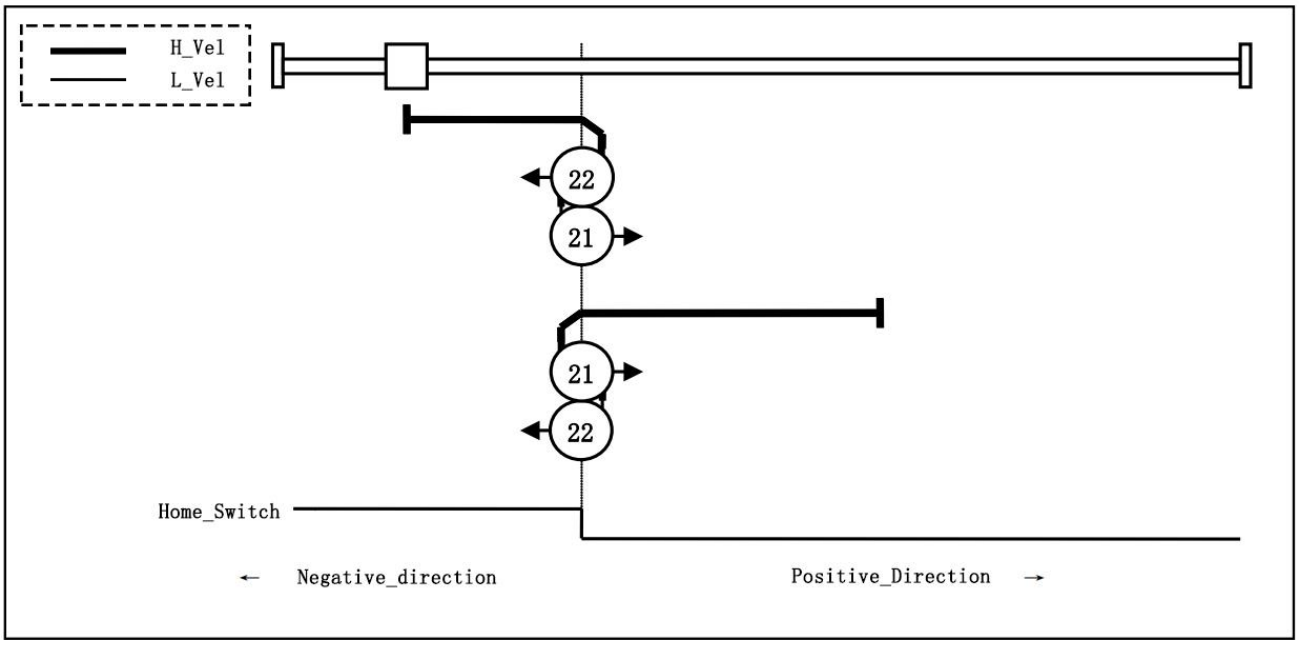
1、朝负方向寻找原点限位

2、触发原点限位，调整至原点限位正方向边缘外侧

3、朝正方向偏移指定距离

4、偏移结束，回零完成

原点限位正方向边缘外侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间。



回零模式 22

回零方式22：以原点限位的正方向端为参考点，往负方向偏移位置为客户设定原点。

动作流程：

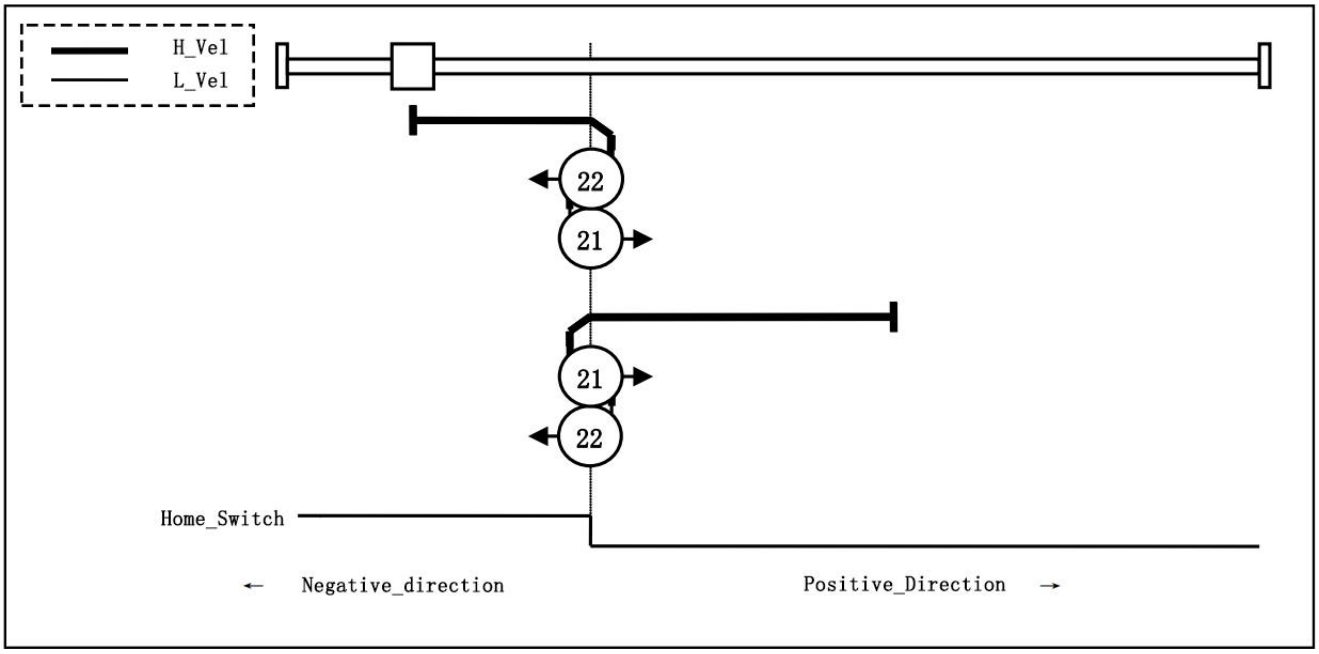
1、朝负方向寻找原点限位

2、触发原点限位，调整至原点限位正方向边缘内侧

3、朝负方向偏移指定距离

4、偏移结束，回零完成

原点限位正方向边缘内侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 23

回零方式23：以原点限位的负方向端为参考点，往负方向偏移位置为客户设定原点。

动作流程：

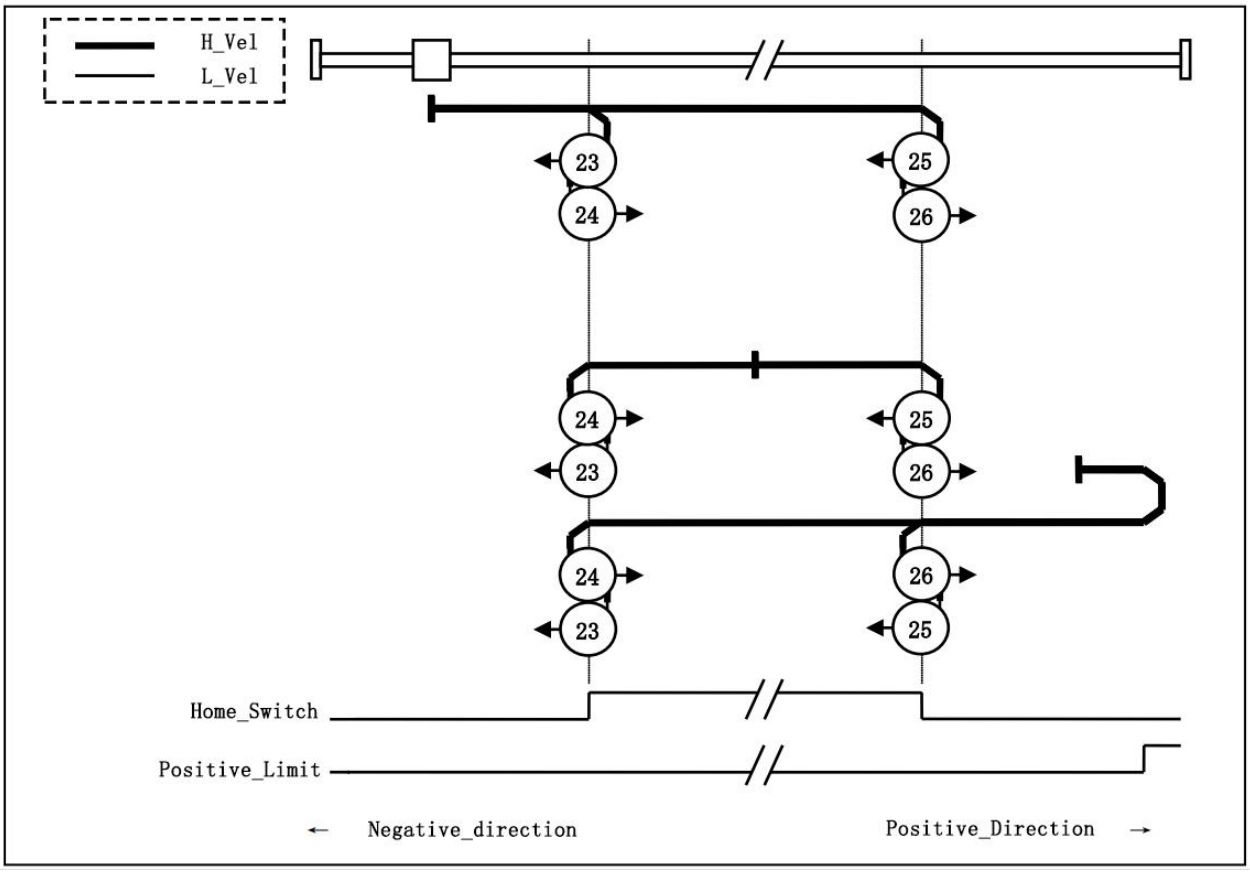
1、朝正方向寻找原点限位，如果搜索过程中触发正限位，则反方向寻找原点限位，若反方向寻找时触发负限位，则回零失败

2、触发原点限位，调整至原点限位负方向边缘外侧

3、开始朝负方向偏移指定距离

4、偏移结束，回零完成

原点限位负方向边缘外侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间。



回零模式 24

回零方式24：以原点限位的负方向端为参考点，往正方向偏移位置为客户设定原点。

动作流程：

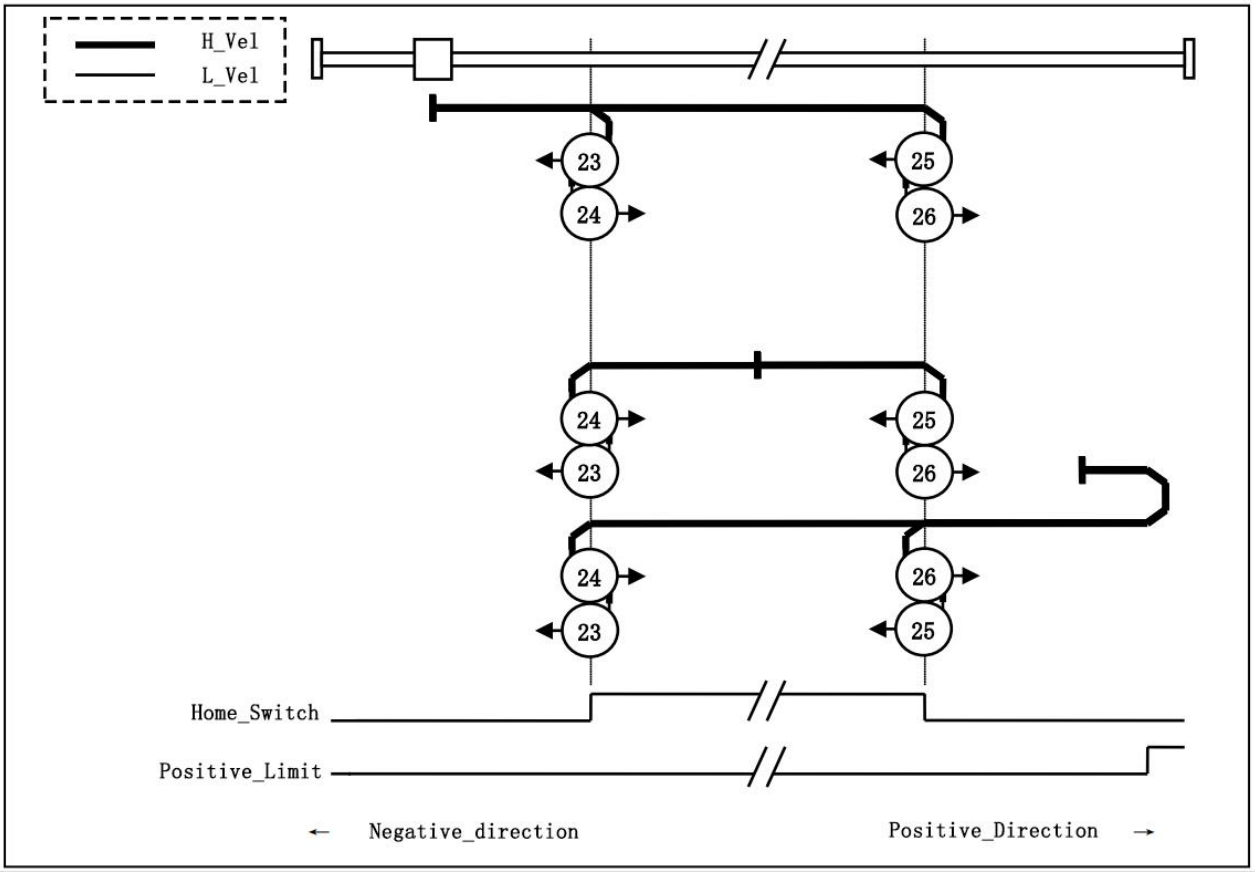
1、朝正方向寻找原点限位，如果搜索过程中触发正限位，则反方向寻找原点限位，若反方向寻找时触发负限位，则回零失败

2、触发原点限位，调整至原点限位负方向边缘内侧

3、开始朝正方向偏移指定距离

4、偏移结束，回零完成

原点限位负方向边缘内侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 25

回零方式25：以原点限位的正方向端为参考点，往负方向偏移位置为客户设定原点。

动作流程：

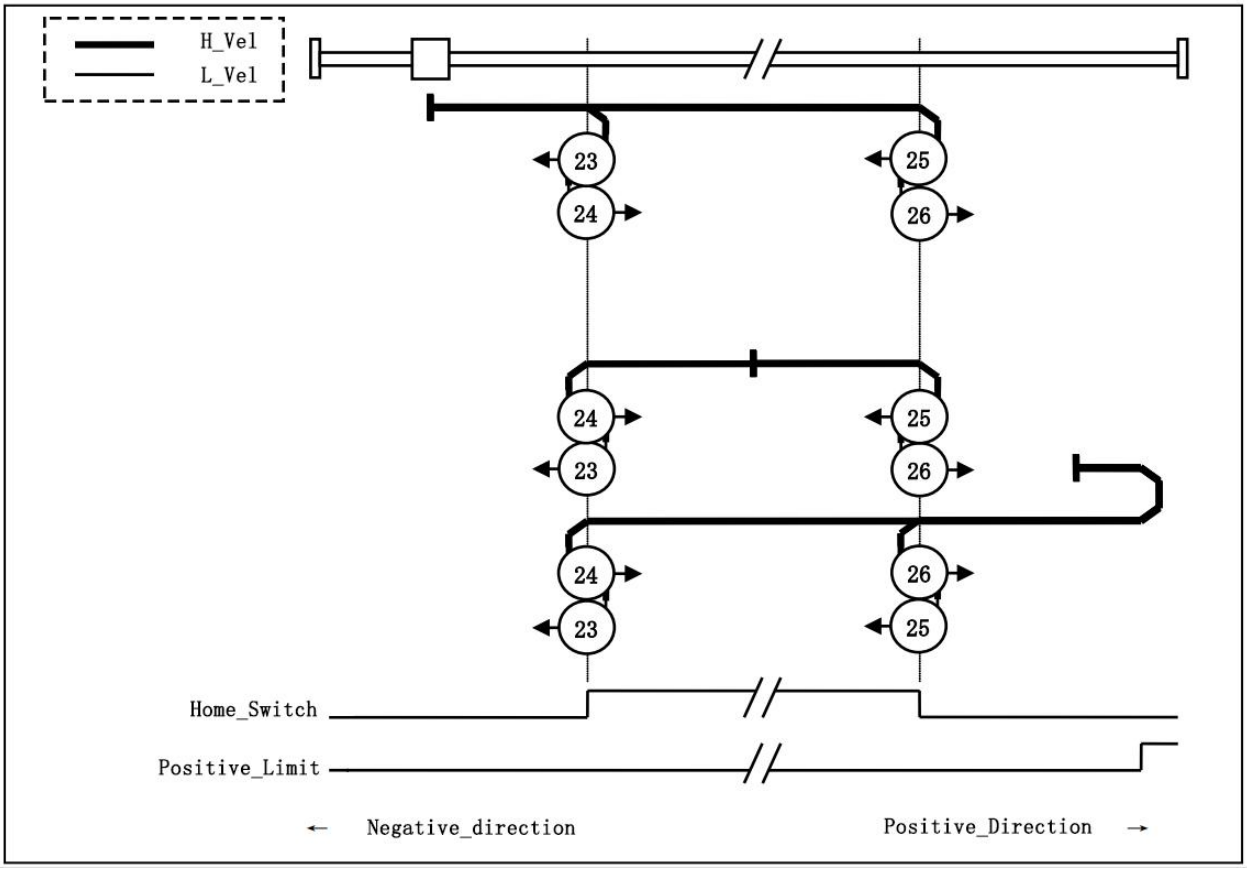
1、朝正方向寻找原点限位，如果搜索过程中触发正限位，则反方向寻找原点限位，若反方向寻找时触发负限位，则回零失败

2、触发原点限位，调整至原点限位正方向边缘内侧

3、开始朝负方向偏移指定距离

4、偏移结束，回零完成

原点限位正方向边缘内侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 26

回零方式26：以原点限位的正方向端为参考点，往正方向偏移位置为客户设定原点。

动作流程：

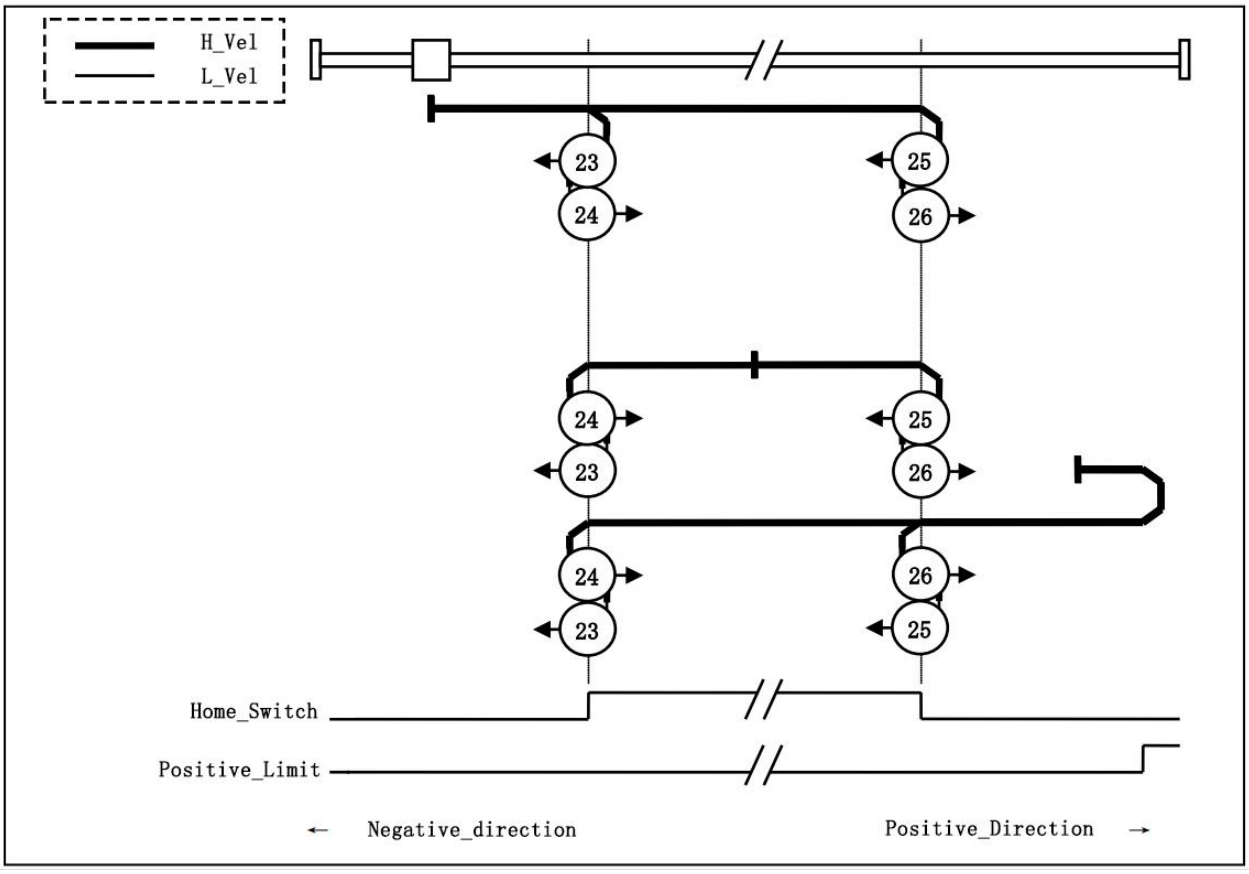
1、朝正方向寻找原点限位，如果搜索过程中触发正限位，则反方向寻找原点限位，若反方向寻找时触发负限位，则回零失败

2、触发原点限位，调整至原点限位正方向边缘外侧

3、开始朝正方向偏移指定距离

4、偏移结束，回零完成

原点限位正方向边缘外侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间。



回零模式 27

回零方式27：以原点限位的正方向端为参考点，往正方向偏移位置为客户设定原点。

动作流程：

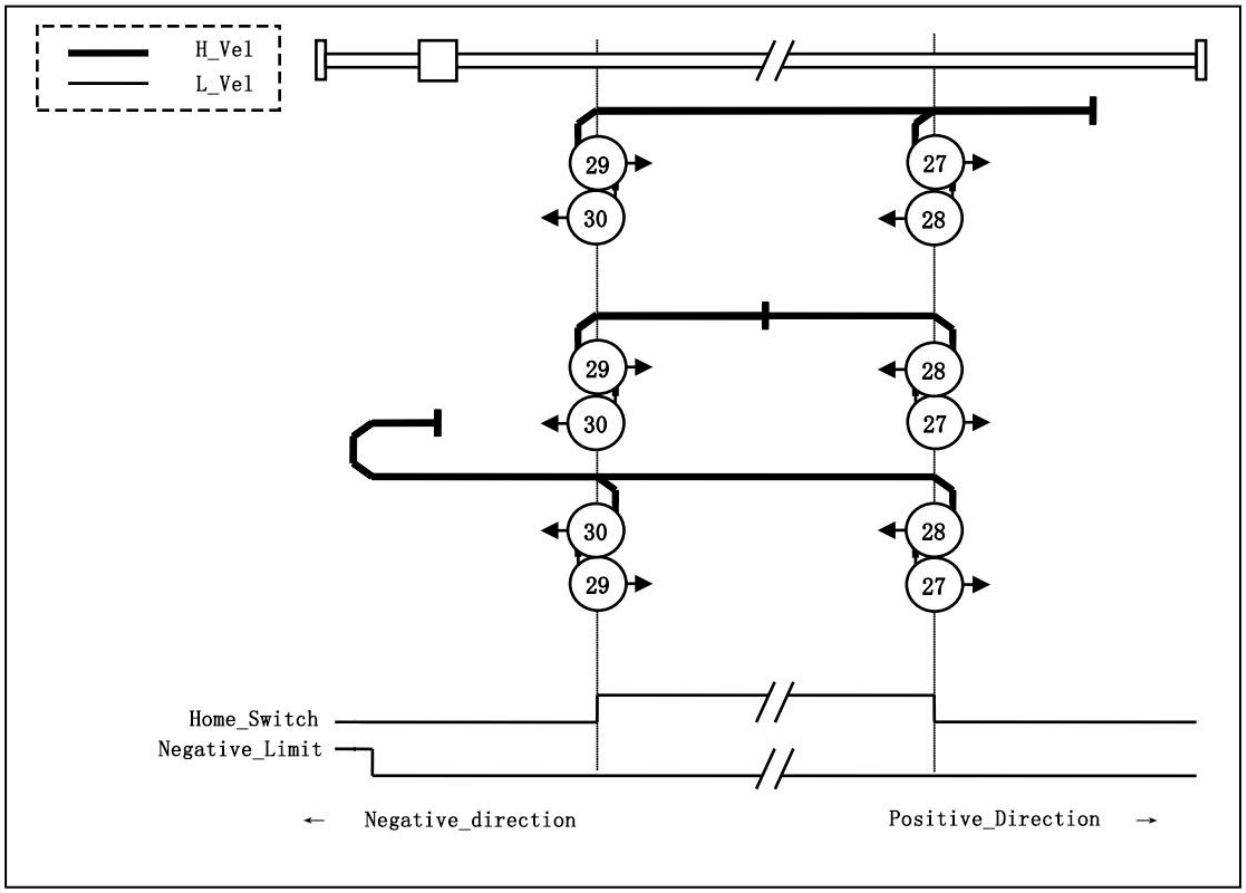
1、朝负方向寻找原点限位，如果搜索过程中触发负限位，则反方向寻找原点限位，若反方向寻找时触发正限位，则回零失败

2、触发原点限位，调整至原点限位正方向边缘外侧

3、开始朝正方向偏移指定距离

4、偏移结束，回零完成

原点限位正方向边缘外侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间。



回零模式 28

回零方式28：以原点限位的正方向端为参考点，往负方向偏移位置为客户设定原点。

动作流程：

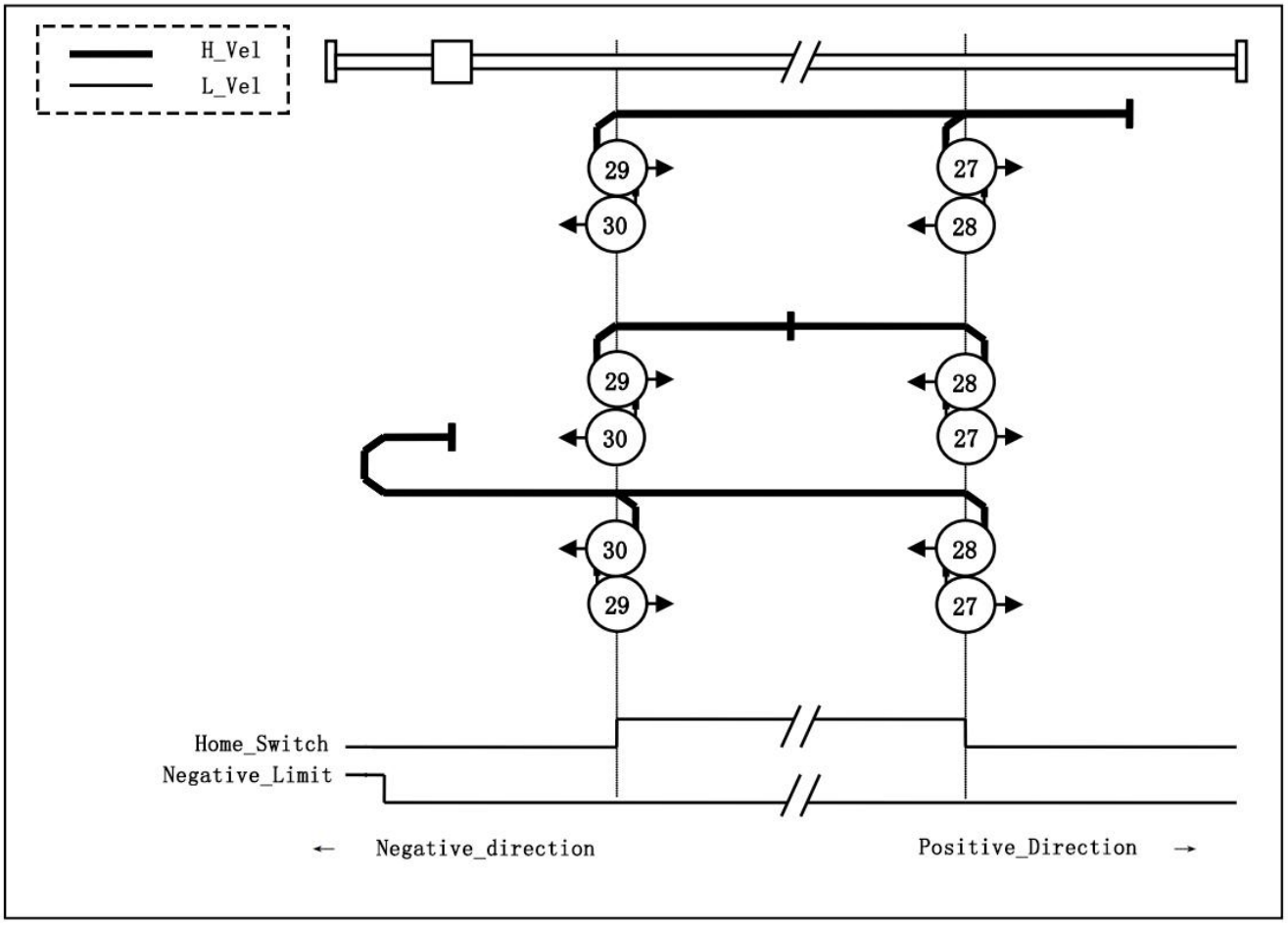
1、朝负方向寻找原点限位，如果搜索过程中触发负限位，则反方向寻找原点限位，若反方向寻找时触发正限位，则回零失败

2、触发原点限位，调整至原点限位正方向边缘内侧

3、开始朝负方向偏移指定距离

4、偏移结束，回零完成

原点限位正方向边缘内侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 29

回零方式29：以原点限位的负方向端为参考点，往正方向偏移位置为客户设定原点。

动作流程：

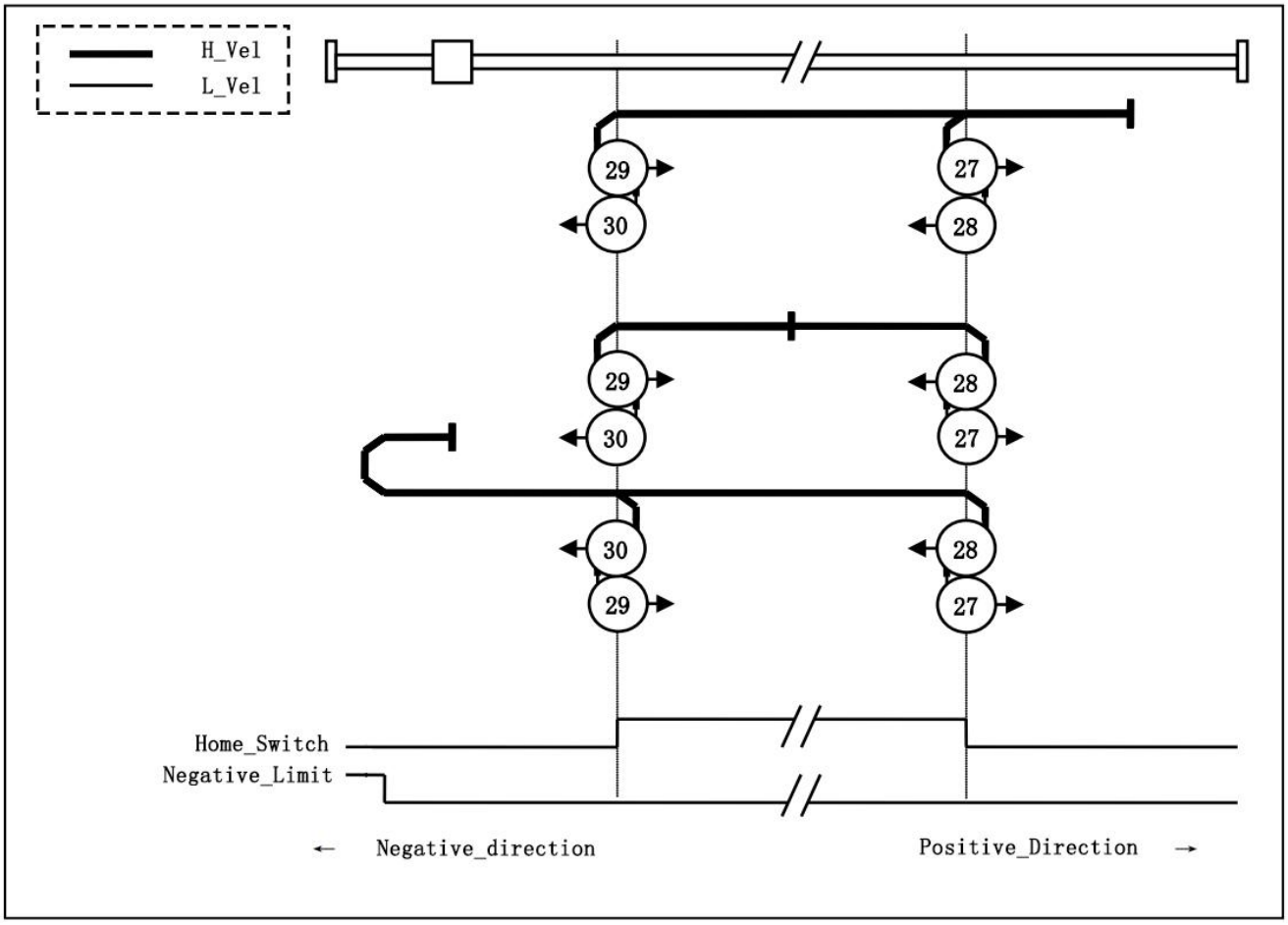
1、朝负方向寻找原点限位，如果搜索过程中触发负限位，则反方向寻找原点限位，若反方向寻找时触发正限位，则回零失败

2、触发原点限位，调整至原点限位负方向边缘内侧

3、开始朝正方向偏移指定距离

4、偏移结束，回零完成

原点限位负方向边缘内侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间后回到触发瞬间。



回零模式 30

回零方式30：以原点限位的负方向端为参考点，往负方向偏移位置为客户设定原点。

动作流程：

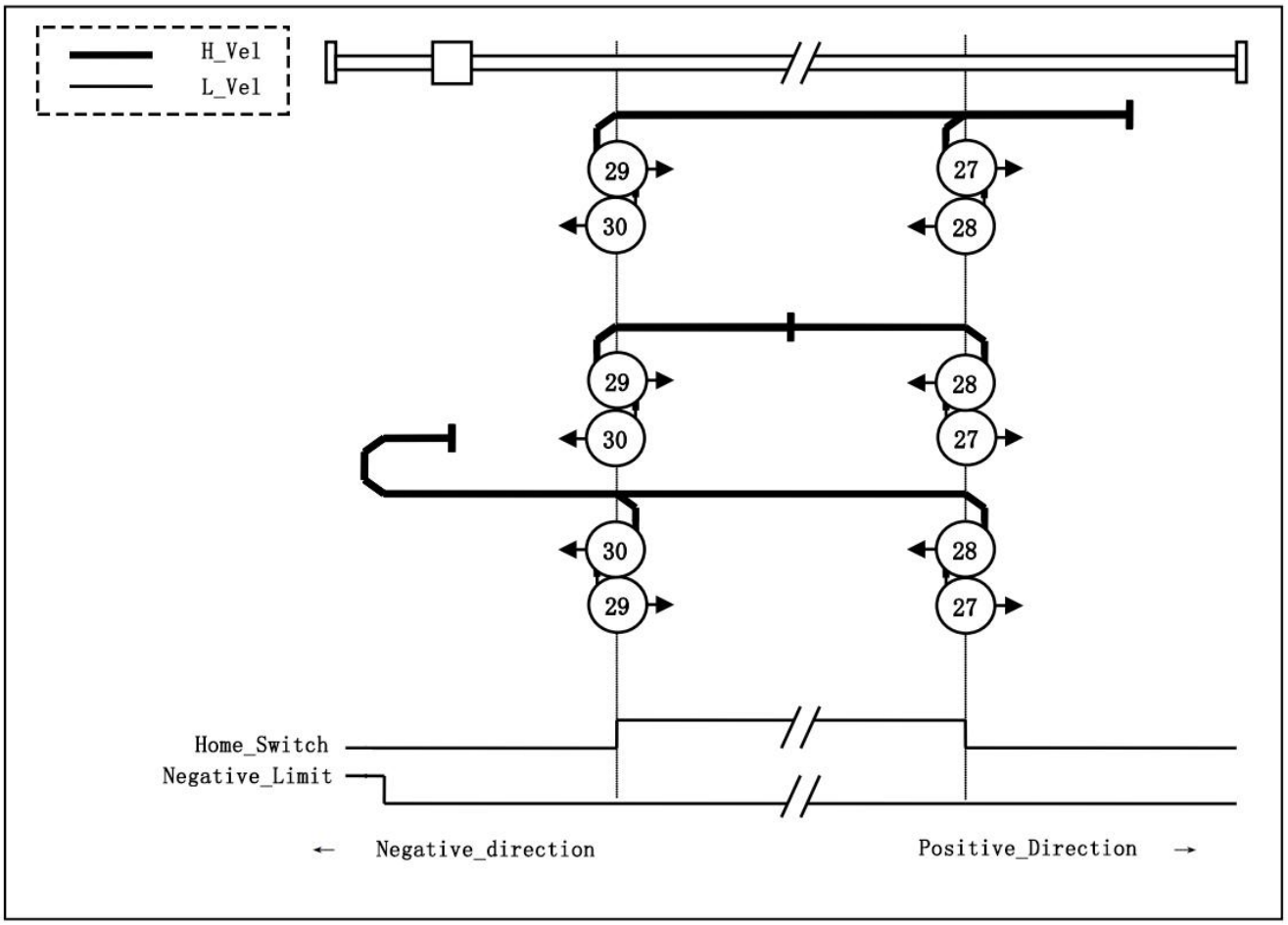
1、朝负方向寻找原点限位，如果搜索过程中触发负限位，则反方向寻找原点限位，若反方向寻找时触发正限位，则回零失败

2、触发原点限位，调整至原点限位负方向边缘外侧

3、开始朝负方向偏移指定距离

4、偏移结束，回零完成

原点限位负方向边缘外侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间。



回零模式 33

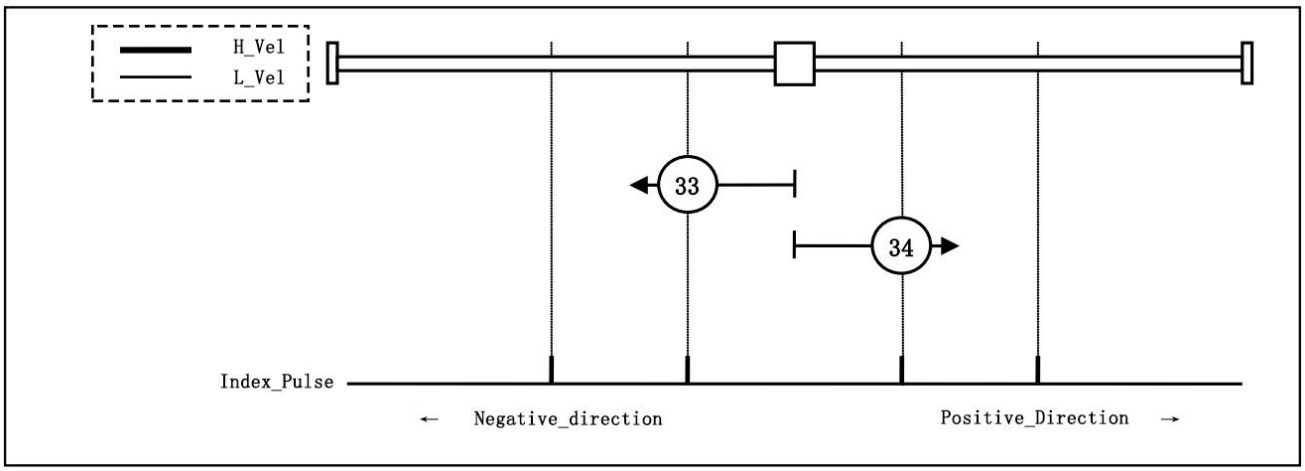
回零方式33：以负方向第一个 Z 信号为零点，往负方向偏移为客户设定原点。

动作流程：

1、朝负方向寻找Z信号

2、触发Z信号，开始朝负方向偏移指定距离

3、偏移完成，回零结束



回零模式 34

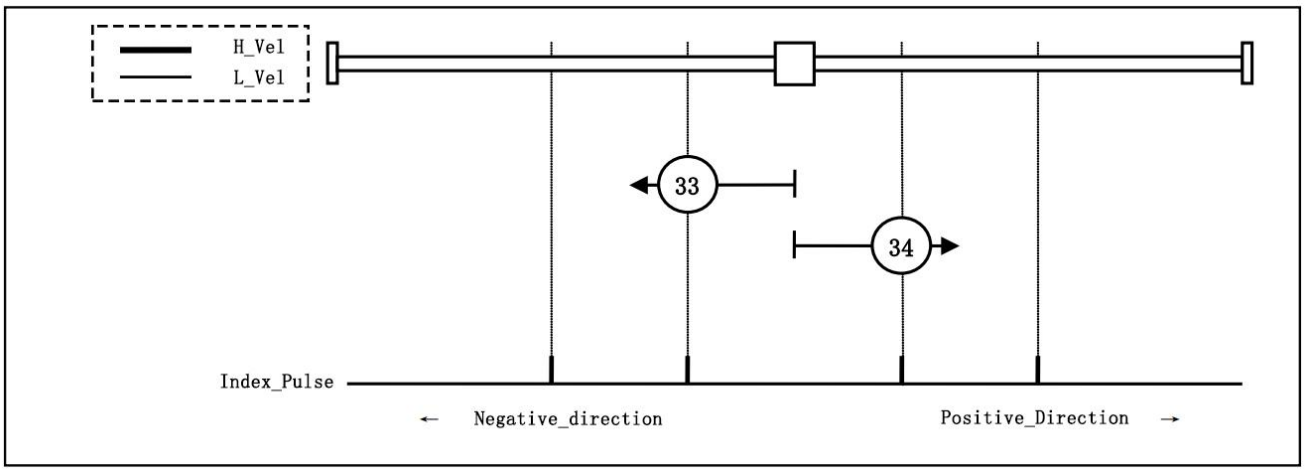
回零方式34：以正方向第一个 Z 信号为零点，往正方向偏移为客户设定原点。

动作流程：

1、朝正方向寻找Z信号

2、触发Z信号，开始朝正方向偏移指定距离

3、偏移完成，回零结束



回零模式 60

回零方式60：以原点限位的正方向端为客户设定原点。

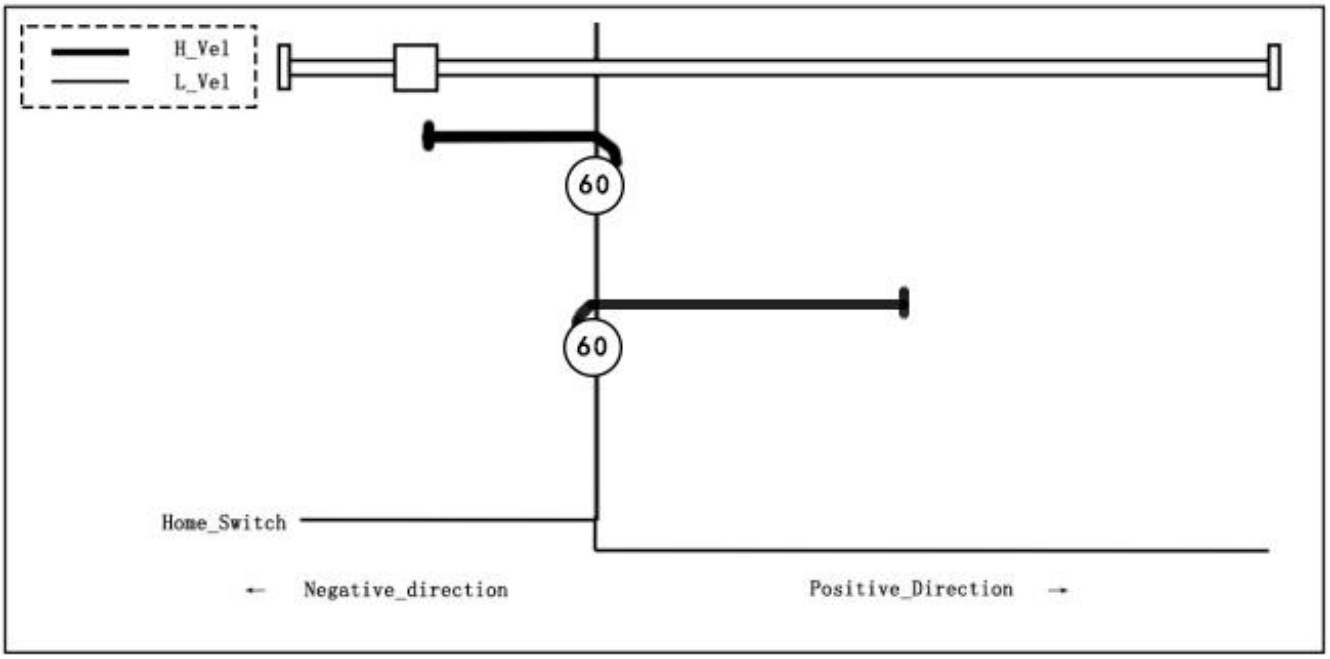
动作流程：

1、朝负方向寻找原点限位

2、触发原点限位，调整至原点限位正方向边缘外侧

3、调整结束，回零完成

原点限位正方向边缘外侧：原点限位信号触发时朝正方向移动直至原点限位信号消失瞬间。



回零模式 61

回零方式61：以原点限位的负方向端为客户设定原点。

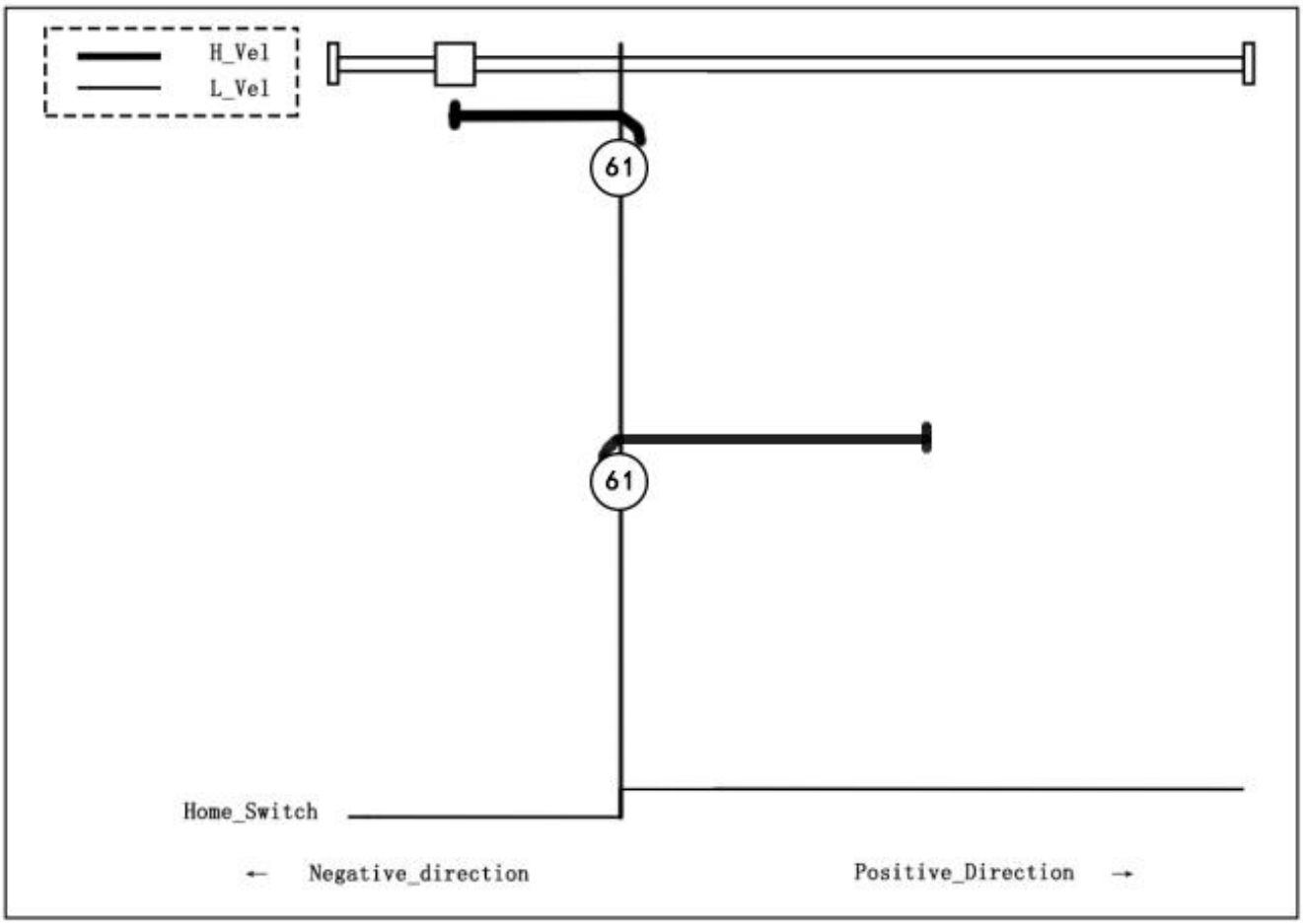
动作流程：

1、朝正方向寻找原点限位

2、触发原点限位，调整至原点限位负方向边缘外侧

3、调整结束，回零完成

原点限位负方向边缘外侧：原点限位信号触发时朝负方向移动直至原点限位信号消失瞬间。



示例：

1. 单轴回零

```
short rtn;
rtn = MCF_Search_Home_Stop_Time_Net(0, 10, 0);      //设置回零触发缓停时间 10ms;
rtn = MCF_Search_Home_Set_Net(0, 21, 0, 0, 0, 10000, 1000, 100, 0, 0);    //设置回零参数
rtn = MCF_Search_Home_Start_Net(0, 0);      //开始回零
```

2. 多轴同时回零

```
short rtn;
rtn = MCF_Search_Home_Stop_Time_Net(0, 100, 0);    //1 轴设置回零触发缓停时间 100ms;
rtn = MCF_Search_Home_Stop_Time_Net(1, 50, 0);      //2 轴设置回零触发缓停时间 50ms;
```

```
rtn = MCF_Search_Home_Stop_Time_Net(2, 50, 0);      //3 轴设置回零触发缓停时间 50ms;
rtn = MCF_Search_Home_Set_Net(0, 21, 0, 0, 0, 50000, 10000, 100, 0, 0);   //设置 1 轴回零参数  模式 19
rtn = MCF_Search_Home_Set_Net(1, 19, 0, 0, 0, 30000, 10000, 100, 0, 0);   //设置 2 轴回零参数
rtn = MCF_Search_Home_Set_Net(2, 20, 0, 0, 0, 10000, 3000, 100, 0, 0);    //设置 3 轴回零参数
rtn = MCF_Search_Home_Start_Net(0, 0);
rtn = MCF_Search_Home_Start_Net(1, 0);
rtn = MCF_Search_Home_Start_Net(2, 0);      //开始回零
```

6.2 回零状态检测

函数原型	short **MCF _Search_Home_Get _State_Net** (unsigned short Axis , unsigned short *Home_State, unsigned short StationNumber);
使用说明	回零状态查询
参数说明	Axis：设置轴号 *Home_State : 返回值定义： 0： 回零成功 31： 回零错误 32：正在回零点 StationNumber： 站号

在回零过程中，可通过上诉函数进行查询回零当前状态

示例：

1. 检测回零状态

```
short rtn = 0;
unsigned short Home_State = 0;
rtn = MCF_Search_Home_Get_State_Net(Axis_1, &Home_State, 0);
if(Home_State == 0) {/*回零成功*/}
else if(Home_State == 31) {/*回零点错误*/}
else if(Home_State == 32) {/*正在回原点*/}
```

第 7 章 单轴点位、多轴同动运动

7.1 JOG模式指令

在 JOG 控制模式下,控制轴以设定的加速度，加速到目标速度，在没有新的速度改变指令一直保持目标速度。设置速度为 0 或者外部条件触发停止，结束速度控制,。

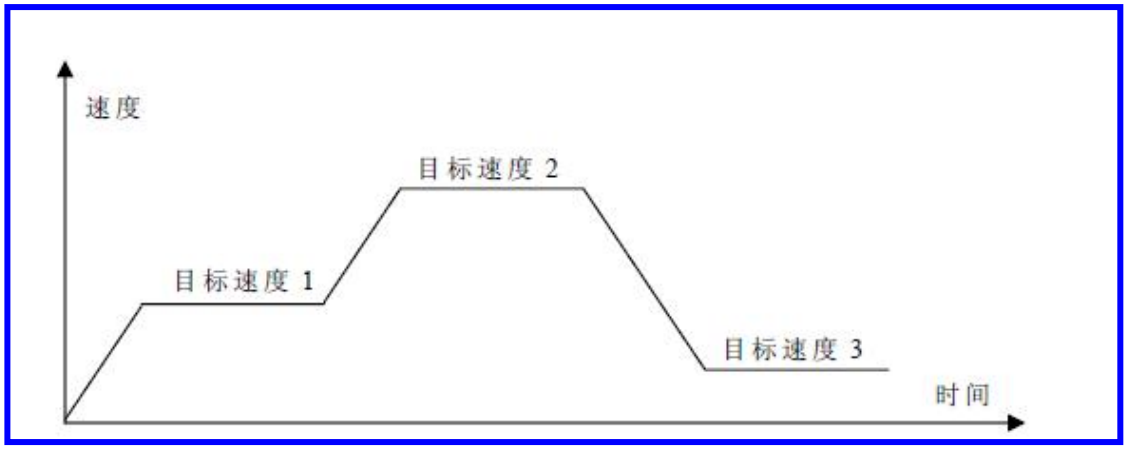


图 7-3

函数原型	short **MCF_JOG_Net** (unsigned short Axis, double dMaxV, double dMaxA, unsigned short StationNumber = 0);
使用说明	速度控制函数
参数说明	Axis：指定轴号 取值：0-Axis_MAX dMaxV：设置运动速度，单位：脉冲/s 取值：0 表示停止速度环控制。 dMaxA：设置运动加速度，单位：脉冲/s*s， 取值： dMaxA>0 StationNumber: 站号设置，默认不填为 0

示例：

1. 轴 1 执行正向连续运动（JOG 模式）
   rtn = MCF_JOG_Net(Axis_1, 10000, 100000);
2. 轴 1 执行负向连续运动（JOG 模式）
   rtn = MCF_JOG_Net(Axis_1, -10000, 10000);

7.2 运动过程中更改位置

在开始单轴T型运动后并且运动未完成下可以调用指令进行修改运动目标位置。注意：对非单轴T型的运动无效，指令作用过程如下：

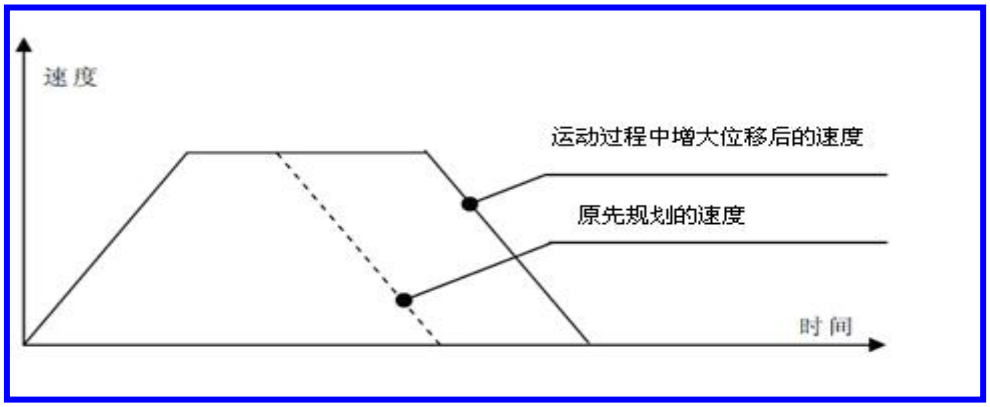


图 7-2-2-1

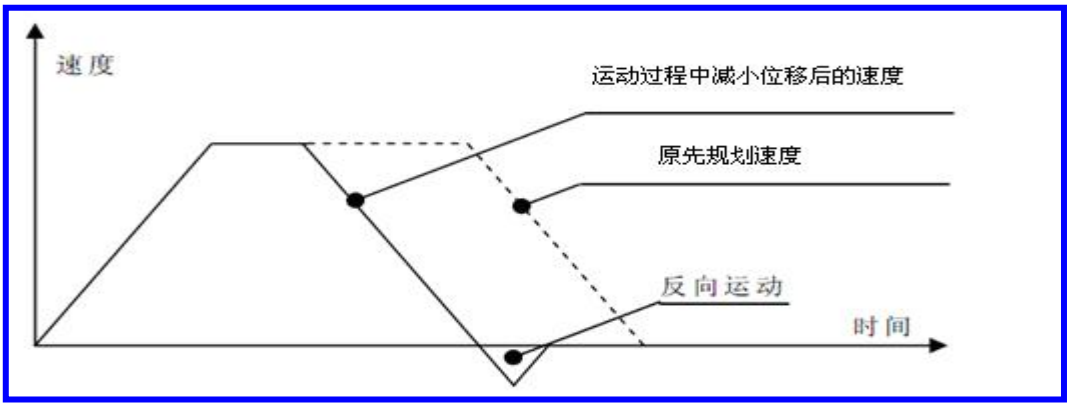


图 7-2-2-2

函数原型	short **MCF_Uniaxial_dDist_Change_Net** (unsigned short Axis, double dDist, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	单轴运动位置改变函数
参数说明	Axis：指定轴号 取值：0-Axis_MAX dDist：设置改变的目标位置，单位：脉冲 Position_Mode：坐标模式， 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为 0

示例：

1. 轴 1 运动过程中改变终点目标位置

```
. . . //运动过程
rtn = MCF_Uniaxial_dDist_Change_Net        (0,10000,0); //改变当前终点目标位置为：10000
```

7.3 运动过程中更改速度

在开始单轴 T 型运动后并且运动未完成下可以调用指令进行修改当前速度和加速度。

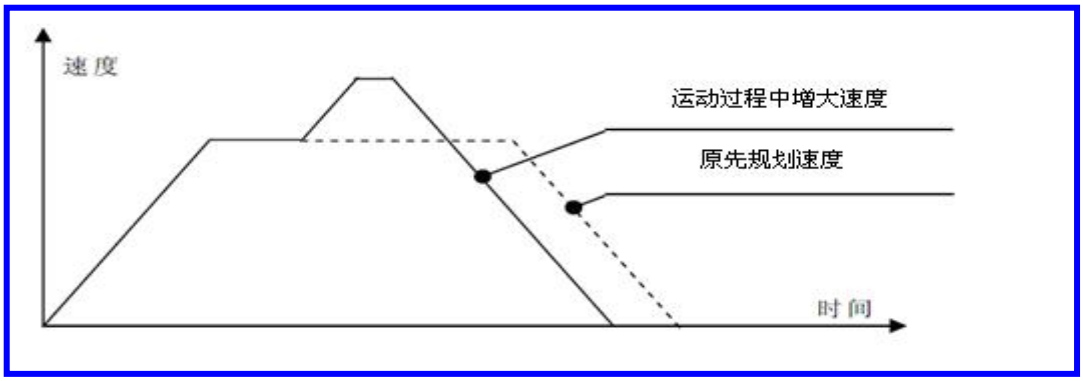


图 7-2-2-3

函数原型	short **MCF_Uniaxial_dMaxV_Change_Net** (unsigned short Axis, double dMaxV, unsigned short StationNumber = 0);
使用说明	单轴运动速度改变函数
参数说明	Axis：指定轴号 取值：0-Axis_MAX dMaxV：设置改变的目标速度，单位：脉冲/s StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Uniaxial_dMaxA_Change_Net** (unsigned short Axis, double dMaxA, unsigned short StationNumber = 0);
使用说明	单轴运动加速度改变函数
参数说明	Axis：指定轴号， 取值：0-Axis_MAX dMaxA：设置改变的目标速度，单位：脉冲/s*s StationNumber：站号设置，默认不填为 0

示例：

1. 轴 1 运动过程中改变速度和加速度

```
..... //运动过程
rtn = MCF_Uniaxial_dMaxA_Change_Net          (0,1000000,0);
//改变当前加速度为：1000000 脉冲/s*s
rtn = MCF_Uniaxial_dMaxV_Change_Net          (0,10000,0);
//改变当前速度为 10000 脉冲/s
```

7.4 速度规划曲线

7.4.1 T 形曲线

用户通过设置**MCF_Set_Axis_Profile_Net**函数中Profile参数来设置运动曲线类型，来进入点位运动曲线功能。并可以设置起始速度和终止速度，下面仅以起始速度和终止速度为0做介绍。

当Profile=1时表示S型曲线，

当Profile=0时表示T型曲线。

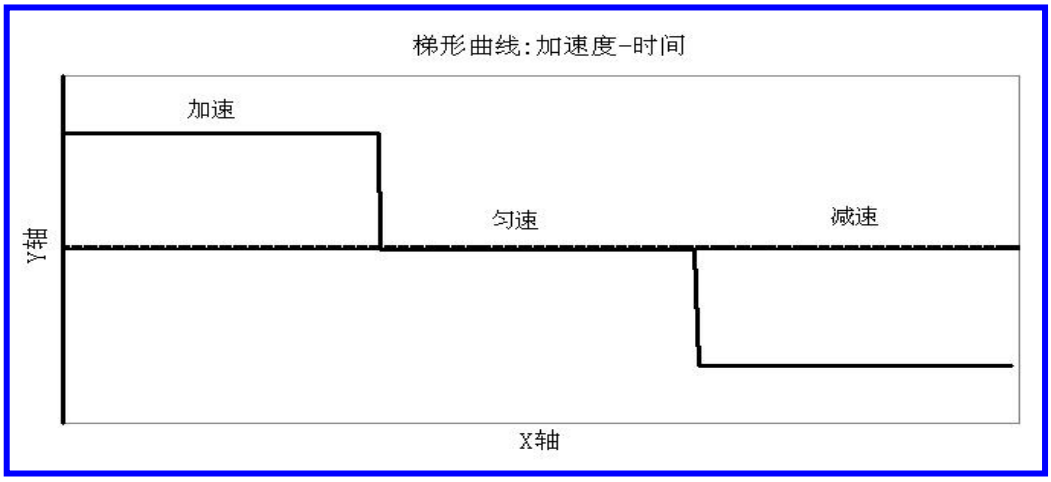


图7-4-1-1

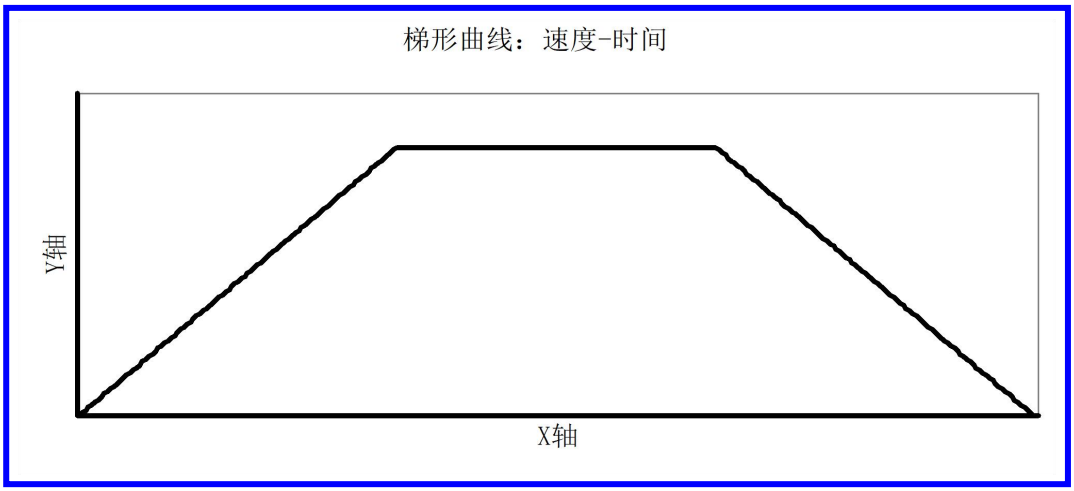


图7-4-1-2

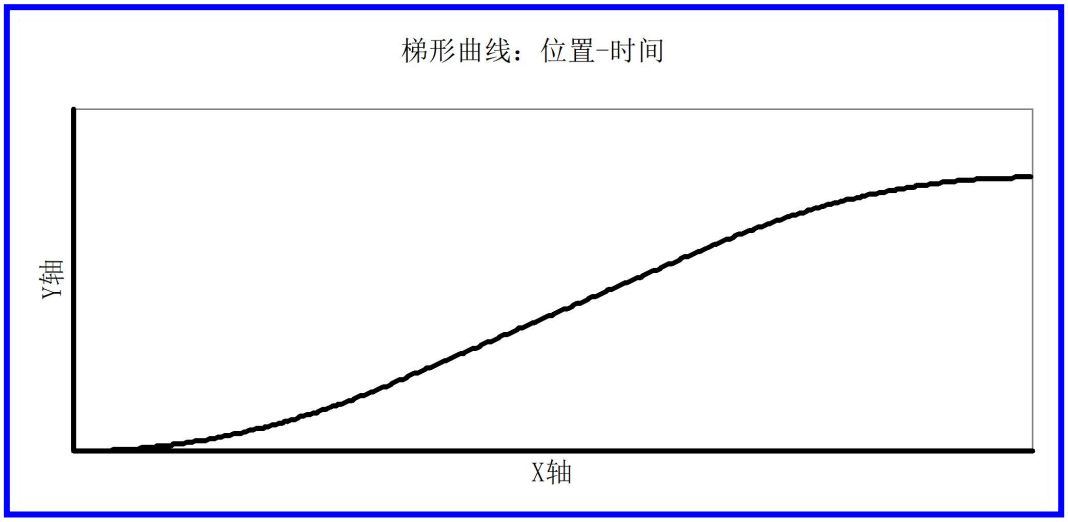


图7-4-1-3

梯形曲线可以划分为3个阶段：

1．加速：速度按照设定的加速度从设定的起始速度加速到目标速度。

2．匀速：保持目标速度，直至到达减速点。

3．减速：速度按照设定的加速度从目标速度减速到终止速度。

梯形曲线可以在运动过程当中修改目标位置和目标速度。如果在运动过程中增大目标位置,运动控制器根据新的目标位置重新计算减速点，到达新的减速点后开始减速。如果在运动过程当中减小目标位置，运动控制器根据新的目标位置重新计算减速点。如果当前位置已经超越新的减速点，运动控制器首先减速到0，然后反向运动到新的目标位置。如果当前位置没有超越新的减速点，运动控制器到达新的减速点后开始减速。

7.4.2 S 形曲线

用户通过设置MCF_Set_Axis_Profile_Net函数中Profile参数来设置运动曲线类型，来进入点位运动曲线功能。并可以设置起始速度和终止速度，下面仅以起始速度和终止速度为0做介绍。

当Profile=1时表示S型曲线，

当Profile=0时表示T型曲线。

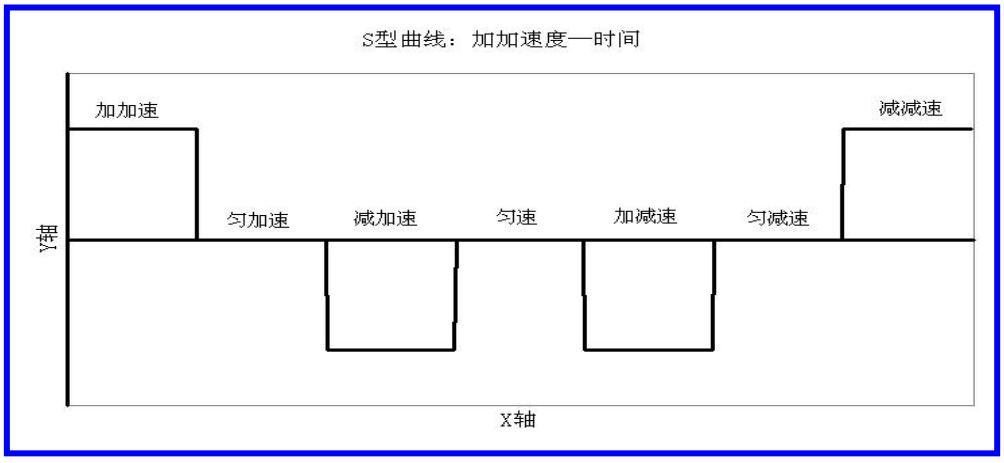


图7-3-1

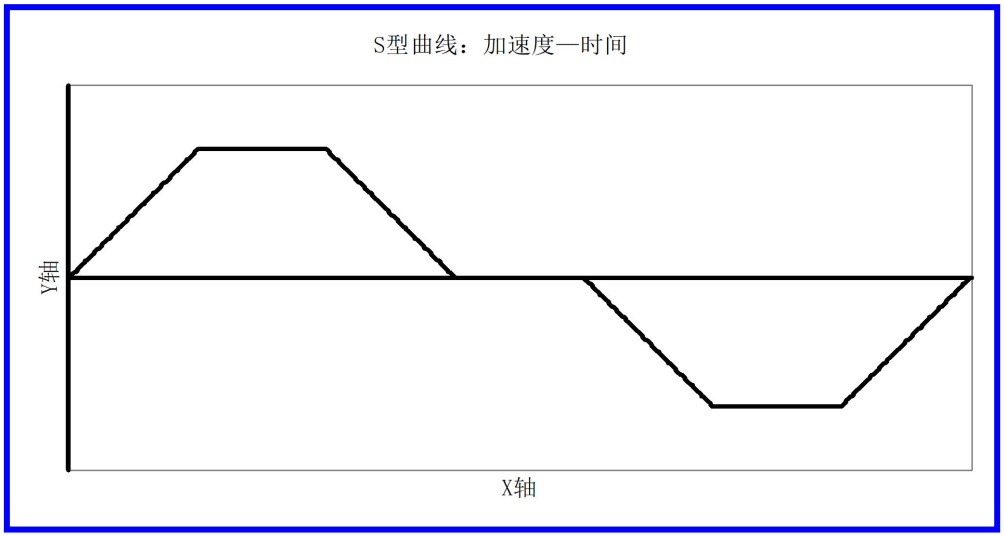


图7-3-2

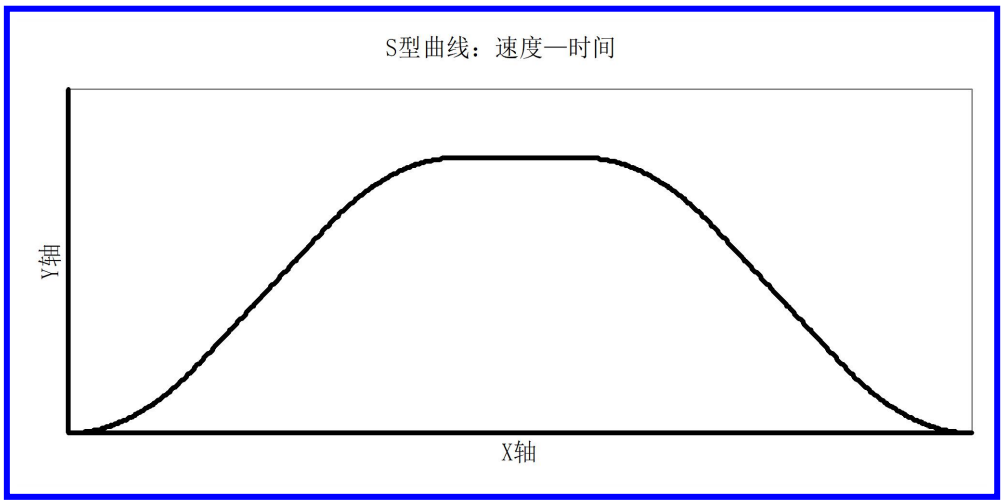


图7-3-3

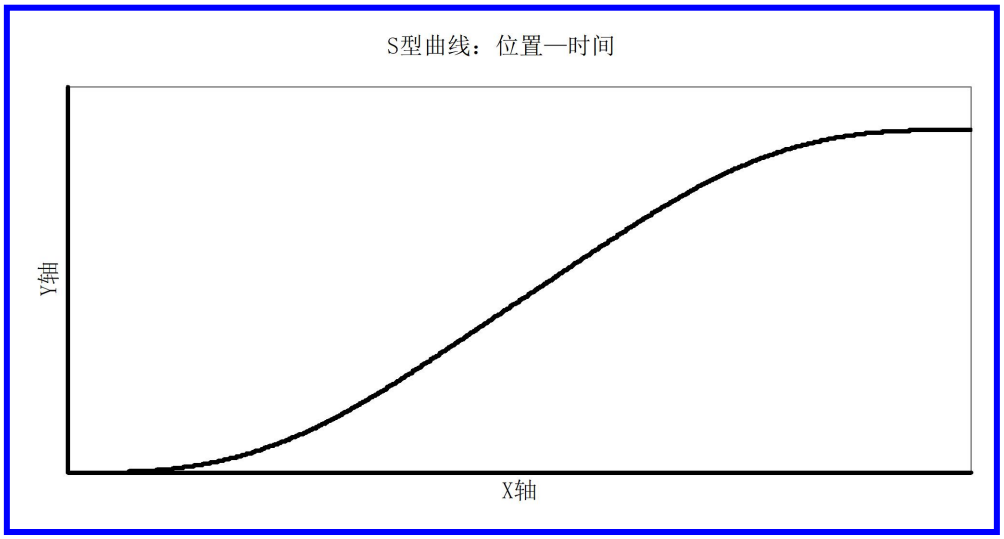


图7-3-4

直线加减速方式在启动和加减速结束是存在加速度突变，产生冲击，因而不适合用于高速运动控制，采用S形加减速，通过对启动阶段即高速阶段的加速度衰减，来保证电机性能的充分发挥和减小启动冲击。

曲线加减速的称法由系统在加减速阶段的速度曲线形状呈S形而得来。正常情况下的S曲线加减速如图，运行过程可分为七段：加加速段、匀加速段、减加速段、匀速段、加减速段、匀减速段和减减速段。

1. 加加速：加速度按照设定的加加速度从0递增到最大加速度，速度按照加速度递增。
2. 匀加速：加速度保持最大加速度不变，速度按照最大加速度递增。
3. 减加速：加速度按照设定的加加速度从最大加速度递减到0，速度按照加速度递增。
4. 匀速：加速度为0，速度保持目标速度不变。
5. 加减速：加速度按照设定的加加速度从0递增到最大加速度，速度按照加速度递减。
6. 匀减速：加速度保持最大加速度不变，速度按照最大加速度递减。
7. 减减速：加速度按照设定的加加速度从最大加速度递减到0，速度按照加速度递减。

示例：

1. 轴 1 轴 0 执行 S 型的点位运动运动

```
rtn = MCF_Set_Axis_Profile_Net (Axis_1,0,5000,50000,500000,Profile_S,0);//设置轴一运动参数
rtn = MCF_Uniaxial_Net (Axis_1,10000, Position_Opposite);//启用轴 0 点动运动
```

7.4.3 曲线规划设置

通过调用曲线设置函数 **MCF_Set_Axis_Profile_Net** 来设置运动曲线参数。

函数原型	short **MCF_Set_Axis_Profile_Net** (unsigned short Axis, double dV_ini, double dMaxV, double dMaxA, double dJerk, double dV_end, unsigned short Profile, unsigned short StationNumber = 0);
使用说明	单轴曲线函数
参数说明	Axis：指定轴号 取值：0-Axis_MAX dV_ini： 设置单轴点动启动速度，单位：pulse/s 取值：dMaxV >dV_ini>=0 dMaxV：设置单轴点动目标速度，单位：pulse/s 取值：dMaxV >0 dMaxA：设置单轴点动加速度，单位：pulse/s*s 取值：dMaxA >0 dJerk：设置单轴点动加加速度，单位：pulse/s*s*s 取值：dJerk >0 dV_end：设置单轴点动终止速度，单位：pulse/s 取值：dMaxV > dV_end >=0

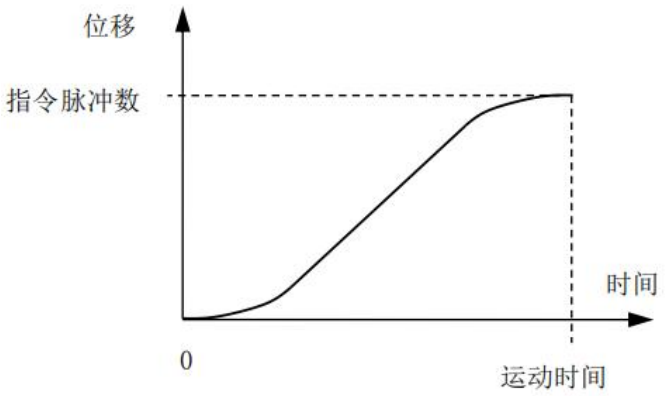
	Profile：设置单轴点动曲线类型 取值： Profile_T 0 Profile_S 1 StationNumber: 站号设置，默认不填为 0

函数原型	short **MCF_Get_Axis_Profile_Net** (unsigned short Axis, double *dV_ini, double *dMaxV, double *dMaxA, double *dJerk, double *dV_end, unsigned short *Profile, unsigned short StationNumber = 0);
使用说明	读取单轴曲线函数
参数说明	Axis：指定轴号 取值：0-Axis_MAX *dV_ini： 读取单轴点动启动速度，单位：pulse/s *dMaxV：读取单轴点动目标速度，单位：pulse/s *dMaxA：读取单轴点动加速度，单位：pulse/s*s *dJerk：读取单轴点动加加速度，单位：pulse/s*s*s *dV_end：读取单轴点动终止速度，单位：pulse/s *Profile：读取单轴点动曲线类型 StationNumber: 站号设置，默认不填为 0

7.5 点位运动

点位运动是指：运动控制卡控制运动平台从当前位置开始以设定的速度运动到指定位置后准确地停止。

点位运动只关注终点坐标，对运动轨迹的精度没有要求。点位运动的运动距离由脉冲数决定，运动速度由脉冲频率决定。



7.5.1 单轴运动

通过调用单轴运动函数 **MCF_Uniaxial_Net** 实现点位运动功能

函数原型	short **MCF_Uniaxial_Net** (unsigned short Axis, long dDist, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	单轴运动函数
参数说明	Axis：指定轴号 取值：0-Axis_MAX dDist：设置点动目标位置，单位：脉冲 Position_Mode：设置坐标模式 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为0

示例：

1. 轴 0 执行点位运动

```
short rtn = 0;
rtn= MCF_Set_Axis_Profile_Net (Axis_1,0,5000,50000,500000,0,0);
//设置轴 0 运动参数
//dV_ini=dV_end=0
// dMaxV=5000 脉冲/s，
//加速度 50000 脉冲/s*s，
```

```
//加加速度 500000 脉冲/s*s*s,
//停止曲线：T 型曲线
rtn = MCF_Uniaxial_Net (Axis_1,10000, Position_Opposite);
//启用轴 0 点动运动
//目标位置：10000 脉冲
//运动坐标为相对坐标
```

7.5.2 多轴连动

多轴同时做点位运动，称之为多轴联动。由于软件处理速度远比机械系统响应速度快，虽然多条运动指令连续发出，需几个微秒，可对机械系统而已，可认为是同时启动。

示例：

1. 轴 1 轴 2 执行联动

```
rtn = MCF_Set_Axis_Profile_Net (Axis_1,0,5000,50000,500000,0,0);
//设置轴 1 运动参数
return = MCF_Uniaxial_Net (Axis_1,10000, Position_Opposite);
//启用轴 1 点动运动
return = MCF_Set_Axis_Profile_Net (Axis_2,0,5000,50000,500000,0,0);
//设置轴 2 运动参数
return = MCF_Uniaxial_Net (Axis_2,10000, Position_Opposite);
//启用轴 2 点动运动
```

7.6 轴运动停止

7.6.1 S 型,T 型平滑停止

用户首先通过设置 MCF_Set_Axis_Stop_Profile_Net 和 MCF_Set_Coordinate_Stop_Profile_Net 函数，为平滑停止指定停止加速度，停止曲线模式为 T 型或者为 S 型。

注意：因为平滑停止的停止曲线模式只能为 T 型或者 S 型，其 Profile 参数的取值只能是 Profile_T 或者 Profile_S。且 Profile_T 和 Profile_S 在宏定义中已经被声明为 0 和 1。

使用 MCF_Axis_Stop_Net 和 MCF_Coordinate_Stop_Net 指令时，将 Mode 设置为 1 能够让控制轴平滑停止。调用平滑停止指令时，运动控制器将目标速度设置为 0，并根据用户所设定的加速度沿着正在运动的方向平滑减速到 0。

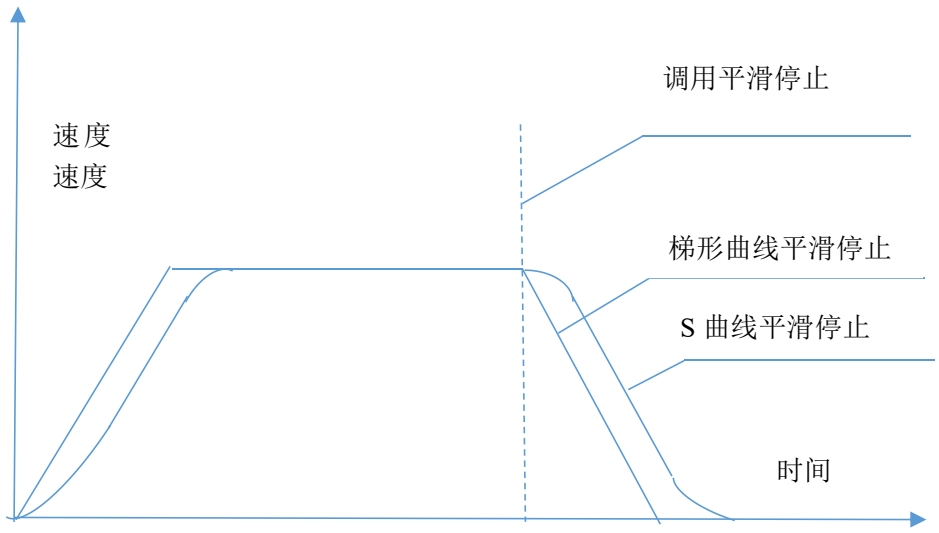


图 7-5 平滑停止速度曲线

7.6.2 单轴停止曲线设置

函数原型	short **MCF_Set_Axis_Stop_Profile_Net** (unsigned short Axis, double dMaxA, double dJerk, unsigned short Profile, unsigned short StationNumber = 0);
使用说明	设置单轴平滑停止参数
参数说明	Axis： 指定轴号 取值：0-Axis_MAX dMaxA： 设置平滑停止加速度 单位：pulse/s*s 取值：dMaxA >0 dJerk：设置平滑停止加加速度 单位：pulse/s*s*s 取值：dJerk >0 Profile：停止曲线模式 取值： Profile_T 0 Profile_S 1 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Axis_Stop_Profile_Net**

	(unsigned short Axis, double *dMaxA, double *dJerk, unsigned short *Profile, unsigned short StationNumber = 0);
使用说明	读取单轴平滑停止参数
参数说明	Axis： 指定轴号 取值：0-Axis_MAX *dMaxA： 返回平滑停止加速度。 *dJerk：返回平滑停止加加速度 *Profile：返回停止曲线模式 StationNumber：站号设置，默认不填为 0

7.7 轴停止

7.7.1 轴停止函数

函数原型	short **MCF_Axis_Stop_Net** (unsigned short Axis, unsigned short Axis_Stop_Mode, unsigned short StationNumber = 0);
使用说明	单轴停止运动
参数说明	Axis：指定轴号 取值：0-Axis_MAX Axis_Stop_Mode：停止模式 取值： Axis_Stop_IMD 0 //紧急停止 Axis_Stop_DEC 1 //减速停止 StationNumber：站号设置，默认不填为 0

示例：

1. 设置指令平滑停止

```
rtn = MCF_Set_Axis_Stop_Profile_Net (0,100000,1000000,Profile_S,0);
//设置轴 0 停止曲线参数
//停止加速度：100000
//停止加加速度：1000000
//停止曲线：S 型
rtn = MCF_Axis_Stop_Net (0,Axis_Stop_DEC,0);
//指令控制轴 0 平滑停止
```

7.7.2 紧急停止

使用MCF_Axis_Stop_Net指令时，将Mode设置为0能够让控制轴紧急停止。调用紧急停止指令时，无论控制轴处于何种加减速模式，都将以极短的时间减速到0。

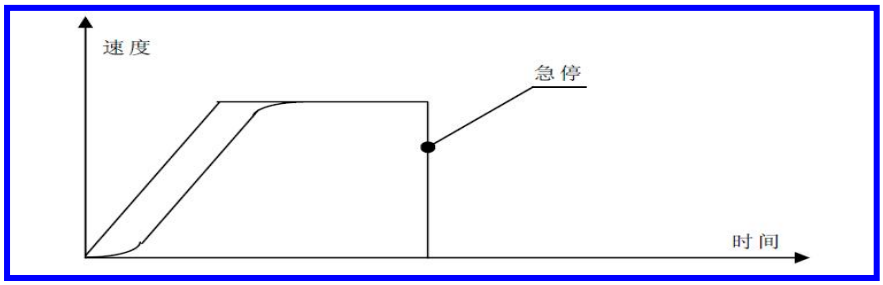


图 7-6 急停速度曲线

示例：

1. 设置指令紧急停止

```
rtn = MCF_Axis_Stop_Net                    (0,Axis_Stop_IMD,0);
```

第 8 章 插补运动控制函数

8.1 坐标系曲线设置

插补（Interpolation），即机床数控系统依照一定方法确定刀具运动轨迹的过程。也可以说，已知曲线上的某些数据，按照某种算法计算已知点之间的中间点的方法，也称为“数据点的密化”；数控装置根据输入的零件程序的信息，将程序段所描述的曲线的起点、终点之间的空间进行数据密化，从而形成要求的轮廓轨迹，这种“数据密化”机能就称为“插补”。

轴控模块的插补功能具有以下特点：

1. 提供多轴直线插补和圆弧插补；
2. 可以实现同时有两个坐标系进行插补运动；
3. 可以将插补功能写入缓冲区；
4. 插补功能在缓冲区中有速度前瞻功能。

函数原型	short **MCF_Set_Coordinate_Profile_Net** (unsigned short Coordinate, double dV_ini, double dMaxV, double dMaxA, double dJerk, double dV_end, unsigned short Profile, unsigned short StationNumber = 0);
使用说明	设置坐标系运动参数
参数说明	Coordinate：指定坐标系 取值：0,1 dV_ini： 设置单轴点动启动速度，单位：pulse/s 取值：dMaxV >dV_ini>=0 dMaxV：设置单轴点动目标速度，单位：pulse/s 取值：dMaxV >0 dMaxA：设置单轴点动加速度，单位：pulse/s*s 取值：dMaxA >0 dJerk：设置单轴点动加加速度，单位：pulse/s*s*s 取值：dJerk >0 dV_end：设置单轴点动终止速度，单位：pulse/s 取值：dMaxV > dV_end >=0

	Profile：设置单轴点动曲线类型 取值： Profile_T 0 //T 型曲线 Profile_S 1 //S 型曲线 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Coordinate_Profile_Net** (unsigned short Coordinate, double *dV_ini, double *dMaxV, double *dMaxA, double *dJerk, double *dV_end, unsigned short *Profile, unsigned short StationNumber = 0);
使用说明	读取坐标系运动参数
参数说明	Coordinate：指定坐标系 取值：0,1 *dV_ini： 读取单轴点动启动速度， 单位：pulse/s *dMaxV：读取单轴点动目标速度， 单位：pulse/s *dMaxA：读取单轴点动加速度， 单位：pulse/s*s *dJerk：读取单轴点动加加速度， 单位：pulse/s*s*s *dV_end：读取单轴点动终止速度， 单位：pulse/s *Profile：读取单轴点动曲线类型 StationNumber：站号设置，默认不填为 0

8.2 圆弧半径插补运动

运动控制卡的任意两轴之间可以进行圆弧插补，运动的方向分为顺时针（CW）和逆时针（CCW），可以根据圆心坐标或半径进行插补。

用户可以设置插补曲线为梯型或者 S 型，可以设置起始速度，运动速度，加速时间，终点速度。

函数原型	short **MCF_Arc2_Radius_Net** (unsigned short Coordinate, unsigned short *Axis_List, long *dDist_List, long Arc_Radius,

	unsigned short Direction, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	圆半径插补运动函数
参数说明	Coordinate：指定坐标系 取值：0,1 *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表 单位：pulse Arc_Radius：设置圆弧半径 单位：pulse Direction：设置圆弧运动方向 取值： Clock_Wise 0 //顺弧 Counter_Clock_Wise 1 //逆弧 Position_Mode：设置坐标模式 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为 0

示例：

1. 轴 0 轴 1 做圆弧半径 T 型插补运动

假设丝杆导程为10mm,电机每转脉冲数为10000pulse/r（四倍频后）， 则脉冲当量为1um/pulse, 1m=1000000pulse。

```
unsigned short   Coordinate = 0;                    //分配到坐标系里面
unsigned short   Axis_List[2] = {0,1},              //依次填入插补轴
double           dDist_List[2]={10000,10000},       //依次填入位置
MCF_Set_Coordinate_Profile_Net(Coordinate ,0,1000000,5000000,10000000,0, Profile_T);
//dV_ini=dV_end=0。
//dMaxV=1m/s=1000000pulse。
//dMaxA=5m/s*s=5000000pulse/s*s。
//dJerk=10m/s*s*s=10000000pulse/s*s*s。
//设置为 T 曲线。
MCF_Arc2_Radius_Net(Coordinate,&Axis_List,&dDist_List,500000 , Clock_Wise, Position_Opposite);
//开始 1 轴,2 轴圆弧插补运动。
//dDist_List[0]=0.1m=100000pulse。
//dDist_List[1]=0.2m=200000pulse。
// Arc_r =500000pulse。
//运动方向为顺时针
//运动坐标为相对坐标。
```

8.3 圆弧圆心插补运动

函数原型	short **MCF_Arc2_Centre_Net** (unsigned short Coordinate, unsigned short *Axis_List, long *dDist_List, long *Center_List, unsigned short Direction, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	圆圆心插补运动函数
参数说明	Coordinate：指定坐标系 取值：0,1 *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表 单位：pulse *Center_List：设置圆弧中心位置 单位：pulse Direction：设置圆弧运动方向 取值： Clock_Wise 0 //顺弧 Counter_Clock_Wise 1 //逆弧 Position_Mode：设置坐标模式 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为 0

8.4 直线插补模式

直线插补（Line Interpolation）这是数控系统上常用的一种插补方式，在此方式中，两点间的插补沿着直线的点群来逼近，沿此直线控制刀具的运动。所谓直线插补就是只能用于实际轮廓是直线的插补方式（如果不是直线，也可以用逼近的方式把曲线用一段线段去逼近，从而每一段线段就可以用直线插补了）.首先假设在实际轮廓起始点处沿 x 方向走一小段（一个脉冲当量），发现终点在实际轮廓的下方，则下一条线段沿 y 方向走一小段，此时如果线段终点还在实际轮廓下方，则继续沿 y 方向走一小段，直到在实际轮廓上方以后，再向 x 方向走一小段，依次循环类推.直到到达轮廓终点为止.这样，实际轮廓就由一段段的

折线拼接而成，虽然是折线，但是如果我们每一段走刀线段都非常小（在精度允许范围内），那么此段折线和实际轮廓还是可以近似地看成相同的曲线的。

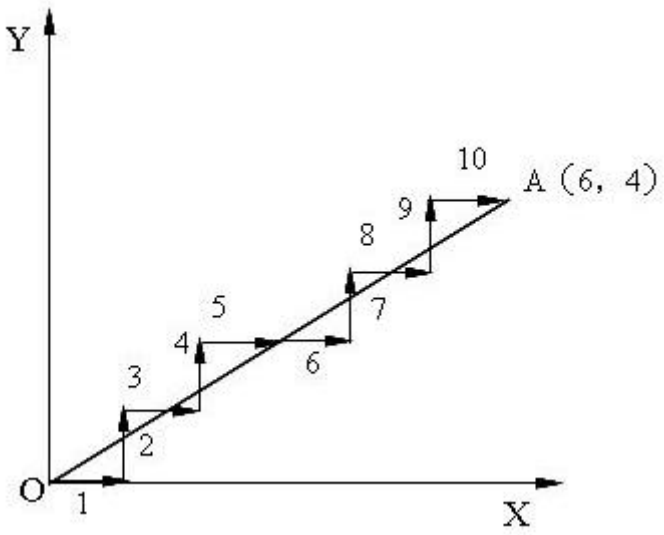


图 8-2 直线插补方式

轴控模块可以通过插补指令进行任意两轴-四轴直线插补。用户可以设置插补曲线为 T 型或者 S 型，可以设置起始速度，运动速度，加速度，加加速度，终点速度。

函数原型	short **MCF_Line2_Net** (unsigned short Coordinate, unsigned short *Axis_List, long *dDist_List, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	启动两轴直线插补
参数说明	Coordinate：指定坐标系 取值：0,1 *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表　单位：pulse Position_Mode：设置坐标模式 取值： Position_Absolute　0　//绝对位置模式 Position_Opposite　1　//相对位置模式 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Line3_Net** (unsigned short Coordinate, unsigned short *Axis_List, long *dDist_List, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	启动三轴直线插补
参数说明	Coordinate：指定坐标系 取值：0,1 *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表 单位：pulse Position_Mode：设置坐标模式 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Line4_Net** (unsigned short Coordinate, unsigned short *Axis_List, long *dDist_List, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	启动四轴直线插补
参数说明	Coordinate：指定坐标系 取值：0,1 *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表 单位：pulse Position_Mode：设置坐标模式 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为 0

示例：

1. 轴 0 轴 1 做 T 型两轴直线插补运动

假设丝杆导程为10mm,电机每转脉冲数为10000pulse/r（四倍频后），则脉冲当量为1um/pulse,1m=1000000pulse。

```
unsigned short    Coordinate = 0;                    //分配到坐标系里面
unsigned short    Axis_List[2] = {0,1},              //依次填入插补轴
Double            dDist_List[2]={10000,10000},       //依次填入位置
MCF_Set_Coordinate_Profile_Net(Coordinate, 0,1000000,5000000,10000000,0, Profile_T);
//dV_ini=dV_end=0。
//dMaxV=1m/s=1000000pulse。
//dMaxA=5m/s*s=5000000pulse/s*s。
//dJerk=10m/s*s*s=10000000pulse/s*s*s。
//设置为 T 曲线。
MCF_Line2_Net(Coordinate,&Axis_List,&dDist_List, Position_Opposite);
//开始 0,1 轴直线插补运动。
//dDist_List[0]=0.1m=100000pulse。
//dDist_List[1]=0.2m=200000pulse。
//运动坐标为相对坐标。
```

8.5 插补运动停止曲线

用户首先通过设置 **MCF_Set_Coordinate_Stop_Profile_Net** 函数，为平滑停止指定停止加速度，停止曲线模式为 T 型或者为 S 型。

注意：因为平滑停止的停止曲线模式只能为 T 型或者 S 型，所以在调用 **MCF_Set_Coordinate_Stop_Profile_Net** 函数时，其 Profile 参数的取值只能是 Profile_T 或者 Profile_S。且 Profile_T 和 Profile_S 在宏定义中已经被声明为 0 和 1。

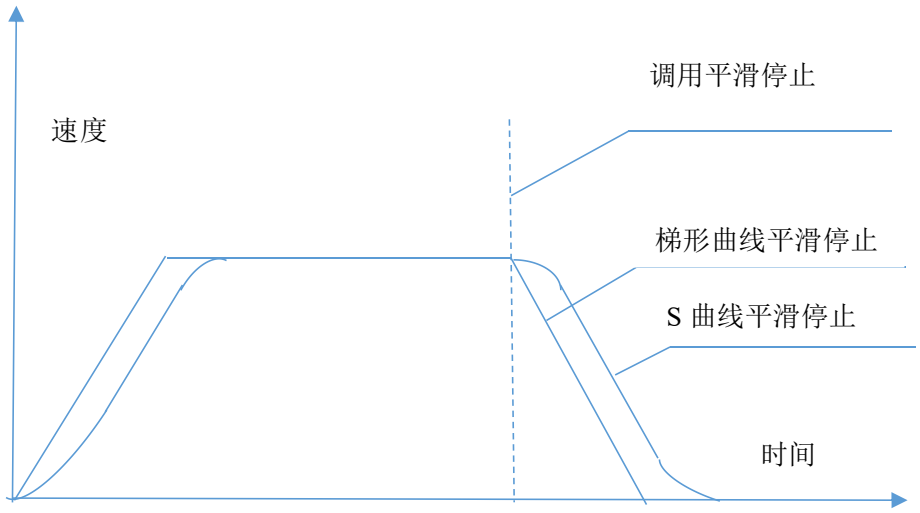


图 8-5 平滑停止速度曲线

函数原型	short **MCF_Set_Coordinate_Stop_Profile_Net**

	(unsigned short Coordinate, double dMaxA, double dJerk, unsigned short Profile, unsigned short StationNumber = 0);
使用说明	设置坐标系平滑停止参数
参数说明	Coordinate：指定坐标系 取值：0,1 dMaxA： 设置平滑停止加速度。 单位：pulse/s*s 取值：dMaxA >0 dJerk：设置平滑停止加加速度 单位：pulse/s*s*s 取值：dJerk >0 Profile：停止曲线模式 取值： Profile_T 0 //T 型曲线 Profile_S 1 //S 型曲线 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Coordinate_Stop_Profile_Net** (unsigned short Coordinate, double *dMaxA, double *dJerk, unsigned short *Profile, unsigned short StationNumber = 0);
使用说明	获取坐标系平滑停止参数
参数说明	Coordinate：指定坐标系 取值：0,1 *dMaxA： 返回平滑停止加速度。 *dJerk：返回平滑停止加加速度 *Profile：返回停止曲线模式 StationNumber：站号设置，默认不填为 0

示例：

1. 设置平滑停止曲线

```
rtn = MCF_Set_Coordinate_Stop_Profile_Net (0,100000, 1000000,Profile_S,0);
//设置停止曲线参数
//坐标系：0
//停止加速度： 100000 脉冲/s*s
```

```
//停止加加速度：1000000 脉冲/s*s*s
//停止曲线：S 型
```

8.6 螺旋线圆半径插补运动

运动控制卡的任意三轴之间可以进行圆柱插补，运动的方向分为顺时针（CW）和逆时针（CCW），使用 MCF_Screw3_Radius_Net 可以根据圆半径进行三轴圆柱插补。
用户可以设置插补曲线为梯型或者 S 型，可以设置起始速度，运动速度，加速时间，终点速度。

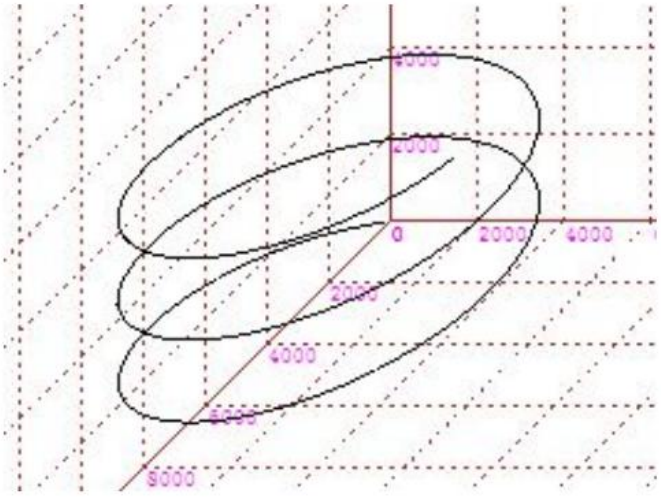


图 8-4 三轴圆柱插补

函数原型	short **MCF_Screw3_Radius_Net** (unsigned short Coordinate, unsigned short *Axis_List, long *dDist_List, long Arc_Radius, unsigned short Direction, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	螺旋线圆半径插补运动函数
参数说明	Coordinate：指定坐标系 取值：0,1 *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表 单位：pulse Arc_Radius：设置圆弧半径 单位：pulse 取值：Arc_Radius>0：优弧, Arc_Radius<0：劣弧 Direction：设置圆弧运动方向

	取值： Clock_Wise 0 //顺弧 Counter_Clock_Wise 1 //逆弧 Position_Mode：设置坐标模式 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为 0

示例：

1. 轴 0 轴 1 轴 2 做圆柱半径 T 型插补运动

假设丝杆导程为10mm,电机每转脉冲数为10000pulse/r（四倍频后），则脉冲当量为1 um/pulse, 1m=1000000pulse。

```
unsigned short   Coordinate = 0;                    //分配到坐标系里面
unsigned short   Axis_List[3] = {0,1,2},            //依次填入插补轴
double           dDist_List[3]={10000,10000,10000},  //依次填入位置
MCF_Set_Coordinate_Profile_Net(Coordinate ,0,1000000,5000000,10000000,0, Profile_T);
//dV_ini=dV_end=0。
//dMaxV=1m/s=1000000pulse。
//dMaxA=5m/s*s=5000000pulse/s*s。
//dJerk=10m/s*s*s=10000000pulse/s*s*s。
//设置为 T 曲线。
MCF_Screw3_Radius_Net(Coordinate,&Axis_List,&dDist_List,500000 , Clock_Wise, Position_Opposite);
//开始 1 轴,2 轴,3 轴圆柱插补运动。
//dDist_List[0]=0.1m=100000pulse。
//dDist_List[1]=0.1m=100000pulse。
//dDist_List[3]=0.1m=100000pulse。
// Arc_r =500000pulse。
//运动方向为顺时针
```

8.7 螺旋线圆圆心插补运动

运动控制卡除了可以使用上述 MCF_Screw3_Radius_Net 来实现根据圆半径进行三轴圆柱插补之外，也可以使用 MCF_Screw3_Centre_Net 可以根据圆心位置进行三轴圆柱插补。

同样，在使用 MCF_Screw3_Centre_Net 进行三轴圆柱插补也可以设置插补曲线为梯型或者 S 型，可以设置起始速度，运动速度，加速时间，终点速度。

函数原型	short **MCF_Screw3_Centre_Net** (unsigned short Coordinate, unsigned short *Axis_List, long *dDist_List, long *Center_List,

	unsigned short Direction, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	螺旋线圆圆心插补运动函数
参数说明	Coordinate：指定坐标系 取值：0,1 *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表 单位：pulse *Center_List：设置圆弧中心位置 单位：pulse Direction：设置圆弧运动方向 取值： Clock_Wise 0 //顺弧 Counter_Clock_Wise 1 //逆弧 Position_Mode：设置坐标模式 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为 0

8.8 坐标系停止

使用MCF_Coordinate_Stop_Net指令时，将Mode设置为0能够让控制轴紧急停止。调用紧急停止指令时，无论控制轴处于何种加减速模式，都将以极短的时间减速到0。

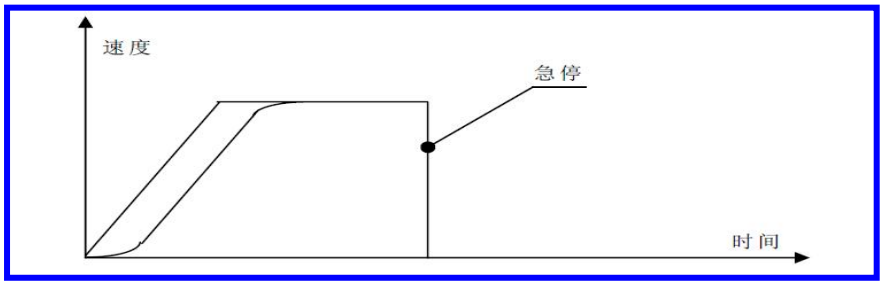


图 8-6 急停速度曲线

将Mode设置为1能够让控制轴平滑停止。调用平滑停止指令时，运动控制器将目标速度设置为0，并根据用户所设定的加速度沿着正在运动的方向平滑减速到0。

函数原型	short **MCF_Coordinate_Stop_Net** (unsigned short Coordinate, unsigned short Coordinate_Stop_Mode,

	unsigned short StationNumber = 0);
使用说明	坐标系停止运动
参数说明	Coordinate：指定坐标系 取值：0,1 Coordinate_Stop_Mode： 取值： Coordinate_Stop_IMD 0 //紧急停止 Coordinate_Stop_DEC 1 //减速停止 StationNumber：站号设置，默认不填为 0

示例：

1. 插补运动平滑停止

```
rtn = MCF_Set_Coordinate_Stop_Profile_Net (0,100000, 1000000,Profile_S,0);
//设置停止曲线参数
//坐标系：0
//停止加速度： 100000 脉冲/s*s
//停止加加速度：1000000 脉冲/s*s*s
//停止曲线：S 型
rtn = MCF_Coordinate_Stop_Net (0,Coordinate_Stop_DEC,0);
//执行轴 1 平滑停止
```

2. 插补运动立即停止

```
rtn = MCF_Coordinate_Stop_Net (0,Coordinate_Stop_IMD,0);
```

第 9 章 缓冲区连续轨迹运动

运动控制卡提供了缓冲区功能，可以通过创建缓冲区， 将需要执行的动作写入缓冲区中执行

函数原型	short **MCF_Buffer_Set_Stop_Profile_Net** (unsigned short Buffer_Number, double dMaxA, double dJerk, unsigned short Profile, unsigned short StationNumber = 0);
使用说明	设置缓冲区平滑停止参数
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） dMaxA： 设置平滑停止加速度。 单位：pulse/s*s 取值：dMaxA >0 dJerk：设置平滑停止加加速度 单位：pulse/s*s*s 取值：dJerk >0 Profile：停止曲线模式 取值： Profile_T 0 //T 型曲线 Profile_S 1 //S 型曲线 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Stop_Net** (unsigned short Buffer_Number, unsigned short Buffer_Stop_Mode, unsigned short StationNumber = 0);
使用说明	缓冲区停止指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Buffer_Stop_Mode：停止模式 取值： Buffer_Stop_IMD 0 //紧急停止 Buffer_Stop_DEC 1 //减速停止 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Change_Velocity_Ratio_Net** (unsigned short Buffer_Number, double Velocity_Ratio, unsigned short StationNumber = 0);
使用说明	设置缓冲区速度倍率
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Velocity_Ratio： 设置缓冲区速度倍率，取值：[0,2] StationNumber: 站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Start_Net** (unsigned short Buffer_Number, unsigned short StationNumber = 0);
使用说明	开辟缓冲区指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） StationNumber: 站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Set_Velocity_Ratio_Enable_Net** (unsigned short Buffer_Number, unsigned short Velocity_Ratio_Enable = 0, unsigned short StationNumber = 0);
使用说明	设置缓冲区速度倍率开关
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Velocity_Ratio_Enable： 设置缓冲区速度倍率开关， 取值： 0：不启用 1：启用 StationNumber: 站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Set_Reduce_Ratio_Net** (unsigned short Buffer_Number, double Reduce_Ratio = 1, unsigned short StationNumber = 0);

使用说明	设置缓冲区前瞻处理降速比
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Reduce_Ratio： 设置前瞻处理降速比 取值：[0,1] StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Set_Profile_Net** (unsigned short Buffer_Number, double dV_ini, double dMaxV, double dMaxA, double dJerk, double dV_end, unsigned short Profile , unsigned short StationNumber = 0);
使用说明	设置缓冲区运动参数指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） dV_ini： 设置单轴点动启动速度，单位：pulse/s 取值：dMaxV >dV_ini>=0 dMaxV：设置单轴点动目标速度，单位：pulse/s 取值：dMaxV >0 dMaxA：设置单轴点动加速度，单位：pulse/s 取值：dMaxA >0 dJerk：设置单轴点动加加速度，单位：pulse/s 取值：dJerk >0 dV_end：设置单轴点动终止速度，单位：pulse/s 取值：dMaxV > dV_end >=0 Profile：设置单轴点动曲线类型 取值： Profile_T 0 //T 型曲线 Profile_S 1 //S 型曲线 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Uniaxial_Net** (unsigned short Buffer_Number, unsigned short Axis, long dDist, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	缓冲区单轴点动指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Axis：指定轴号 取值：0-Axis_MAX dDist：设置点动目标位置，单位：脉冲 Position_Mode：设置坐标模式 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Sync_Follow_Net** (unsigned short Buffer_Number, unsigned short Axis, long dDist, unsigned short StationNumber = 0);
使用说明	缓冲区单轴运动距离同步跟随(法向跟随)（写在跟随动作语句前面）
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Axis：指定轴号(除被跟随的轴) 取值：0-Axis_MAX dDist：设置跟随距离；单位：脉冲（可根据转换角度换算成电机脉冲数） StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Line2_Net** (unsigned short Buffer_Number, unsigned short *Axis_List, long *dDist_List, unsigned short Position_Mode,

	unsigned short StationNumber = 0);
使用说明	缓冲区两轴直线插补指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表　　单位：pulse Position_Mode：设置坐标模式 取值： Position_Absolute　0　//绝对位置模式 Position_Opposite　1　//相对位置模式 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Line3_Net** (unsigned short Buffer_Number, unsigned short *Axis_List, long *dDist_List, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	缓冲区三轴直线插补指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表　　单位：pulse Position_Mode：设置坐标模式 取值： Position_Absolute　0　//绝对位置模式 Position_Opposite　1　//相对位置模式 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Line4_Net** (unsigned short Buffer_Number , unsigned short *Axis_List, long *dDist_List, unsigned short Position_Mode,

	unsigned short StationNumber = 0);
使用说明	缓冲区四轴直线插补指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表　　单位：pulse Position_Mode：设置坐标模式 取值： Position_Absolute　0　//绝对位置模式 Position_Opposite　1　//相对位置模式 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Arc_Radius_Net** (unsigned short Buffer_Number, unsigned short *Axis_List, long *dDist_List, long Arc_Radius, unsigned short Direction, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	缓冲区半径圆弧插补
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表　　单位：pulse Arc_Radius：设置圆弧半径　　单位：pulse Direction：设置圆弧运动方向 取值： Clock_Wise　　0　//顺弧 Counter_Clock_Wise　　1　//逆弧 Position_Mode：设置坐标模式 取值： Position_Absolute　0　//绝对位置模式 Position_Opposite　1　//相对位置模式

	StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Arc_Centre_Net** (unsigned short Buffer_Number, unsigned short *Axis_List, long *dDist_List, long *Center_List, unsigned short Direction, unsigned short Position_Mode, unsigned short StationNumber = 0);
使用说明	缓冲区圆心圆弧插补指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） *Axis_List：插补轴列表 取值：0-Axis_MAX *dDist_List：目的坐标列表 单位：pulse *Center_List：设置圆弧中心位置 单位：pulse Direction：设置圆弧运动方向 取值： Clock_Wise 0 //顺弧 Counter_Clock_Wise 1 //逆弧 Position_Mode：设置坐标模式 取值： Position_Absolute 0 //绝对位置模式 Position_Opposite 1 //相对位置模式 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Delay_Net** (unsigned short Buffer_Number, unsigned long number, unsigned short StationNumber = 0);
使用说明	缓冲区延时指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Number：设置缓冲区延时时间，单位：

	StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Set_Output_Bit_Net** (unsigned short Buffer_Number, unsigned short Bit_Number, unsigned short output, unsigned short StationNumber = 0);
使用说明	缓冲区操作输出信号指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Bit_Number：指定输出信号 Bit 位 Output：指定输出电平 取值： 0：输出低电平 1：输出高电平 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Wait_Input_Bit_Net** (unsigned short Buffer_Number, unsigned short Bit_Number, unsigned short Logic, unsigned short Time_Out, unsigned short StationNumber = 0);
使用说明	缓冲区等待输入信号指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Bit_Number：指定输出信号 Bit 位 Logic：指定等待输入电平 取值： 0：输出低电平 1：输出高电平 Time_Out：指定等待时间，单位：ms 取值：[0,65534] StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_End_Net** (unsigned short Buffer_Number, unsigned short StationNumber = 0);
使用说明	缓冲区指令写入完成指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Execute_Net** (unsigned short Buffer_Number, unsigned short Mode, unsigned short StationNumber = 0);
使用说明	启动缓冲区指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） Mode：缓冲区执行模式 取值： 0：表示根据启动与终止速度执行模式 1：表示根据拐点系数的速度前瞻模式 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Execute_BreakPoint_Net** (unsigned short Buffer_Number, unsigned short StationNumber = 0);
使用说明	缓冲区暂停恢复指令
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Get_State_Net** (unsigned short Buffer_Number, unsigned short *Execute_State, unsigned short *Execute_Number, unsigned short StationNumber = 0);
使用说明	缓冲区状态查询
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区）

	*Execute_State： 返回值定义： 0 正常命令执行成功 1 正在执行 2 EMG 立即紧急停止 3 EMG 减速紧急停止 4 ALM 立即停止 5 ALM 减速停止 6 伺服使能立即停止 7 伺服使能减速停止 8 指令编码器误差立即停止 9 指令编码器误差减速停止 10 Index 立即停止 11 Index 减速停止 12 原点立即停止 13 原点减速停止 14 正硬限位立即停止 15 正硬限位减速停止 16 负硬限位立即停止 17 负硬限位减速停止 18 正软限位立即停止 19 正软限位减速停止 20 负软限位立即停止 21 负软限位减速停止 22 命令立即停止 23 命令减速停止 24 其它原因立即停止 25 其它原因减速停止 26 未知原因立即停止 27 未知原因减速停止 28 外部 IO 减速停止 29 缓冲区正在执行 30 缓冲区 IO 等待超时 *Execute_Number：执行缓冲区号 StationNumber：指定模块站号，不填默认为 0
函数原型	short **MCF_Buffer_Get_Remainder_Space_Net**

	(unsigned short Buffer_Number, unsigned short *Remainder_Space_Ratio, unsigned short StationNumber = 0);
使用说明	缓冲区剩余可插入指令空间百分比查询
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） *Remainder_Space_Ratio：剩余缓冲区空间百分比（范围：0 - 100） StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Insert_Start_Net** (unsigned short Buffer_Number, unsigned short StationNumber = 0);
使用说明	缓冲区开始插入(建议查询到剩余有一半以上空间)
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Insert_End_Net** (unsigned short Buffer_Number, unsigned short StationNumber = 0);
使用说明	缓冲区结束插入
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Buffer_Count_Occupy_Space_Net** (unsigned short Buffer_Number, unsigned short *Occupy_Space_Ratio, unsigned short StationNumber = 0);
使用说明	计算加入指令所占用的空间百分比（在调用 MCF_Buffer_Start_Net 或者 MCF_Buffer_Insert_Start_Net 后开始从 0 计算）
参数说明	Buffer_Number：指定缓冲区号 取值：0（目前仅支持一个缓冲区） *Occupy_Space_Ratio：加入指令所占用的空间百分比 StationNumber：站号设置，默认不填为 0

9.1 单个缓冲区连续运动

为了方便地实现多段连续轨迹运动，用户可先将轨迹描述或参数指令存放在该缓冲区中（ 以缓冲区满为限），然后发出执行指令

示例:

1. 缓冲区点位运动

```
MCF_Buffer_Start_Net（0）       // 开辟 BANK 为 0
MCF_Buffer_Set_Profile_Net(0,0,10000,100000,1000000,0,PROFILE_T);
 //必须紧接着设置运动曲线参数，影响直到//下一个运动曲线参数出现
MCF_Buffer_Uniaxial_Net(0,Axis_1, 100000, Position_Opposite);     //运动轨迹
MCF_Buffer_Uniaxial_Net(0,Axis_2, 100000, Position_Opposite);     //运动轨迹
MCF_Buffer_Set_Profile_Net(0,0,20000,100000,1000000,0,PROFILE_T);//更新运动曲线参数
MCF_Buffer_Uniaxial_Net(0,Axis_3, 100000, Position_Opposite);     //运动轨迹
MCF_Buffer_End_Net(0, &number);//关闭当前打开缓冲区,存储完成并下载数据
MCF_Buffer_Execute_Net(0,1);                //启动 BANK0 连续轨迹运动
```

2. 缓冲区直线插补

```
unsigned long    Command_Number;
unsigned short Axis_List[2] = {Axis_1, Axis_2}; //设定轴号为 1 轴   2 轴
unsigned short dDist_List[2] = {10000, 100000};
 //设置轴号对应坐标为 1 轴：10000 脉冲  2 轴 100000 脉冲
MCF_Buffer_Start_Net(0);       // 开辟 BANK 为 0
MCF_Buffer_Set_Profile_Net(0,0,10000,100000,1000000,0,PROFILE_T);
//必须紧接着设置运动曲线参数，影响直到//下一个运动曲线参数出现
MCF_Buffer_Line2_Net(0, &Axis_List[Axis_1],    &dDist_List[Axis_1], Position_Opposite,0);
//2 轴直线插补，  相对位置
MCF_Buffer_End_Net(0, &number);//关闭当前打开缓冲区,存储完成并下载数据
MCF_Buffer_Execute_Net(0,Execute_Mode_1);              //启动 BANK0 连续轨迹运动
```

3. 缓冲区圆半径圆弧插补

```
unsigned long    Command_Number;
unsigned short Axis_List[2]    = {Axis_1, Axis_2};//设定轴号为 1 轴   2 轴
unsigned short dDist_List[2] = {10000, 100000};
//设置轴号对应坐标为 1 轴：10000 脉冲  2 轴 100000 脉冲
unsigned short Radius            = 5000;
MCF_Buffer_Start_Net(0);       // 开辟 BANK 为 0
MCF_Buffer_Set_Profile_Net(0,0,10000,100000,1000000,0,PROFILE_T);
//必须紧接着设置运动曲线参数，影响直到//下一个运动曲线参数出现
MCF_Buffer_Arc_Radius_Net            (0,&Axis_List[Axis_1],&dDist_List[Axis_1],
Radius,Clock_Wise,Position_Opposite, 0);
//2 轴圆半径圆弧插补，  相对位置
MCF_Buffer_End_Net(0, &Command_Number); //关闭当前打开缓冲区,存储完成并下载数据
MCF_Buffer_Execute_Net(0,1);              //启动 BANK0 连续轨迹运动
```

4. 缓冲区圆圆心圆弧插补

```
unsigned long    Command_Number;
```

```
unsigned short Axis_List[2]   = {Axis_1, Axis_2};//设定轴号为 1 轴   2 轴
unsigned short dDist_List[2] = {10000, 100000};
//设置轴号对应坐标为 1 轴：10000 脉冲 2 轴 100000 脉冲
unsigned short Center_List[2]= {5000,5000};//设置圆心点为 5000,5000；
MCF_Buffer_Start_Net(0);      // 开辟 BANK 为 0
MCF_Buffer_Set_Profile_Net(0,0,10000,100000,1000000,0,PROFILE_T);
//必须紧接着设置运动曲线参数，影响直到//下一个运动曲线参数出现
MCF_Buffer_Arc_Centre_Net(0, &Axis_List[Axis_1], &dDist_List[Axis_1],
&Center_List[Axis_1], Clock_Wise, Position_Opposite, 0);
//2 轴圆圆心圆弧插补，  相对位置
MCF_Buffer _End_Net (0,0);   //关闭当前打开缓冲区,存储完成并下载数据
MCF_Buffer_Execute_Net(0,1);              //启动 BANK0 连续轨迹运动
```

9.2 多个缓冲区连续运动(暂未开放)

缓冲区分配为2个BANK，这样用户就可以实现多个多段连续轨迹运动，并且可以在BANK指令和BANK指令之间做其他操作。这样就可以提前添入运动指令,提高指令执行效率.

示例:

1. 多个缓冲区切换

```
MCF_Buffer_Start_Net（0）       // 开辟 BANK 为 0
MCF_Buffer_Set_Profile_Net(0,0,10000,100000,1000000,0,PROFILE_T);
//必须紧接着设置运动曲线参数，影响直到下一个运动曲线参数出现
MCF_Buffer_Uniaxial_Net(0,Axis_1, 100000, Position_Opposite);    //运动轨迹
MCF_Buffer_Uniaxial_Net(0,Axis_2, 100000, Position_Opposite);    //运动轨迹
MCF_Buffer_Set_Profile_Net(0,0,20000,100000,1000000,0,PROFILE_T);
                                                          //更新运动曲线参数
MCF_Buffer_Uniaxial_Net(0,Axis_3, 100000, Position_Opposite);    //运动轨迹
MCF_Buffer _End_Net (0,0);   //关闭当前打开缓冲区,存储完成并下载数据
MCF_Buffer_Start_Net（1）       // 开辟 BANK 为 0
        MCF_Buffer_Set_Profile_Net(1,0,10000,100000,1000000,0,PROFILE_T);
                          //必须紧接着设置运动曲线参数，影响直到下一个运动曲线参数出现
MCF_Buffer_Line2_Net（1，&Axis_List[0], &dDist_List[0], Position_Opposite）;
MCF_Buffer_Set_Profile_Net(1,0,20000,100000,1000000,0,PROFILE_T);
                                                          //更新运动曲线参数
MCF_Buffer_Uniaxial_Net(1,Axis_3, 100000, Position_Opposite);    //运动轨迹
MCF_Buffer _End_Net (1,0);   //关闭当前打开缓冲区,存储完成并下载数据
MCF_Buffer_Execute_Net(0, 1);              //启动 BANK0 连续轨迹运动
do
{
MCF_Buffer_Get_State_Net（0, &Execute_State,&Execute_Number）;
Sleep(100);
}while(Execute_State == ERR_Buffer_Excute);  //判断缓冲区 0 是否完成
MCF_Buffer_Execute_Net (1, 1);                  //启动 BANK1 连续轨迹运动
do
{
MCF_Buffer_Get_State_Net（1, &Execute_State,&Execute_Number）;
```

```
Sleep(100);
}while(Execute_State == ERR_Buffer_Excute);              //判断缓冲区 1 是否完成
MCF_Buffer_Execute_Net (0, 1);                  //再次启动 BANK0 连续轨迹运动
do
{
MCF_Buffer_Get_State_Net（0, &Execute_State,&Execute_Number）;
Sleep(100);
}while(Execute_State == ERR_Buffer_Excute); //判断缓冲区 0 是否完成
```

9.3 通用IO缓冲区输出

MCF_Buffer_Set_Output_Bit_Net指令可以在缓冲区中直接对通用输出端口进行操作.

示例:

```
1.通用 IO 缓冲区输出
MCF_Buffer_Start_Net(0); // 打开 BANK 为 0，空间为 1*1024 的缓冲区
MCF_Buffer_Set_Output_Bit_Net(0,0,0);     //输出 BIT0 为 0
MCF_Buffer_Delay_Net（0,1000）;  //延时 1000MS
MCF_Buffer_Set_Output_Bit_Net (0,0,1);     //输出 BIT1 为 1
MCF_Buffer _End_Net (0,0);   //关闭当前打开缓冲区,存储完成并下载数据
MCF_Buffer_Execute_Net (0,0);                  //启动 BANK0 连续轨迹运动
```

9.4 通用IO缓冲区输入

MCF_Buffer_Wait_Input_Bit _Net指令可以在缓冲区中直接对通用输出端口进行操作.

示例:

```
1.通用 IO 缓冲区输入
MCF_Buffer_Start_Net(0); // 打开 BANK 为 0，空间为 1*1024 的缓冲区
MCF_Buffer_Wait_Input_Bit _Net(0,0,0,1000,0);
//指定等待 IO0 是否有低电平信号， 等待时间：1000ms
MCF_Buffer_Wait_Input_Bit _Net(0,1,0,1000,0);
//指定等待 IO1 是否有低电平信号， 等待时间：1000ms
MCF_Buffer _End_Net (0,0);   //关闭当前打开缓冲区,存储完成并下载数据
MCF_Buffer_Execute _Net(0,0);                  //启动 BANK0 连续轨迹运动
```

9.5 缓冲区断点重启

MCF_Buffer_Execute_BreakPoint_Net指令可以对指定BANK进行断点恢复。

示例:

```
1.缓冲区断点重启
//首先执行一个缓冲区
MCF_Buffer_Start_Net(0); // 打开 BANK 为 0，空间为 1*1024 的缓冲区
MCF_Buffer_Set_Profile_Net(0,0,10000,100000,1000000,0,PROFILE_T);
MCF_Buffer_Line2_Net（0，&Axis_List[0], &dDist_List[0], Position_Opposite）;//两轴直线插补
```

```
        MCF_Buffer _End_Net (0,0);   //关闭当前打开缓冲区,存储完成并下载数据
        MCF_Buffer_Execute_Net (0,0);                    //启动 BANK0 连续轨迹运动
    //2  缓冲区执行过程中  被外部触发停止或者用户调用运动停止
        MCF_Buffer_Stop_Net(0, 0);
    //3  调用缓冲区断点启动函数继续缓冲区运动
        MCF_Buffer_Execute_BreakPoint_Net(0);
```

9.6 缓冲区实现圆弧插补+法向跟随

MCF_Buffer_Sync_Follow_Net指令在特定场合实现刀具跟随旋转（刀具切割泡沫 等需要刀具跟随旋转的情况）。

示例:

实现一段半径为10000，目标位置为Axis_1 = 10000，Axis_2 = 10000的圆弧线段，法向跟随：Axis_3 = 1000。

```
unsigned short Axis_List[2] = {Axis_1, Axis_2};
long Dist_List[2] = {10000, 10000};
MCF_Buffer_Start_Net(0); // 打开 BANK 为 0
MCF_Buffer_Set_Profile_Net(0,0,10000,100000,1000000,0,PROFILE_T);
MCF_Buffer_Sync_Follow_Net(0,Axis_3,1000,0);
// 轴 3 对下面圆弧指令进行法向跟随，一定要写在跟随命令的前面
MCF_Buffer_Arc_Radius_Net (0,&Axis_List[0],&dDist_List[0],10000,0,1);
// 轴 1 和轴 2 做圆弧插补
MCF_Buffer _End_Net (0,0);   //关闭当前打开缓冲区,存储完成并下载数据
MCF_Buffer_Execute_Net (0,0);      //启动 BANK0 连续轨迹运动
```

9.7 缓冲区边执行边传输

运动控制卡提供缓冲区边执行边传输的功能。 缓冲区中所需要加入的指令占用空间大于缓冲区空间时，可先插入一部分空间的指令运行，在剩余空间足够的情况下再将剩下的缓冲区空间压入，可依次循环该步奏多次插入运动指令。

示例： 在循环中，实现边运行边传输数据的功能

示例说明：缓冲区可插入的运动指令较多，这里主要介绍的是实现思路，具体代码可参考 VC 例程-缓冲区-演示 1 插入缓冲区，具体的插入缓冲区的大小可自己去调节控制

注意： 1.插入缓冲区的大小不能大于剩余空间的大小，否则插入失败。

2.缓冲区运行中才可以插入缓冲区，未开始执行或者执行完成的情况下无法插入缓冲区。

```
Buffer_Step_1:  //压入 90%空间的缓冲区指令，  完成后跳到 Buffer_Step_1
While(1)
{
    MCF_Buffer_Start_Net(0);                                                    // 创建缓冲区
    MCF_Buffer_Set_Profile_Net(0,0,10000,100000,1000000,0,PROFILE_T);           // 设置运动参数
    MCF_Buffer_Get_Remainder_Space_Net(Buffer_Number, &Remainder_Space_Ratio, 0);
    if(Remainder_Space_Ratio < 10)            //检查剩余空间小于 10%时，  结束缓冲区同时启动缓冲区
    {
        MCF_Buffer _End_Net (0,0);          //关闭当前打开缓冲区,存储完成并下载数据
        MCF_Buffer_Execute_Net (0,0);       //启动 BANK0 连续轨迹运动
```

```
            Goto Buffer_Step_2:
        }
        else
        {
            //压入缓冲区运动指令
        }
}

Buffer_Step_2: 监视剩余空间是否大于 50%时，满足条件时跳到 Buffer_Step_3
While(1)
{
        MCF_Buffer_Get_Remainder_Space_Net(Buffer_Number, &Remainder_Space_Ratio, 0);
        if(Remainder_Space_Ratio > 50)//检查剩余空间大于 50%时，  插入缓冲区
        {
            MCF_Buffer_Insert_Start_Net(0,0);
            Goto Buffer_Step_3:
        }
}

Buffer_Step_3: 插入小于等于 50%的缓冲区空间的运动指令(剩余空间大于等于 50%)
{
        // 插入缓冲区填充指令
        {
            MCF_Buffer_Count_Occupy_Space_Net(Buffer_Number, &Occupy_Space_Ratio, 0);
            if(Occupy_Space_Ratio >= 50)//检查插入的空间数量是否大于等于 50%，满足条件结束插入缓冲区
            {
                MCF_Buffer_Insert_End_Net (0,0);   //插入缓冲区结束
            }
            else
            {
                //压入缓冲区运动指令
            }
        }
}
```

第 10 章 示波器 10K 采样频率数据捕捉

运动控制卡支持10K频率数据采集功能

支持电压范围：-10V~10V；

分辨率：16位 最小接收信号0.15；

支持通道数：8路；

10.1 数据捕捉打开/关闭

函数原型	short **MCF_Capture_Open_Net** (unsigned short Capture_Mode = 0);
使用说明	数据捕捉打开函数（必须在 MCF_Open_Net 函数前使用）
参数说明	Capture_Mode：采集模式 取值： Capture_Keep 0 //连续采样 Capture_Rise_Input_15 1 //输入 Bit_Input_15 上升沿触发捕抓开始 Capture_Fall_Input_15 2 //输入 Bit_Input_15 下升沿触发捕抓开始

函数原型	short **MCF_Capture_Close_Net** ();
使用说明	数据捕捉关闭函数
参数说明	

10.2 数据捕捉检查数据更新

函数原型	short **MCF_Capture_State_Net** (unsigned short *Capture_State);
使用说明	数据捕捉检查数据更新函数
参数说明	*Capture_State: 0: 数据更新中 1: 数据保持（更新完成）

10.3 读取采样连续的 1000 个位置命令数据

函数原型	short **MCF_Capture_Read_Command_Net** (unsigned short Axis,

	long *Command);
使用说明	数据捕捉读取函数
参数说明	Axis：轴号 *Command：命令数据

10.4 读取采样连续的 1000 个编码器数据

函数原型	short **MCF_Capture_Read_Encoder_Net** (unsigned short Axis, long *Encoder);
使用说明	数据捕捉读取函数
参数说明	Axis：轴号 *Encoder：编码数据

10.5 读取采样连续的 1000 个模拟量数据

函数原型	**MCF_Capture_Read_AD_Net** (unsigned short Axis, long *AD);
使用说明	数据捕捉读取函数
参数说明	Axis：轴号 *AD：采集数据

10.6 ADC 采样滤波

函数原型	short **MCF_Capture_Filter_AD_Net** (unsigned short Axis, double Filter_Coefficient = 1);
使用说明	ADC 采样滤波
参数说明	Axis：轴号 Filter_Coefficient：滤波系数，不填默认为 1

10.7 数据捕捉频率设置

函数原型	short **MCF_Capture_Frequency_Net** (unsigned short Capture_Frequency = 1, unsigned short StationNumber = 0);
使用说明	数据捕捉频率设置
参数说明	Capture_Frequency：频率； 取值：1　1K 采样频率 2　2K 采样频率 5　5K 采集频率 10　10K 采集频率 50　50K 采集频率 StationNumber：站号

示例：

```
1. 打开 AD 采用功能
unsigned short Station_Number[50]={0, 1, 2, 3, 4};//设置站号
unsigned short Station_Type[50]={2,0,2,2,2};  //设置对应类型
MCF_Capture_Open_Net                          (0);     //打开 AD 采集卡，连续采样
MCF_Open_Net(1,&Station_Number[0],&Station_Type[0]); //打开运动控制卡
2. 查询 AD 采用数据
Long Change_1[1000];//通道 1
Long Change_2[1000];//通道 2
Long Change_3[1000];//通道 3
Long Change_4[1000];//通道 4
{// 客户设置 100MS 以内定时，建议 50MS 检查一次
    MCF_Capture_State_Net(&Capture_State);//判断数据是否更新
    if (Capture_State == 1)//更新完成
    {
        MCF_Capture_Read_AD_Net(0,&Change_1[0]);//读取轴 1 AD 通道数据
        MCF_Capture_Read_AD_Net(1,&Change_2[0]);//读取轴 2 AD 通道数据
        MCF_Capture_Read_AD_Net(2,&Change_3[0]);//读取轴 3 AD 通道数据
        MCF_Capture_Read_AD_Net(3,&Change_4[0]);//读取轴 4 AD 通道数据
    }
}
```

10.8 数据捕捉缓存时间设置

函数原型	short MCF_Capture_Time_Net (unsigned short Capture_Time_1MS = 100, unsigned short StationNumber = 0);
使用说明	数据捕捉缓存时间设置
参数说明	Capture_Time_1MS：缓存时间； 取值：[2,1000]（2 的倍数） StationNumber：站号

第 11 章 电子齿轮控制

电子齿轮模式能够将两轴或多轴联系起来，实现精确的同步运动，从而替代传统的机械齿轮连接。一般被跟随的轴叫主轴，跟随的轴叫从轴。在电子齿轮模式下，有以下几个功能:

1: 1个主轴能够驱动多个从轴;

2: 从轴可以跟随主轴的规划位置、编码器位置。

传动比：主轴速度与从轴速度的比例。电子齿轮模式能够灵活的设置传动比，节省机械系统的安装时间。当主轴速度变化时，从轴会根据设定好的传动比自动改变速度。电子齿轮模式也能够在运动过程中修改传动比。

11.1 电子齿轮设置

函数原型	short **MCF_Set_Gear_Net** (unsigned short Axis, unsigned long Follow_Axis, unsigned long Denominator, unsigned long Molecule, unsigned long Follow_Source, unsigned long Dir, unsigned short StationNumber = 0);
使用说明	设置电子齿轮参数参数
参数说明	Axis：设置跟随的从轴 Follow_Axis：设置运动的主轴 Denominator： 设置电子齿轮比的分母 取值： (0，+∞) Molecule： 设置电子齿轮比的分子 取值： (0，+∞) Follow_Source：设置从轴跟随主轴的命令还是编码器 取值： 0：跟随命令 1：跟随编码器 Dir：设置跟随的方向 取值： 0：只跟随正方向 1：只跟随负方向 2：正负方向跟随 3: 正负方向跟随往正方向运动

	4：正负方向跟随往负方向运动 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Gear_Net** (unsigned short Axis, unsigned long *Follow_Axis, unsigned long *Denominator, unsigned long *Molecule, unsigned long *Follow_Source, unsigned long *Dir, unsigned short StationNumber = 0);
使用说明	读取电子齿轮参数参数
参数说明	Axis：设置跟随的从轴 *Follow_Axis：返回运动的主轴 *Denominator：返回电子齿轮比的分母 返回值：（0，+∞） *Molecule：返回电子齿轮比的分子 返回值：（0，+∞） *Follow_Source：返回从轴跟随主轴的命令还是编码器 *Dir：返回跟随的方向 StationNumber：站号设置，默认不填为 0

11.2 电子齿轮开关

函数原型	short **MCF_Set_Gear_Enable_Net** (unsigned short Axis, unsigned short Enable, unsigned short StationNumber = 0);
使用说明	设置电子齿轮功能启动关闭
参数说明	Axis：设置被跟随的主轴 Enable：电子齿轮使能 取值： 0：关闭电子齿轮 1：开启电子齿轮 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Gear_Enable_Net**

	(unsigned short Axis, unsigned short*Enable, unsigned short StationNumber = 0);
使用说明	设置电子齿轮功能状态
参数说明	Axis：设置被跟随的主轴 *Enable：返回电子齿轮启动关闭状态 StationNumber：站号设置，默认不填为 0

11.3 电子齿轮运动距离后自动关闭

函数原型	short **MCF_Set_Gear_Auto_Disable_Net** (unsigned short Axis, long dDist, unsigned short StationNumber = 0);
使用说明	电子齿轮运动距离后自动关闭
参数说明	Axis：设置轴 dDist：运动距离 StationNumber：站号设置，默认不填为 0

示例：

1. 两轴电子齿轮模式运动

```
rtn =MCF_Set_Gear_Net(Axis_1, Axis_2, 1, 4, 0, 0);
// Axis_1 和 Axis_2 两轴电子齿轮模式   从轴：Axis_1      主轴（运动轴）：Axis_2
// Axis_1 和 Axis_2 电子齿轮比 4/1
// Axis_1 跟随 Axis_2 的命令位置
rtn = MCF_Set_Gear_Enable_Net(Axis_1, 0);
//设置电子齿轮参数后，开启电子齿轮使能
```

第 12 章 位置比较输出函数

运动控制卡提供了位置触发输出信号的函数，包括单轴低速位置比较、单轴高速位置比较。当电机运动到预先设置的位置时，自动触发特定的输出口。该功能在轨迹运动中用于控制点胶阀的开关、触发照相机快门等动作十分方便。

12.1 设置一维位置比较器

函数原型	short **MCF_Set_Compare_Config_Net** (unsigned short Axis, unsigned short Enable, unsigned short Compare_Source, unsigned short StationNumber = 0);
使用说明	设置一维位置比较器
参数说明	Axis：位置比较通道； Enable：设置位置比较使能 取值： 0：取消使能 1：使能 Compare_Source：设置比较类型 取值： 0：命令位置比较 1：编码器值比较 StationNumber：站号设置，默认不填为 0

函数原型	short **MCF_Get_Compare_Config_Net** (unsigned short Axis, unsigned short *Enable, unsigned short *Compare_Source, unsigned short StationNumber = 0);
使用说明	获取一维位置比较器参数设置
参数说明	Axis：位置比较通道； *Enable：获取比较使能设置 *Compare_Source：获取比较类型设置 StationNumber：站号设置，默认不填为 0

12.2 清除一维位置所有/当前比较点/关闭任意点

函数原型	short **MCF_Clear_Compare_Points_Net** (unsigned short Axis, unsigned short StationNumber = 0);
使用说明	清除一维位置比较点
参数说明	Axis：位置比较通道； StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Clear_Compare_Current_Points_Net** (unsigned short Axis, unsigned short StationNumber = 0);
使用说明	清除当前比较点数据
参数说明	Axis：位置比较通道； StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Disable_Compare_Any_Points_Net** (unsigned short Axis, unsigned long Point_Number, unsigned short StationNumber = 0);
使用说明	禁用指定比较点
参数说明	Axis：位置比较通道； Point_Number：指定禁用比较点号(以全部添加比较点开始计算取余 256) StationNumber：指定模块站号，不填默认为 0

12.3 添加一维位置比较点

函数原型	short **MCF_Add_Compare_Point_Net** (unsigned short Axis, long Position, unsigned short Dir, unsigned short Action, unsigned short Actpara, unsigned short StationNumber = 0);
使用说明	添加一维位置比较点
参数说明	Axis：位置比较通道；

Position：位置比较点距离电机运动开始的位置

Dir： 位置比较点触发方向

取值：

0：小于等于

1：大于等于

2：正负同方向小于等于

3：正负同方向大于等于

注意：模式 2 和模式 3， 是为了处理在一个设备工作周期中， 编码器（位置）储存变量溢出后，导致数据无法正常比较的情况。

现象：设定当前位置是正值，比较点位置为正数时，进行比较。当给定的比较点位置为负值时，不比较，只有当前位置值增加到溢出，值变为负数，才跟给定的负的比较值进行比较。避免了不同符号数值之间比较错误的情况，设备运行过程中无需进行数据清零或者比较功能停止/重启的情况。

Action：触发动作

取值：

action	actpara	功能
0	IO 号	禁用输出
1	IO 号	IO 断路输出
2	IO 号	IO 开漏输出低电平
3	IO 号	取反 IO
5	IO 号	输出 500us 低脉冲
6	IO 号	输出 1ms 低脉冲
7	IO 号	输出 10ms 低脉冲
8	IO 号	输出 100ms 低脉冲
9	IO 号	输出 2ms 低脉冲
10	IO 号	输出 3ms 低脉冲
11	IO 号	输出 4ms 低脉冲
12	IO 号	输出 5ms 低脉冲
13	IO 号	输出 6ms 低脉冲
14	IO 号	输出 7ms 低脉冲

	15	IO 号	输出 8ms 低脉冲
	16	IO 号	输出 9ms 低脉冲
	17	IO 号	输出 10ms 低脉冲
	18	IO 号	输出 20ms 低脉冲
	19	IO 号	输出 30ms 低脉冲
	20	IO 号	输出 40ms 低脉冲
	21	IO 号	输出 50ms 低脉冲
	22	IO 号	输出 60ms 低脉冲
	23	IO 号	输出 70ms 低脉冲
	24	IO 号	输出 80ms 低脉冲
	25	IO 号	输出 90ms 低脉冲
	26	IO 号	输出 100ms 低脉冲
	27	IO 号	输出 200ms 低脉冲
	28	IO 号	输出 300ms 低脉冲
	29	IO 号	输出 400ms 低脉冲
	30	IO 号	输出 500ms 低脉冲
	Actpara,：输出位置 取值： DO00 - DO31 StationNumber：站号设置，默认不填为 0		

12.4 读取当前一维比较点位置

函数原型	short **MCF_Get_Compare_Current_Point_Net** (unsigned short Axis, long *Position, unsigned short StationNumber = 0);
使用说明	读取当前一维比较点位置

参数说明	Axis：位置比较通道； *Position：获取触发位置 StationNumber：站号设置，默认不填为 0

12.5 查询已经比较过的一维比较点个数

函数原型	short **MCF_Get_Compare_Points_Runned_Net** (unsigned short Axis, long *Point_Number, unsigned short StationNumber = 0);
使用说明	查询已经比较过的一维比较点个数(注意数据溢出)
参数说明	Axis：位置比较通道； Point_Number：比较点个数 StationNumber：站号设置，默认不填为 0

12.6 查询可以加入的一维比较点个数

函数原型	short **MCF_Get_Compare_Points_Remained_Net** (unsigned short Axis, long *Point_Number, unsigned short StationNumber = 0);
使用说明	查询可以加入的一维比较点个数
参数说明	Axis：位置比较通道； *Point_Number：可以加入的一维比较点个数 StationNumber：站号设置，默认不填为 0

12.7 查询所有未完成一维比较点个数和位置

函数原型	short **MCF_Get_Compare_Points_Incomplete_Net** (unsigned short Axis, unsigned short *Incomplete_Number, long *Incomplete_Position, unsigned short StationNumber = 0);
使用说明	查询所有未完成一维比较点个数和位置
参数说明	Axis：位置比较通道； *Incomplete_Number：存储未比较点的个数； *Incomplete_Position：存储未比较点的位置数组； 需 256 个元素大小的数组 StationNumber：指定模块站号，不填默认为 0

注意事项： 每个轴的位置比较都是独立进行的

执行位置比较时，每个比较点的触发是按照添加的比较点顺序执行的，即如果一个比较点没有被触发比较动作，那么后面的比较点是不会起作用的；

示例：

1. 一维位置比较

```
//设置轴号 0，比较器使能， 比较源：指令位置， 站号：0；
MCF_Set_Compare_Config_Net                    (0, 1, 0, 0);
//清除 0 轴比较点个数                          设置轴号 0， 站号：0；
MCF_Clear_Compare_Points_Net                  (0, 0);
//添加 0 轴比较点 位置 10000PULSE 比较模式：1（大于等于）， 触发功能：2（置为低电平）
端口触发 IO：0 站号：0；
MCF_Add_Compare_Point_Net                     (0, 10000, 1, 2, 0, 0);
//添加 0 轴比较点 位置 20000PULSE 比较模式：1（大于等于）， 触发功能：1（置为高电平）
端口触发 IO：3 站号：0；
MCF_Add_Compare_Point_Net                     (0, 20000, 1, 1, 3, 0);
//0 轴按 3000PULSE/s 的速度往正方向持续运动
MCF_JOG_Net                                   (0, 3000, 100000, 0);
```

运行结果：

当运动到10000脉冲是 通用D000输出低电平 当运动到20000脉冲公式 通用DI03输出高电平；

12.8 绑定光源频闪功能

函数原型	short **MCF_Set_Compare_Light_Frequency_Net** (unsigned short Axis, unsigned short Light_Number, unsigned short Frequency_Enable, unsigned short StationNumber = 0);
使用说明	设置一维位置比较器光源频闪功能 **（先设置好频闪模式， 参考 第 16 章）**
参数说明	Axis：位置比较通道； Light_Number：光源通道； Frequency_Enable：启用光源； 取值： 0 不启用 1 启用 StationNumber：指定模块站号，不填默认为 0

第 13 章 PWM 输出

运动控制卡支持 PWM 输出功能，用户只需要简单设置 PWM 输出通道、频率及占空比参数即可实现 PWM 输出功能，非常方便。

13.1 设置 PWM 输出参数

函数原型	short **MCF_Set_Pwm_Config_Net** (unsigned short Channel, unsigned short Enable, unsigned short Output_Port_Config, unsigned short Output_Start_Logic, unsigned short Logic, unsigned short StationNumber = 0);
使用说明	设置 PWM 输出参数
参数说明	Channel：通道号 取值： PWM_Channel_1 0 //第 1 通道 PWM_Channel_2 1 //第 2 通道 Enable：PWM 使能 取值： PWM_Disable 0 //PWM 不使能 PWM_Enable 1 //PWM 使能 Output_Port_Config：输出位号 取值： PWM_Output_14 14 //输出点 14 PWM_Output_15 15 //输出点 15 PWM_AXIS_4_Pluse 22 //轴 4 脉冲输出点 PWM_AXIS_4_Dir 23 //轴 4 方向输出点 Output_Start_Logic： 输出开始电平 取值： PWM_Start_Low_Logic 0 //PWM 默认触点闭合,硬件灯亮 PWM_Start_High_Logic 1//PWM 默认触点打开,硬件灯灭 StationNumber: 站号设置，默认不填为 0

函数原型	short **MCF_Get_Pwm_Config_Net** (unsigned short Channel,

	unsigned short *Enable, unsigned short *Output_Port_Config, unsigned short *Output_Start_Logic, unsigned short StationNumber = 0);
使用说明	读取 PWM 输出参数
参数说明	Channel：通道号 *Enable：读取 PWM 使能设置 *Output_Port_Config：读取输出位号 *Output_Start_Logic： 读取输出开始电平 StationNumber：站号设置，默认不填为 0

13.2 输出 PWM 信号

函数原型	short **MCF_Set_Pwm_Output_Net** (unsigned short Channel, unsigned long Frequency, unsigned long DutyCycle, unsigned long Pwm_Number, unsigned short StationNumber = 0);
使用说明	输出 PWM 信号
参数说明	Channel：通道号 取值： PWM_Channel_1 0 //第 1 通道 PWM_Channel_2 1 //第 2 通道 Frequency：频率 取值：[0,1000000] DutyCycle：占空比 取值：[0,100] Pwm_Number：PWM 个数 单位:MS 取值：(0，2^31-1] **其中 0xFFFFFFFF：表示一直输出** StationNumber：站号设置，默认不填为 0

13.3 PWM 完成信号

函数原型	short **MCF_Get_Pwm_State_Net** (unsigned short Channel, unsigned short *Finish,

	unsigned short StationNumber = 0);
使用说明	PWM 完成信号**(暂未开放)**
参数说明	Channel：通道号 取值： PWM_Channel_1 0 //第 1 通道 PWM_Channel_2 1 //第 2 通道 *Finish：完成状态 返回值定义： PWM_Free 0 //PWM 输出空闲 PWM_Busy 1 //PWM 输出忙 StationNumber: 站号设置，默认不填为 0

示例：

1.DO14 输出 100HZ 占空比位 50%的波形

```
rtn = MCF_Set_Pwm_Config_Net(PWM_Channel_1,1,0,0,0);
//设置通道 1
//输出使能：使能
//输出位置：DO14
//开始电平：低电平
rtn = MCF_Set_Pwm_Output_Net(PWM_Channel_1,100,50,100,0);
//设置通道 1
//输出频率：100HZ
//占空比：50%
//PWM 数量：100
```

第 14 章 手轮

支持手轮控制

支持通道数：1路；

控制轴数：4轴；

倍率：1倍率、10倍率、100倍率

12.8 绑定光源通道

函数原型	short **MCF_Set_Compare_Light_Frequency_Net** (unsigned short Axis, unsigned short Light_Number, unsigned short Frequency_Enable, unsigned short StationNumber = 0);
使用说明	设置一维位置比较器光源频闪功能 （光源模式为频闪情况才可以绑定　参考第 16 章）
参数说明	Axis：位置比较通道； Light_Number：光源通道号； Frequency_Enable：光源使能； 取值　0　解开绑定 　　1　绑定 StationNumber：指定模块站号，不填默认为 0

14.1 开启手轮功能

函数原型	short **MCF_Hand_Wheel_Open_Net** (unsigned short Dir, unsigned short StationNumber = 0);
使用说明	打开手轮功能
参数说明	Dir：跟随类型 取值： Dir_P_T_P 0 //跟随正同方向走 Dir_N_T_N 1 //跟随负同方向走 Dir_PN_T_PN 2 //跟随正负同方向走 Dir_PN_T_P 3 //跟随正负都往正方向走 Dir_PN_T_N 4 //跟随正负都往负方向走 StationNumber：指定模块站号，不填默认为 0

14.2 关闭手轮功能

函数原型	short **MCF_Hand_Wheel_Close_Net** (unsigned short StationNumber = 0);
使用说明	关闭手轮功能
参数说明	StationNumber：指定模块站号，不填默认为 0

14.3 设置硬件手轮编码器通道

函数原型	short **MCF_Hand_Wheel_Config_Encoder_Net** (unsigned short Axis, unsigned short StationNumber = 0);
使用说明	设置硬件手轮编码器通道
参数说明	Axis：轴号，对应的编码器 StationNumber：指定模块站号，不填默认为 0

14.4 设置硬件手轮速率配置

14.4.1 设置硬件手轮速率输入点

函数原型	short **MCF_Hand_Wheel_Config_X1_Net** (unsigned short Bit_Input_Number,

	unsigned short StationNumber = 0);
使用说明	设置硬件手轮速率配置输入点
参数说明	Bit_Input_Number：设置输入点位号 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Hand_Wheel_Config_X10_Net** (unsigned short Bit_Input_Number, unsigned short StationNumber = 0);
使用说明	设置硬件手轮速率配置输入点
参数说明	Bit_Input_Number：设置输入点位号 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Hand_Wheel_Config_X100_Net** (unsigned short Bit_Input_Number, unsigned short StationNumber = 0);
使用说明	设置硬件手轮速率配置输入点
参数说明	Bit_Input_Number：设置输入点位号 StationNumber：指定模块站号，不填默认为 0

14.4.2 设置硬件手轮速率大小

函数原型	short **MCF_Hand_Wheel_Speed_X1_Net** (unsigned short Speed_X = 1, unsigned short StationNumber = 0);
使用说明	设置硬件手轮速率大小
参数说明	Speed_X：设置倍率大小；　默认值：1 倍率； StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Hand_Wheel_Speed_X10_Net** (unsigned short Speed_X = 10, unsigned short StationNumber = 0);
使用说明	设置硬件手轮速率大小
参数说明	Speed_X：设置倍率大小；　默认值：10 倍率； StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Hand_Wheel_Speed_X100_Net** (unsigned short Speed_X = 100, unsigned short StationNumber = 0);
使用说明	设置硬件手轮速率大小
参数说明	Speed_X：设置倍率大小；　默认值：100 倍率； StationNumber：指定模块站号，不填默认为 0

14.5 设置硬件手轮轴号配置输入点

函数原型	short **MCF_Hand_Wheel_Config_Axis_Net** (unsigned shrot Axis, unsigned short Bit_Input_Number, unsigned short StationNumber = 0);
使用说明	设置硬件手轮速率配置输入点
参数说明	Axis：输出脉冲轴号 Bit_Input_Number：设置输入点位号 StationNumber：指定模块站号，不填默认为 0

14.6 设置手轮运动平滑滤波时间

函数原型	short **MCF_Hand_Wheel_Config_Filter_Time_Net** (unsigned short Axis, unsigned long Filter_Time_1MS, unsigned short StationNumber = 0);
使用说明	设置手轮运动平滑滤波时间 **注意：在手轮使用中有调用该函数设置滤波时间不为 0，则手轮关闭时需要调用该函数设置滤波时间为 0。**
参数说明	Axis：输出脉冲轴号 Filter_Time_1MS：设置滤波时间　取值：1 - 1000ms StationNumber：指定模块站号，不填默认为 0

示例：

1. 设置手轮

```
MCF_Hand_Wheel_Config_Encoder_Net(Axis_1, 0);      //设置通道为轴 1 对应编码器；
MCF_Hand_Wheel_Config_X1_Net(Bit_Input_0, 0);      //设置 DI00 为 X1 倍率对应输入点
MCF_Hand_Wheel_Config_X10_Net(Bit_Input_1, 0);     //设置 DI01 为 X10 倍率对应输入点
MCF_Hand_Wheel_Config_X100_Net(Bit_Input_2, 0);    //设置 DI02 为 X100 倍率对应输入点
```

```
MCF_Hand_Wheel_Speed_X1_Net(2, 0);                    //修改 X1 的倍率为 2 倍率
MCF_Hand_Wheel_Speed_X10_Net(5, 0);                   //修改 X10 的倍率为 5 倍率
MCF_Hand_Wheel_Speed_X100_Net(10, 0);                 //修改 X100 的倍率为 10 倍率
MCF_Hand_Wheel_Config_Axis_Net(Axis_1, Bit_Input_3, 0); //设置 DI3 为对应轴 1 输入轴
MCF_Hand_Wheel_Config_Axis_Net(Axis_2, Bit_Input_4, 0); //设置 DI4 为对应轴 2 输入轴
MCF_Hand_Wheel_Config_Axis_Net(Axis_3, Bit_Input_5, 0); //设置 DI5 为对应轴 3 输入轴
MCF_Hand_Wheel_Config_Axis_Net(Axis_4, Bit_Input_6, 0); //设置 DI6 为对应轴 4 输入轴
MCF_Hand_Wheel_Open_Net(2, 0);   //打开手轮，  方向：跟随正负方向；
```

第 15 章 模拟量输入输出

运动控制卡支持模拟量输入输出功能，支持通过模拟量输入捕获其他位号的模拟量和轴位置

15.1 读取单次 ADC 采样

函数原型	short **MCF_Single_Read_AD_Net** (unsigned short Channel, short *AD, unsigned short StationNumber = 0);
使用说明	读取单次 ADC 采样
参数说明	Channel：通道号 *AD：AD 采集数据 StationNumber：指定模块站号，不填默认为 0

15.2 读取单次 DAC 输出

函数原型	short **MCF_Single_Write_DA_Net** (unsigned short Channel, short DA, unsigned short StationNumber = 0);
使用说明	读取单次 DAC 输出
参数说明	Channel：通道号 DA：DA 输出； StationNumber：指定模块站号，不填默认为 0

15.3 设置 AD 双向比较器停止对应轴

函数原型	short **MCF_Set_AD_Compare_Net** (unsigned short Channel, short AD_Compare, unsigned short Stop_Axis, unsigned short StationNumber = 0);
使用说明	设置 AD 双向比较器停止对应轴
参数说明	Channel：通道号 AD_Compare：AD 采集数据 Stop_Axis：对应停止轴

	StationNumber：指定模块站号，不填默认为 0

15.4 设置 AD 触发数值

函数原型	short **MCF_Set_AD_Capture_Net** (unsigned short Channel, short AD_Capture, unsigned short StationNumber = 0);
使用说明	设置 AD 触发数值
参数说明	Channel：通道号　取值：0 - 7； Stop_Axis：对应停止轴 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Clear_AD_Capture_Net** (unsigned short C_1, unsigned short C_2, unsigned short C_3, unsigned short C_4, unsigned short C_5, unsigned short C_6, unsigned short C_7, unsigned short C_8, unsigned short StationNumber = 0);
使用说明	清除 AD 捕获模式和数值
参数说明	C_1：通道 1;　取值：0：保持不变　1：清除复位; C_2：通道 2;　取值：0：保持不变　1：清除复位; C_3：通道 3;　取值：0：保持不变　1：清除复位; C_4：通道 4;　取值：0：保持不变　1：清除复位; C_5：通道 5;　取值：0：保持不变　1：清除复位; C_6：通道 6;　取值：0：保持不变　1：清除复位; C_7：通道 7;　取值：0：保持不变　1：清除复位; C_8：通道 8;　取值：0：保持不变　1：清除复位; StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Capture_AD_1_Net** (short *AD_5,

	long *Position_1, unsigned short StationNumber = 0);
使用说明	读取到达设置通道 1 的 AD 值，捕获通道 5 的 AD 值和轴 X 位置
参数说明	AD_5：通道 5 Position_1：轴 X（轴 1）位置 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Capture_AD_2_Net** (short *AD_6, long *Position_1, unsigned short StationNumber = 0);
使用说明	读取到达设置通道 2 的 AD 值，捕获通道 6 的 AD 值和轴 X 位置
参数说明	AD_6：通道 6 Position_1：轴 X（轴 1）位置 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Capture_AD_3_Net** (short *AD_7, long *Position_1, unsigned short StationNumber = 0);
使用说明	读取到达设置通道 3 的 AD 值，捕获通道 7 的 AD 值和轴 X 位置
参数说明	AD_7：通道 7 Position_1：轴 X（轴 1）位置 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Capture_AD_4_Net** (short *AD_8, long *Position_1, unsigned short StationNumber = 0);
使用说明	读取到达设置通道 4 的 AD 值，捕获通道 8 的 AD 值和轴 X 位置
参数说明	AD_8：通道 8 Position_1：轴 X（轴 1）位置 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Capture_AD_5_Net**

	(short *AD_1, long *Position_1, unsigned short StationNumber = 0);
使用说明	读取到达设置通道 5 的 AD 值，捕获通道 1 的 AD 值和轴 X 位置
参数说明	AD_1：通道 1 Position_1：轴 X（轴 1）位置 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Capture_AD_6_Net** (short *AD_2, long *Position_1, unsigned short StationNumber = 0);
使用说明	读取到达设置通道 6 的 AD 值，捕获通道 2 的 AD 值和轴 X 位置
参数说明	AD_2：通道 2 Position_1：轴 X（轴 1）位置 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Capture_AD_7_Net** (short *AD_3, long *Position_1, unsigned short StationNumber = 0);
使用说明	读取到达设置通道 7 的 AD 值，捕获通道 3 的 AD 值和轴 X 位置
参数说明	AD_3：通道 3 Position_1：轴 X（轴 1）位置 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Capture_AD_8_Net** (short *AD_4, long *Position_1, unsigned short StationNumber = 0);
使用说明	读取到达设置通道 8 的 AD 值，捕获通道 4 的 AD 值和轴 X 位置
参数说明	AD_4：通道 4 Position_1：轴 X（轴 1）位置 StationNumber：指定模块站号，不填默认为 0

15.5 设置触发位置数值

函数原型	short **MCF_Set_Position_Capture_AD_Net** (unsigned short Channel, long Position_1, unsigned short StationNumber = 0);
使用说明	设置触发位置数值
参数说明	Channel：通道号 Position_1：轴 X（轴 1）位置 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Position_Capture_AD_Net** (unsigned short Channel, short *AD, unsigned short StationNumber = 0);
使用说明	获取触发后对应的通道 AD 值
参数说明	Channel：通道号 AD：返回对应通道的 AD 值 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Clear_Position_Capture_AD_Net** (unsigned short C_1, unsigned short C_2, unsigned short C_3, unsigned short C_4, unsigned short C_5, unsigned short C_6, unsigned short C_7, unsigned short C_8, unsigned short StationNumber = 0);
使用说明	清除触发位置对应存储的 AD 值
参数说明	C_1：通道 1;　取值：0：保持不变　1：清除复位; C_2：通道 2;　取值：0：保持不变　1：清除复位; C_3：通道 3;　取值：0：保持不变　1：清除复位; C_4：通道 4;　取值：0：保持不变　1：清除复位;

	C_5：通道 5;　取值：0：保持不变　1：清除复位; C_6：通道 6;　取值：0：保持不变　1：清除复位; C_7：通道 7;　取值：0：保持不变　1：清除复位; C_8：通道 8;　取值：0：保持不变　1：清除复位; StationNumber：指定模块站号，不填默认为 0

15.6 获取 AD 最大值

函数原型	short **MCF_Get_Limit_AD_Net** (unsigned short Channel, short *MAX_AD, short *MIN_AD, unsigned short StationNumber = 0);
使用说明	获取触发后对应的通道 AD 值
参数说明	Channel：通道号 MAX_AD：返回对应通道的最大 AD 值 MIN_AD,：返回对应通道的最小 AD 值 StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Clear_Limit_AD_Net** (unsigned short C_1, unsigned short C_2, unsigned short C_3, unsigned short C_4, unsigned short C_5, unsigned short C_6, unsigned short C_7, unsigned short C_8, unsigned short StationNumber = 0);
使用说明	清除存储的 AD 值
参数说明	C_1：通道 1;　取值：0：保持不变　1：清除复位; C_2：通道 2;　取值：0：保持不变　1：清除复位; C_3：通道 3;　取值：0：保持不变　1：清除复位; C_4：通道 4;　取值：0：保持不变　1：清除复位; C_5：通道 5;　取值：0：保持不变　1：清除复位; C_6：通道 6;　取值：0：保持不变　1：清除复位;

	C_7：通道 7;　取值：0：保持不变　1：清除复位; C_8：通道 8;　取值：0：保持不变　1：清除复位; StationNumber：指定模块站号，不填默认为 0

示例：

1. 读取单次 ADC 采样

```
unsigned short AD;
MCF_Single_Read_AD_Net(1, &AD);
```

2. 设置单次 DAC 输出

```
short DA = 10;
MCF_Single_Write_DA_Net(1, 10);
```

第 16 章 光源控制函数

运动控制卡支持光源控制功能，支持通过光源控制卡来控制光源的启动模式

16.1 设置光源模式

调用 **MCF_Set_Light_Mode_Net** 设置此通道的光源模式；

函数原型	short **MCF_Set_Light_Mode_Net** (unsigned short Channel, unsigned short Light_Mode, unsigned short StationNumber = 0);
使用说明	设置光源通道的模式
参数说明	Channel：光源通道 Light_Mode：光源模式 取值： 0：关闭 1:24V 常亮 2:24V 频闪 3:48V 爆闪 StationNumber：指定模块站号，不填默认为 0

16.2 设置电流保护

调用 **MCF_Set_Light_Current_Net** 设置光源过载保护；

函数原型	short **MCF_Set_Light_Current_Net** (unsigned short Channel, unsigned short Max_Current, unsigned short StationNumber = 0);
使用说明	设置光源电流过载保护
参数说明	Channel：光源通道 Max_Current：过载保护电流　　单位：mA StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Light_Current_1_4_Net** (unsigned short *Current_1,

	unsigned short *Current_2, unsigned short *Current_3, unsigned short *Current_4, unsigned short StationNumber = 0);
使用说明	读取光源通道 1 到通道 4 的电流
参数说明	*Current_1：光源通道 1 电流　　单位 mA *Current_2：光源通道 2 电流　　单位 mA *Current_3：光源通道 3 电流　　单位 mA *Current_4：光源通道 4 电流　　单位 mA StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Get_Light_Current_5_8_Net** (unsigned short *Current_5, unsigned short *Current_6, unsigned short *Current_7, unsigned short *Current_8, unsigned short StationNumber = 0);
使用说明	读取光源通道 5 到通道 8 的电流
参数说明	*Current_5：光源通道 5 电流　　单位 mA *Current_6：光源通道 6 电流　　单位 mA *Current_7：光源通道 7 电流　　单位 mA *Current_8：光源通道 8 电流　　单位 mA StationNumber：指定模块站号，不填默认为 0

16.3 设置光源输出

函数原型	short **MCF_Set_Light_Output_Net** (unsigned short Channel, unsigned short Light_Size, unsigned short StationNumber = 0);
使用说明	设置光源输出函数
参数说明	Channel：光源通道 Light_Size：亮度/时间 取值： 　　常亮模式下为亮度：0~255

	频闪或爆闪模式下为时间：0~1000　单位：ms StationNumber：指定模块站号，不填默认为 0

16.4 设置输入触发光源输出

函数原型	short **MCF_Set_Light_Trigger_Net** (unsigned short Channel, unsigned short Bit_Input_Number, unsigned long Trigger_Mode, unsigned short StationNumber = 0);
使用说明	设置光源输出函数
参数说明	Channel：光源通道 Bit_Input_Number：触发 DI 口位　范围：DI00-DI47 Trigger_Mode：硬件触发模式 取值： 0：关闭 1：低电平(触点闭合,硬件灯亮)触发 2：高电平(触点闭合,硬件灯亮)触发 StationNumber：指定模块站号，不填默认为 0

第 17 章 系统函数

17.1 模块版本号

调用 **MCF_Get_Version_Net** 获取当前运动控制卡的程序版本；

函数原型	short **MCF_Get_Version_Net** (unsigned long *Version, unsigned short StationNumber = 0);
使用说明	获取固件版本
参数说明	*Version：返回模块固件版本 StationNumber：指定模块站号，不填默认为 0

17.2 序列号

调用 **MCF_Get_Serial_Number_Net** 获取当前运动控制卡 64 位序列号；

函数原型	short **MCF_Get_Serial_Number_Net** (INT64 *Serial_Number, unsigned short StationNumber = 0);
使用说明	序列号
参数说明	*Serial_Number：序列号 StationNumber：指定模块站号，不填默认为 0

17.3 模块运行时间

调用 **MCF_Get_Run_Time_Net** 获取运动卡通电至今的时间；

函数原型	short **MCF_Get_Serial_Number_Net** (unsigned long *Run_Time, unsigned short StationNumber = 0);
使用说明	获取运动卡通电至今的时间
参数说明	*Run_Time：通电时间（单位：秒） StationNumber：指定模块站号，不填默认为 0

17.4 Flash 读写

调用 **MCF_Flash_Write_Net** 和 **MCF_Flash_Read_Net** 可以对控制卡进行 Flash 缓冲区写入与读取（Labview 平台不支持 Flash 读写函数）；

函数原型	short **MCF_Flash_Write_Net** (unsigned long Pass_Word_Setup, unsigned long *Flash_Write_Data, unsigned short StationNumber = 0);
使用说明	获取运动卡通电至今的时间
参数说明	Pass_Word_Setup：Flash 写入密码 *Flash_Write_Data：Flash 写入内容（为 256 个数组） StationNumber：指定模块站号，不填默认为 0

函数原型	short **MCF_Flash_Read_Net** (unsigned long Pass_Word_Check, unsigned long *Flash_Read_Data, unsigned short StationNumber = 0);
使用说明	获取运动卡通电至今的时间
参数说明	Pass_Word_Check：Flash 读取密码 *Flash_Read_Data：Flash 读取内容（为 256 个数组） StationNumber：指定模块站号，不填默认为 0

17.8 回调函数

函数原型	short **MCF_Set_CallBack_Net** (long CallBack, uint Time_1MS);
使用说明	系统定时回调函数 （注意：不能在回调函数里面做暂用时间过长，工作量比较大的代码（如：打印信息到窗口显示），适合做一些结果综合处理）
参数说明	CallBack：回调函数名 Time_1MS：回调周期时间；

17.9 系统日志函数

函数原型	short **MCF_Set_Log_Enable_Net** (unsigned long Code, unsigned short Enable);
使用说明	命令采集使能函数
参数说明	Code：命令编号 Enable：命令采集使能 取值： 0 //不进行采集 1 //进行采集

函数原型	short **MCF_Get_Log_Count_Net** (unsigned long *Count);
使用说明	获取指令采集总数
参数说明	Count：采集到的命令总数

函数原型	short **MCF_Get_Log_Data_Net** (unsigned long Count, short *Return, unsigned long *Code, unsigned long *Length, unsigned long *Data, unsigned short *StationNumber, char *Log);
使用说明	命令采集使能函数
参数说明	Count：采集到的命令号码 Return：该命令返回值 Code：该命令编号 Length：该命令参数长度(未开放) Data：该命令参数 StationNumber：该命令站号 Log：该命令文字记录

指令参数说明

```
/*****************************************************************************************************************
                                                0 全局宏定义
*****************************************************************************************************************/
//0.0 Axis
#define Axis_1                  0     //第 1 轴
#define Axis_2                  1     //第 2 轴
#define Axis_3                  2     //第 3 轴
#define Axis_4                  3     //第 4 轴
#define Axis_5                  4     //第 5 轴
#define Axis_6                  5     //第 6 轴
#define Axis_7                  6     //第 7 轴
#define Axis_8                  7     //第 8 轴
#define Axis_9                  8     //第 9 轴
#define Axis_10                 9     //第 10 轴
#define Axis_11                 10     //第 11 轴
#define Axis_12                 11     //第 12 轴
#define Axis_13                 12     //第 13 轴
#define Axis_14                 13     //第 14 轴
#define Axis_15                 14     //第 15 轴
#define Axis_16                 15     //第 16 轴
#define Axis_17                 16     //第 17 轴
#define Axis_18                 17     //第 18 轴
#define Axis_19                 18     //第 19 轴
#define Axis_20                 19     //第 20 轴
#define Axis_21                 20     //第 21 轴
#define Axis_22                 21     //第 22 轴
#define Axis_23                 22     //第 23 轴
#define Axis_24                 23     //第 24 轴

//0.1 Coordinate
#define Coordinate_0            0     //第 0 坐标系
#define Coordinate_1            1     //第 1 坐标系
//0.2 Buffer_Number
#define Buffer_Number_0         0     //第 0 缓冲区
//0.3 Position_Mode
#define Position_Absolute       0     //绝对位置模式
#define Position_Opposite       1     //相对位置模式
//0.4 Profile
#define Profile_T               0     //T 型曲线
```

```
#define Profile_S                       1       //S 型曲线
//0.5 Direction
#define Clock_Wise                      0       //顺弧
#define Counter_Clock_Wise              1       //逆弧
/*****************************************************************************************************************
                                    1 控制卡打开函数
*****************************************************************************************************************/
//1.0
#define Mode_Series                     0     //串联模式
#define Mode_Parallel                   1     //NTC0016并联模式
//1.1     Station_Type[]
#define Station_Type_24I16O             0     //NIO0808R/NIO1616R/NIO2416
#define Station_Type_48I32O             1     //NIO4832/NIO3232/NIO4000
#define Station_Type_4D                 2     //NMC1200R/NMC1400/NMC3400/NMC3401/NMC3201
#define Station_Type_8D                 3     //NMC5800/NMC5600R/NMC1800/NMC1600R
#define Station_Type_16D                4     //NMC5120R/NMC5160
#define Station_Type_DM                 5     //LMC3400/LMC3100
#define Station_Type_8E24I16O           6     //EIO0840
#define Station_Type_NAD0804            7     //NAD0804/NAD0402/NAD0808/NAD0400/NAD0004/NAD0002
#define Station_Type_NIO0040            8     //NIO0040
/*****************************************************************************************************************
                                    2 输入输出函数
*****************************************************************************************************************/
//2.3.1 Bit_Output_Number
#define Bit_Output_0                    0       //输出 0
#define Bit_Output_1                    1       //输出 1
#define Bit_Output_2                    2       //输出 2
#define Bit_Output_3                    3       //输出 3
#define Bit_Output_4                    4       //输出 4
#define Bit_Output_5                    5       //输出 5
#define Bit_Output_6                    6       //输出 6
#define Bit_Output_7                    7       //输出 7
#define Bit_Output_8                    8       //输出 8
#define Bit_Output_9                    9       //输出 9
#define Bit_Output_10                   10      //输出 10
#define Bit_Output_11                   11      //输出 11
#define Bit_Output_12                   12      //输出 12
#define Bit_Output_13                   13      //输出 13
#define Bit_Output_14                   14      //输出 14
#define Bit_Output_15                   15      //输出 15
#define Bit_Output_16                 16     //NMC3400/NMC5800 扩展卡输出 16
#define Bit_Output_17                 17     //NMC3400/NMC5800 扩展卡输出 17
#define Bit_Output_18                 18     //NMC3400/NMC5800 扩展卡输出 18
```

```
#define Bit_Output_19              19    //NMC3400/NMC5800 扩展卡输出 19
#define Bit_Output_20              20    //NMC3400/NMC5800 扩展卡输出 20
#define Bit_Output_21              21    //NMC3400/NMC5800 扩展卡输出 21
#define Bit_Output_22              22    //NMC3400/NMC5800 扩展卡输出 22
#define Bit_Output_23              23    //NMC3400/NMC5800 扩展卡输出 23
#define Bit_Output_24              24    //NMC3400/NMC5800 扩展卡输出 24
#define Bit_Output_25              25    //NMC3400/NMC5800 扩展卡输出 25
#define Bit_Output_26              26    //NMC3400/NMC5800 扩展卡输出 26
#define Bit_Output_27              27    //NMC3400/NMC5800 扩展卡输出 27
//2.3.2 Bit_Output_Logic
#define Bit_Output_Close              0     //低电平(触点闭合,硬件灯亮)
#define Bit_Output_Open               1     //高电平(触点断开,硬件灯灭)
//2.4.1 Bit_Input_Number
#define Bit_Input_0                   0     //输入 0
#define Bit_Input_1                   1     //输入 1
#define Bit_Input_2                   2     //输入 2
#define Bit_Input_3                   3     //输入 3
#define Bit_Input_4                   4     //输入 4
#define Bit_Input_5                   5     //输入 5
#define Bit_Input_6                   6     //输入 6
#define Bit_Input_7                   7     //输入 7
#define Bit_Input_8                   8     //输入 8
#define Bit_Input_9                   9     //输入 9
#define Bit_Input_10                  10    //输入 10
#define Bit_Input_11                  11    //输入 11
#define Bit_Input_12                  12    //输入 12
#define Bit_Input_13                  13    //输入 13
#define Bit_Input_14                  14    //输入 14
#define Bit_Input_15                  15    //输入 15
#define Bit_Input_16              16    //NMC3400 扩展卡输入 16
#define Bit_Input_17              17    //NMC3400 扩展卡输入 17
#define Bit_Input_18              18    //NMC3400/NMC5800 扩展卡输入 18
#define Bit_Input_19              19    //NMC3400/NMC5800 扩展卡输入 19
#define Bit_Input_20              20    //NMC3400/NMC5800 扩展卡输入 20
#define Bit_Input_21              21    //NMC3400/NMC5800 扩展卡输入 21
#define Bit_Input_22              22    //NMC3400/NMC5800 扩展卡输入 22
#define Bit_Input_23              23    //NMC3400/NMC5800 扩展卡输入 23
#define Bit_Input_24              24    //NMC3400/NMC5800 扩展卡输入 24
#define Bit_Input_25              25    //NMC3400/NMC5800 扩展卡输入 25
#define Bit_Input_26              26    //NMC3400/NMC5800 扩展卡输入 26
#define Bit_Input_27              27    //NMC3400/NMC5800 扩展卡输入 27
#define Bit_Input_28              28    //NMC3400/NMC5800 扩展卡输入 28
#define Bit_Input_29              29    //NMC3400/NMC5800 扩展卡输入 29
```

```
#define Bit_Input_30                30    //NMC3400/NMC5800 扩展卡输入 30
#define Bit_Input_31                31    //NMC3400/NMC5800 扩展卡输入 31
//2.4.2 Bit_Input_Logic
#define Bit_Input_Close                0     //低电平(触点闭合,硬件灯亮)
#define Bit_Input_Open                 1     //高电平(触点断开,硬件灯灭)
//2.7     Bit_Input_Fall
#define Bit_Input_Fall_Check           0     //下降沿(触点闭合,硬件灯亮瞬间)检查中
#define Bit_Input_Fall_Trigger         1     //下降沿(触点闭合,硬件灯亮瞬间)触发
/*****************************************************************************************************************
                                        3 轴专用输入输出函数
*****************************************************************************************************************/
//3.1 Servo_Logic
#define Servo_Close                    0     //低电平(触点闭合)
#define Servo_Open                     1     //高电平(触点断开)
//3.2 Alarm_Logic
#define Alarm_Close                    0     //低电平(触点闭合)
#define Alarm_Open                     1     //高电平(触点断开)
//3.3 Servo_Alarm_State
#define Servo_Alarm_Close              0     //低电平(触点闭合)
#define Servo_Alarm_Open               1     //高电平(触点断开)
//3.4 Servo_INP_State
#define Servo_INP_Close                0     //低电平(触点闭合)
#define Servo_INP_Open                 1     //高电平(触点断开)
//3.5 Z_State
#define Z_Logic_L                      0     //Z 相低电平(触点闭合)
#define Z_Logic_H                      1     //Z 相高电平(触点断开)
//3.6 Home_State
#define Home_Close                     0     //低电平(触点闭合,硬件灯亮)
#define Home_Open                      1     //高电平(触点断开,硬件灯灭)
//3.7 Positive_Limit_State
#define Positive_Limit_Close           0     //低电平(触点闭合,硬件灯亮)
#define Positive_Limit_Open            1     //高电平(触点断开,硬件灯灭)
//3.8 Negative_Limit_State
#define Negative_Limit_Close           0     //低电平(触点闭合,硬件灯亮)
#define Negative_Limit_Open            1     //高电平(触点断开,硬件灯灭)
/*****************************************************************************************************************
                                        4 轴设置函数
*****************************************************************************************************************/
//4.1 Pulse_Mode
#define Pulse_Dir_H                    0     //脉冲方向(默认)
#define Pulse_Dir_L                    1     //脉冲方向
#define Pulse_CW_CCW                   2     //双脉冲
#define Pulse_CCW_CW                   3     //双脉冲
```

```
#define Pulse_AB                4     //AB 相位
#define Pulse_BA                5     //AB 相位
/*************************************************************************************************************
                                    5 轴硬件触发停止运动函数
*************************************************************************************************************/
//5.1     EMG_Mode
#define EMG_Trigger_Close       0     //不使用紧急停止功能
#define EMG_Trigger_Low_IMD     1     //低电平(触点闭合,硬件灯亮)触发紧急停止
#define EMG_Trigger_Low_DEC     2     //低电平(触点闭合,硬件灯亮)触发减速停止
#define EMG_Trigger_High_IMD    3     //高电平(触点断开,硬件灯灭)触发紧急停止
#define EMG_Trigger_High_DEC    4     //高电平(触点断开,硬件灯灭)触发减速停止
//5.3 Trigger_Mode
#define Soft_Limit_Close        0     //软件限位关闭
#define Soft_Limit_Open         1     //软件限位打开
//5.4 Trigger_Mode
#define Trigger_Close           0     //关闭电平触发(默认)
#define Trigger_Low_IMD         1     //低电平(触点闭合,硬件灯亮)触发紧急停止
#define Trigger_Low_DEC         2     //低电平(触点闭合,硬件灯亮)触发减速停止
#define Trigger_High_IMD        3     //高电平(触点断开,硬件灯灭)触发紧急停止
#define Trigger_High_DEC        4     //高电平(触点断开,硬件灯灭)触发减速停止
/*************************************************************************************************************
                                    6 轴回原点函数
*************************************************************************************************************/
//6.1.1 Limit_Logic
//6.1.2 Home_Logic
//6.1.3 Index_Logic
#define Low_Logic               0     //低电平(触点闭合,硬件灯亮)触发
#define High_Logic              1     //高电平(触点断开,硬件灯灭)触发
/*************************************************************************************************************
                                    7 点位运动控制函数
*************************************************************************************************************/
//7.7 Axis_Stop_Mode
#define Axis_Stop_IMD           0     //紧急停止
#define Axis_Stop_DEC           1     //减速停止
/*************************************************************************************************************
                                    8 插补运动控制函数
*************************************************************************************************************/
//8.8 Coordinate_Stop_Mode
#define Coordinate_Stop_IMD     0     //紧急停止
#define Coordinate_Stop_DEC     1     //减速停止
/*************************************************************************************************************
                                    9 缓冲区函数
*************************************************************************************************************/
```

```
//9.2 Coordinate_Stop_Mode
#define Buffer_Stop_IMD                 0       //紧急停止
#define Buffer_Stop_DEC                 1       //减速停止
//9.5 Velocity_Ratio_Enable
#define Velocity_Ratio_Clsoe            0       //速度倍率关闭
#define Velocity_Ratio_Open             1       //速度倍率打开
//9.16 Execute_Mode
#define Execute_Mode_0                  0       //按照用户曲线参数执行
#define Execute_Mode_1                  1       //根据两轴运动指令间的拐角做速度规划
/*************************************************************************************************************
                                10 示波器 10K 采样频率数据捕捉函数
*************************************************************************************************************/
//10.2 Capture_Mode
#define Capture_Keep                    0       //连续采样
#define Capture_Rise_Input_15           1      //输入 Bit_Input_15 上升沿触发捕抓开始
#define Capture_Fall_Input_15           2      //输入 Bit_Input_15 上升沿触发捕抓开始
//10.2 Capture_State
#define Data_Keep                       0       //数据更新
#define Data_Updata                     1       //数据保持
//10.7 Capture_Frequency
#define Frequency_1K                    1       //1K 采样周期
#define Frequency_2K                    2       //2K 采样周期
#define Frequency_5K                    5       //5K 采样周期
#define Frequency_10K                   10      //10K 采样周期
#define Frequency_50K                   50      //50K 采样周期 (目前只支持 NAD0X0X)
/*************************************************************************************************************
                                11 电子齿轮控制函数
*************************************************************************************************************/
//11.1.1 Follow_Source
#define Follow_Command                  0       //跟随命令
#define Follow_Encode                   1       //跟随编码器
//11.1.2 Dir
#define Dir_P_T_P                       0       //跟随正同方向走
#define Dir_N_T_N                       1       //跟随负同方向走
#define Dir_PN_T_PN                     2       //跟随正负同方向走
#define Dir_PN_T_P                      3       //跟随正负都往正方向走
#define Dir_PN_T_N                      4       //跟随正负都往负方向走
//11.2 Gear_Enable
#define Gear_Close                      0       //电子齿轮关闭
#define Gear_Open                       1       //电子齿轮打开
/*************************************************************************************************************
                                                12 位置比较输出函数
*************************************************************************************************************/
```

```
/*****************************************************************************************************************
                                                13 PWM 输出函数
*****************************************************************************************************************/
//13.1.1 Channel
#define PWM_Channel_1                   0       //第 1 通道
#define PWM_Channel_2                   1       //第 2 通道
//13.1.2 Enable
#define PWM_Disable                     0       //PWM 不使能
#define PWM_Enable                      1       //PWM 使能
//13.1.3 Output_Port_Config
#define PWM_Output_14                   14      //输出点 14
#define PWM_Output_15                   15      //输出点 15
#define PWM_AXIS_4_Pluse                22      //轴 4 脉冲输出点
#define PWM_AXIS_4_Dir                  23      //轴 4 方向输出点
//13.1.3 Output_Start_Logic
#define PWM_Start_Low_Logic             0       //PWM 默认触点闭合,硬件灯亮
#define PWM_Start_High_Logic            1       //PWM 默认触点打开,硬件灯灭
//13.3.1 *Finish
#define PWM_Free                        0       //PWM 输出空闲
#define PWM_Busy                        1       //PWM 输出忙
/*****************************************************************************************************************
                                                16 光源控制器函数
*****************************************************************************************************************/
//16.1.1 Channel
#define Light_Channel_1                 0       //第 1 通道
#define Light_Channel_2                 1       //第 2 通道
#define Light_Channel_3                 2       //第 3 通道
#define Light_Channel_4                 3       //第 4 通道
#define Light_Channel_5                 4       //第 5 通道
#define Light_Channel_6                 5       //第 6 通道
#define Light_Channel_7                 6       //第 7 通道
#define Light_Channel_8                 7       //第 8 通道
```

函数返回值查询表

函数名称	返回值																				
MCF_Open_Net	0	-19	-18	-16	-20	-21															
MCF_Close_Net	0																				
MCF_Set_Output_Net	0	-19	-18	-17																	
MCF_Get_Output_Net	0	-19	-18	-17																	
MCF_Set_Output_Bit_Net	0	-19	-18	-17	-9																
MCF_Get_Output_Bit_Net	0	-19	-18	-17	-9																
MCF_Get_Input_Net	0	-19	-18	-17																	
MCF_Get_Input_Bit_Net	0	-19	-18	-17	-8																
MCF_Set_EMG_Bit_Net	0	-19	-18	-17	-8																
MCF_Set_Pulse_Mode_Net	0	-19	-18	-17	-16																
MCF_Get_Pulse_Mode_Net	0	-19	-18	-17	-16																
MCF_Set_Soft_Limit_Net	0	-19	-18	-17	-16																
MCF_Get_Soft_Limit_Net	0	-19	-18	-17	-16																
MCF_Set_Soft_Limit_Enable_Net	0	-19	-18	-17	-16																
MCF_Get_Soft_Limit	0	-19	-18	-17	-16																

_Enable_Net																				
MCF_Set_Alarm _Trigger_Net	0	-19	-18	-17	-16															
MCF_Get_Alarm _Trigger_Net	0	-19	-18	-17	-16															
MCF_Set_Index _Trigger_Net	0	-19	-18	-17	-16															
MCF_Get_Index _Trigger_Net	0	-19	-18	-17	-16															
MCF_Set_Home _Trigger_Net	0	-19	-18	-17	-16															
MCF_Get_Home _Trigger_Net	0	-19	-18	-17	-16															
MCF_Set_ELP _Trigger_Net	0	-19	-18	-17	-16															
MCF_Get_ELP_Trigger _Net	0	-19	-18	-17	-16															
MCF_Set_ELN_Trigger _Net	0	-19	-18	-17	-16															
MCF_Get_ELN_Trigger _Net	0	-19	-18	-17	-16															
MCF_Search_Home_Set _Net	0	-19	-18	-17	-16															
MCF_Search_Home _Start_Net	0	-19	-18	-17	-16															
MCF_Search_Home_Stop _Net	0	-19	-18	-17	-16															
MCF_Search_Home_Get _State_Net	0	-19	-18	-17	-16															
MCF_JOG_Net	0	-19	-18	-17	-16					2	3	4	5	14	15	16	17	18	19	20	21
MCF_Uniaxial_dDist _Change_Net	0	-19	-18	-17	-16					2	3	4	5	14	15	16	17	18	19	20	21
MCF_Uniaxial_dMaxV _Change_Net	0	-19	-18	-17	-16															
MCF_Set_Axis_Profile _Net	0	-19	-18	-17	-16	-13														
MCF_Get_Axis_Profile _Net	0	-19	-18	-17	-16															
MCF_Uniaxial_Net	0	-19	-18	-17	-16				1	2	3	4	5	14	15	16	17	18	19	20	21

MCF_Set_Axis_Stop_Profile_Net	0	-19	-18	-17	-16	-13														
MCF_Get_Axis_Stop_Profile_Net	0	-19	-18	-17	-16															
MCF_Axis_Stop_Net	0	-19	-18	-17	-16															
MCF_Set_Coordinate_Profile_Net	0	-19	-18	-17				-10	-13											
MCF_Get_Coordinate_Profile_Net	0	-19	-18	-17				-10												
MCF_Arc2_Radius_Net	0	-19	-18	-17	-16	-15	-14	-10	1	2	3	4	5	14	15	16	17	18	19	20	21
MCF_Arc2_Centre_Net	0	-19	-18	-17	-16	-15	-14	-10	1	2	3	4	5	14	15	16	17	18	19	20	21
MCF_Line2_Net	0	-19	-18	-17	-16	-15	-14	-10	1	2	3	4	5	14	15	16	17	18	19	20	21
MCF_Line3_Net	0	-19	-18	-17	-16	-15	-14	-10	1	2	3	4	5	14	15	16	17	18	19	20	21
MCF_Line4_Net	0	-19	-18	-17	-16	-15	-14	-10	1	2	3	4	5	14	15	16	17	18	19	20	21
MCF_Set_Coordinate_Stop_Profile_Net	0	-19	-18	-17				-10	-13											
MCF_Get_Coordinate_Stop_Profile_Net	0	-19	-18	-17				-10												
MCF_Coordinate_Stop_Net	0	-19	-18	-17				-10												
MCF_Set_Servo_Enable_Net	0	-19	-18	-17	-16															
MCF_Get_Servo_Enable_Net	0	-19	-18	-17	-16															
MCF_Set_Servo_Alarm_Reset_Net	0	-19	-18	-17	-16															
MCF_Get_Servo_Alarm_Reset_Net	0	-19	-18	-17	-16															
MCF_Get_Servo_Alarm_Net	0	-19	-18	-17	-16															
MCF_Get_Servo_INP_Net	0	-19	-18	-17	-16															
MCF_Get_Z_Net	0	-19	-18	-17	-16															
MCF_Get_Home_Net	0	-19	-18	-17	-16															
MCF_Get_Positive_Limit_Net	0	-19	-18	-17	-16															

MCF_Get_Negative_Limit_Net	0	-19	-18	-17	-16															
MCF_Set_Position_Net	0	-19	-18	-17	-16															
MCF_Get_Position_Net	0	-19	-18	-17	-16															
MCF_Set_Encoder_Net	0	-19	-18	-17	-16															
MCF_Get_Encoder_Net	0	-19	-18	-17	-16															
MCF_Get_Vel_Net	0	-19	-18	-17	-16															
MCF_Clear_Axis_State_Net	0	-19	-18	-17	-16															
MCF_Get_Axis_State_Net	0	-19	-18	-17	-16															
MCF_Get_Trigger_Position_Net	0	-19	-18	-17	-16															
MCF_Set_Gear_Net	0	-19	-18	-17	-16															
MCF_Get_Gear_Net	0	-19	-18	-17	-16															
MCF_Set_Gear_Enable_Net	0	-19	-18	-17	-16															
MCF_Get_Gear_Enable_Net	0	-19	-18	-17	-16															
MCF_Set_Gear_Auto_Disable_Net	0	-19	-18	-17	-16															
MCF_Buffer_Set_Stop_Profile_Net	0	-19	-18	-17	-6	-13														
MCF_Buffer_Stop_Net	0	-19	-18	-17	-6															
MCF_Buffer_Change_Velocity_Ratio_Net	0	-19	-18	-17	-6															
MCF_Buffer_Start_Net	0	-19	-18	-17	-6															
MCF_Buffer_Set_Velocity_Ratio_Enable_Net	0	-19	-18	-17	-6															
MCF_Buffer_Set_Reduce_Ratio_Net	0	-19	-18	-17	-6			-3	-2											
MCF_Buffer_Set	0	-19	-18	-17	-6	-13		-3	-2											

_Profile_Net																				
MCF_Buffer_Uniaxial_Net	0	-19	-18	-17	-6		-5	-3	-2											
MCF_Buffer_Line2_Net	0	-19	-18	-17	-6		-5	-3	-2											
MCF_Buffer_Line3_Net	0	-19	-18	-17	-6		-5	-3	-2											
MCF_Buffer_Line4_Net	0	-19	-18	-17	-6		-5	-3	-2											
MCF_Buffer_Arc_Radius_Net	0	-19	-18	-17	-6		-5	-3	-2											
MCF_Buffer_Arc_Centre_Net	0	-19	-18	-17	-6		-5	-3	-2											
MCF_Buffer_Delay_Net	0	-19	-18	-17	-6		-5	-3	-2											
MCF_Buffer_Set_Output_Bit_Net	0	-19	-18	-17	-6		-5	-3	-2											
MCF_Buffer_Wait_Input_Bit_Net	0	-19	-18	-17	-6		-5	-3	-2											
MCF_Buffer_End_Net	0	-19	-18	-17	-6			-3	-2											
MCF_Buffer_Execute_Net	0	-19	-18	-17	-6			-3	-2	-1										
MCF_Buffer_Execute_BreakPoint_Net	0	-19	-18	-17	-6			-3	-2	-1										
MCF_Buffer_Get_State_Net	0	-19	-18	-17	-6															
MCF_Capture_Open_Net	0	-19	-18	-17																
MCF_Capture_State_Net	0	-19	-18	-17																
MCF_Capture_Read_Command_Net	0	-19	-18	-17	-22															
MCF_Capture_Read_Encoder_Net	0	-19	-18	-17	-22															
MCF_Capture_Read_AD_Net	0	-19	-18	-17	-22															
MCF_Get_Version_Net	0	-19	-18	-17																

函数返回值

返回值	定义	意义
重复打开卡		
-24	ERR_Open_Already	控制卡已经打开
系统时间超时		
-23	ERR_Command_Time_Out	超时
数据捕获错误		
-22	ERR_Capture_Empty	数据捕获空
控制卡系统错误		
-21	ERR_Beyond_Station_Type	超出站点类型
-20	ERR_Beyond_Station_Length	超出站点长度
-19	ERR_Beyond_Station_Number	超出站点设置
-18	ERR_Open_Station_Fail	打开站点不成功
-17	ERR_Link_Break	链接中断
指令参数错误		
-16	ERR_Axis_Number	没有此轴号
-15	ERR_Axis_Inter_Number	表示轴数量超出设置范围
-14	ERR_NOT_Set_Profile	没有设置对应的规划参数
--13	ERR_Set_Profile_ERR	错误设置的规划参数
-12	ERR_Profile_Calculate	表示曲线参数计算出错
-11	ERR_Arc_Radius	MCF_Arc_Radius_Net 目标和源坐标不能设置为同意点
坐标系参数错误		
-10	ERR_Coordinate_Number	没有此坐标系
输入输出参数错误		
-9	ERR_Output_Number	没有此输出点
-8	ERR_Input_Number	没有此输入点
缓冲区参数错误		
-7	ERR_Buffer_Malloc_Fail	动态开辟本地线程内存失败
-6	ERR_Buffer_Number	没有此缓存
-5	ERR_Buffer_Space_Enough	缓冲区空间不足

-4	ERR_Buffer_Inter_Number	超出缓冲区最大插补轴
-3	ERR_Buffer_NO_Start	没有开启缓冲区
-2	ERR_Buffer_NO_Profile	没有设置缓冲曲线参数
-1	ERR_Buffer_NO_End_Buffer	没有结束缓冲区
正常状态		
0	Funtion_Success	正常命令执行成功
轴运动执行状态		
1	ERR_Axis_Busy	正在执行
2	IMD_STOP_AT_EMG	EMG 立即紧急停止
3	DEC_STOP_AT_EMG	EMG 减速紧急停止
4	IMD_STOP_AT_ALM	ALM 立即停止
5	DEC_STOP_AT_ALM	ALM 减速停止
6	IMD_STOP_AT_Servo	伺服使能立即停止
7	DEC_STOP_AT_Servo	伺服使能减速停止
8	IMD_STOP_AT_Pos_Error	指令编码器误差立即停止
9	DEC_STOP_AT_Pos_Error	指令编码器误差减速停止
10	IMD_STOP_AT_Index	Index 立即停止
11	DEC_STOP_AT_Index	Index 减速停止
12	IMD_STOP_AT_Home	原点立即停止
13	DEC_STOP_AT_Home	原点减速停止
14	IMD_STOP_AT_ELP	正硬限位立即停止
15	DEC_STOP_AT_ELP	正硬限位减速停止
16	IMD_STOP_AT_ELN	负硬限位立即停止
17	DEC_STOP_AT_ELN	负硬限位减速停止
18	IMD_STOP_AT_SOFT_ELP	正软限位立即停止
19	DEC_STOP_AT_SOFT_ELP	正软限位减速停止
20	IMD_STOP_AT_SOFT_ELN	负软限位立即停止
21	DEC_STOP_AT_SOFT_ELN	负软限位减速停止
22	IMD_STOP_AT_CMD	命令立即停止
23	DEC_STOP_AT_CMD	命令减速停止
24	IMD_STOP_AT_OTHER	其它原因立即停止
25	IMD_STOP_AT_LINK	网络通讯中断立即停止
26	IMD_STOP_AT_UNKOWN	未知原因立即停止

27	DEC_STOP_AT_UNKOWN	未知原因减速停止
28	DEC_STOP_AT_DEC	外部 IO 减速停止
缓冲区执行状态		
29	ERR_Buffer_Excute	缓冲区正在执行
30	ERR_Buffer_Stop	缓冲区停止
回零点执行状态		
31	ERR_Home_Wrong	回原点错误
32	ERR_Home_Excute	正在回原点
位置比较缓冲区状态		
33	ERR_Compare_Full	位置比较缓冲区满
PWM 通道超出范围		
34	ERR_PWM_Number	PWM 通道太大
命令限制		
35	ERR_Command_Limit	卡不支持此命令
筛选保护机制自动停止轴		
100	ERR_Input_0_TimeOut	物件检测无料超时停止轴运动
101	ERR_Trig_Blow_OK_TimeOut	物件吹气 OK 超时停止轴运动
102	ERR_Trig_Blow_NG_NumberOut	物件吹气连续 NG 停止轴运动
103	ERR_Sorting_Forbid_Command	筛选过程中禁止设置该函数